U-MVP: Encode Locally, Decode Globally for Feed-Forward 3D Gaussian Splatting

Seungkwon Yang* **Gyeongjin Kang*** **Hwasik Jeong**
Byeongjin Kang **Hyeongbin Cho** **Eunbyung Park†**

Yonsei University

*Equal contribution †Corresponding author

Abstract

Multi-view transformers for feed-forward 3D reconstruction must jointly solve two tasks. They need to extract precise per-view features and reason about cross-view geometric correspondences. Existing architectures distribute cross-view attention across the network without distinguishing the roles of individual layers, leaving each layer to handle both objectives together. We revisit this design through a layer-wise probing analysis of feed-forward geometry transformers and observe that trained models tend to divide these two roles across layers. Early layers tend to specialize in per-frame feature extraction, while spatial cognition and cross-view reasoning emerge predominantly in later layers. This suggests that separating the two roles across the network can allocate compute more effectively. Building on this observation, we propose U-MVP, a U-shaped multi-view transformer for feed-forward 3D Gaussian Splatting that reflects this structure in the architecture itself. The encoder applies frame-wise attention to preserve per-view detail, the decoder introduces multi-view attention for cross-view geometric reasoning, and skip connections fuse the two streams so that appearance and geometry are recombined at decoding time. To scale to dense view regimes, we replace several global interaction blocks at the bottleneck with a grouping and swapping scheme that preserves cross-view information flow at lower cost. U-MVP performs competitively with feed-forward and optimization-based baselines across a wide range of view settings, generalizes to unseen datasets, and achieves strong results on 3D reconstruction and novel view synthesis. Project page: `https://yang-gwon2.github.io/UMVP/`.

1 Introduction

Recovering the 3D structure of a scene from a collection of 2D images is a long-standing problem in computer vision, with applications ranging from robotics [12, 18] and autonomous driving [45, 34] to augmented reality [59] and 3D content creation [53, 63]. Images taken from different viewpoints provide complementary observations of the same scene, and its 3D structure is recovered by relating these observations to one another. This involves determining which pixels correspond, how the cameras are arranged, and where surfaces lie in space. Modeling interactions across views is therefore the common thread of stereo matching [51, 50], structure-from-motion [40], multi-view stereo [41], and novel view synthesis [35, 27].

Classical pipelines recover 3D structure in stages. They extract features from each image, match them across views, triangulate, and refine the result through iterative optimization. Recent learning-based methods instead collapse the entire pipeline into a single feed-forward network, and transformers [46] have proven especially well suited to this role. By treating patches from all input images as one token sequence, self-attention lets any patch attend to any other in any view, so correspondences are learned rather than engineered. Building on this property, recent models map image tokens directly to 3D

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

representations such as point clouds [49, 48, 30] or 3D Gaussians [64, 62, 24]. They reconstruct a scene in a single forward pass, without explicit matching or per-scene optimization.

Applying attention across all views and all layers, however, becomes a significant bottleneck as the number of input views grows, since its cost scales quadratically with the total number of tokens. This has motivated a growing body of work on scalable multi-view architectures, particularly for posed feed-forward 3D Gaussian Splatting (3DGS). Long-LRM [64] largely replaces attention with Mamba-based state-space models to achieve linear scaling, iLRM [23] routes cross-view interaction through a compact scene representation, and MVP [24] organizes attention into a dual hierarchy along the spatial and view axes. Through these efforts, the number of input views has grown from 4 in GS-LRM [62] to 256 in MVP, and feed-forward models now reach quality comparable to optimization-based 3DGS [27, 58, 33] within seconds.

These approaches make cross-view interaction cheaper; we ask instead where in the network it is most worth investing. Existing designs allocate cross-view attention uniformly across layers or by resolution stage, rather than according to the roles that individual layers play. To answer this question, we examine where the compute of a multi-view transformer currently goes and where cross-view reasoning tends to occur, which leads to two observations.

Where does a multi-view model spend its compute? In MVP, a considerable share of the compute is devoted to per-frame processing rather than to cross-view reasoning. The module that fuses multi-resolution features for the final prediction operates on each frame independently, yet it accounts for a substantial portion of the inference time, especially when only a few views are given. By comparison, the cross-view attention layers, which are central to multi-view reasoning, account for a relatively smaller part of the overall budget.

Where does cross-view reasoning occur? Our layer-wise probing of a multi-view transformer suggests that cross-view reasoning occurs mostly in later layers. Early layers tend to attend within each view and build per-frame features, whereas deeper layers become sensitive to cross-view geometric consistency and selectively gather information from other views. This trend holds across the tested view counts and agrees with prior analyses of learned geometric priors [6] and with designs that separate per-frame processing from cross-frame modeling [3, 30].

These observations motivate U-MVP, a U-shaped multi-view transformer that reorganizes the components of MVP around them. Its encoder extracts per-view features with frame-wise attention, its decoder performs cross-view reasoning as it restores spatial resolution, and lightweight skip connections in the spirit of U-Net [39] replace the aggregation module. The two paths complement each other: the frame-wise encoder preserves high-frequency appearance details, the cross-view decoder builds geometry-aware representations, and the skip connections let the final prediction draw on both. At the bottleneck, MVP still applies global attention across all views, whose cost grows quadratically with the number of views. We replace several of these layers with view-swapped group-wise attention, which preserves cross-view information flow at a lower cost.

Our contributions are summarized as follows.

- We analyze feed-forward multi-view transformers and find that a large share of MVP's compute goes to per-frame aggregation and that cross-view reasoning tends to emerge in later layers.
- We propose U-MVP, which reflects these observations through a frame-wise encoder, a cross-view decoder, skip-connected fusion, and a group-swap bottleneck.
- On DL3DV, U-MVP reaches higher PSNR than the compared feed-forward methods from 16 to 256 input views, and its PSNR is similar to or higher than that of optimization-based 3DGS. Results on unseen datasets are comparable to those of the baselines, and the method can also be applied in the unposed setting.

2 Method

2.1 Preliminaries

Problem setting. Given N input images $\{I_i\}_{i=1}^N$, where $I_i \in \mathbb{R}^{H\times W\times 3}$, with their camera parameters $\{\pi_i\}_{i=1}^N$, where each $\pi_i = (K_i, E_i)$ consists of the intrinsics K_i and the extrinsics $E_i \in SE(3)$, a feed-forward model f_θ predicts a set of 3D Gaussian primitives in a single forward pass,

$$\mathcal{G} = f_\theta\big(\{(I_i, \pi_i)\}_{i=1}^N\big) = \{(\mu_g, s_g, q_g, \alpha_g, c_g)\}_{g=1}^{N_\mathcal{G}}, \tag{1}$$

where each Gaussian is described by its position $\mu_g \in \mathbb{R}^3$, scale $s_g \in \mathbb{R}^3$, rotation quaternion $q_g \in \mathbb{R}^4$, opacity α_g, and color c_g, where the latter two are view-dependent. Following prior work [62, 24], we adopt pixel-aligned prediction, in which one Gaussian is predicted for each input pixel, so that $N_{\mathcal{G}} = N \times H \times W$. The predicted Gaussians are rendered into a target view with camera π_t by a differentiable rasterizer $\mathcal{R}$, yielding $\hat{I}_t = \mathcal{R}(\mathcal{G}, \pi_t)$. Because rendering is differentiable, f_θ is trained end-to-end by comparing $\hat{I}_t$ with the ground-truth image I_t over a set of target views.

Multi-view Pyramid Transformer. MVP [24] is a feed-forward model that reconstructs 3D Gaussians from N posed input images. Its core component is a dual attention hierarchy that operates along two complementary axes. Along the inter-view axis, the scope of attention widens over three stages, starting from frame-wise self-attention in Stage 1, moving to group-wise self-attention over consecutive subsets of M views in Stage 2, and ending with global self-attention across all N views in Stage 3. Along the intra-view axis, depthwise strided convolutions reduce the spatial resolution between stages, so that tokens in Stages 1, 2, and 3 correspond to patch sizes of 8, 16, and 32, while the hidden dimension doubles at each stage. Features from all three stages are then fused by a Pyramidal Feature Aggregation (PFA) module, which upsamples and merges them along a top-down residual path, and a linear head decodes the resulting tokens into per-pixel 3D Gaussian primitives [27]. This hierarchical design allows MVP to achieve competitive reconstruction quality at a low inference cost, and we adopt it as the starting point of our work.

2.2 Architectural Investigation

Feature aggregation. Although MVP is designed around multi-view attention, a considerable portion of its cost is spent elsewhere. As shown in Fig. 1, the convolutional PFA module, which operates on each frame independently, accounts for more than half of the total latency at every view count, reaching 66% at sparse views. The transformer blocks, which contain cross-view attention, take up only the remaining share. This suggests that the compute budget could be allocated more effectively. We therefore replace PFA with a light U-Net [39] style design, in which features from earlier encoder layers are fused into the decoder through skip connections at progressively finer resolutions (Sec. 2.3). This design aims to retain multi-scale aggregation at a lower cost, so that more of the budget can be directed toward the cross-view attention layers that support multi-view consistency.

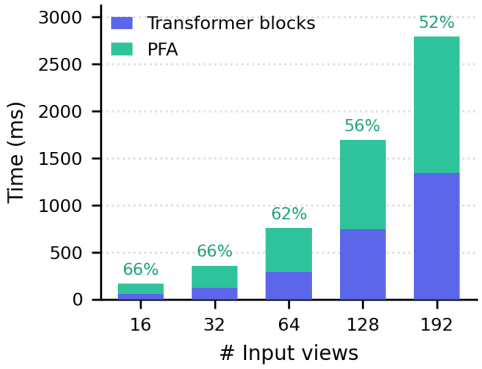

Figure 1: **Latency breakdown of MVP.** Inference time of the transformer blocks and the PFA module across input view counts on a single RTX PRO 6000 GPU. Much of the latency comes from the frame-wise PFA module rather than multi-view attention.

Layer-wise probing analysis. Inspired by work on decomposing temporal causality in video diffusion models [3] and on geometric priors in multi-view transformers [6], we study how global attention layers encode geometry through cross-view interaction. We train a 3DGS model for posed novel view synthesis on DL3DV [31] with 15 alternating attention [48] layers arranged as 1/2/9/2/1 across fine, mid, coarse, mid, and fine stages. The first and last layers, the only ones at the finest resolution, are frame-wise, giving one global layer in each mid stage and five in the coarse stage. To assess spatial cognition [29], we introduce one incorrect camera pose into the input image and camera pairs and test whether each layer can identify it. At every global layer, a lightweight linear head outputs one scalar per token; the scalars are averaged per view, and a softmax across views gives the probability of each view being the corrupted one. We also probe the first frame-wise layer (L0), which has no access to other views and thus serves as a noise floor.

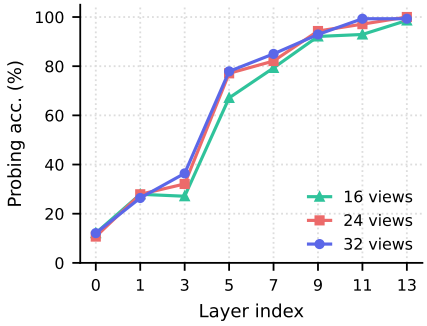

Figure 2: Pose inconsistency probing accuracy per layer. L0 is frame-wise and therefore serves as a noise floor.

As shown in Fig. 2, accuracy stays relatively close to this floor in the early layers and rises sharply at L5, approaching 100% in the last layers.

To see how this geometric awareness arises, we examine where each token attends in the other views. For a query token, we take the location in another view that receives the most attention, averaged over heads, and measure its distance to the epipolar line of the query given by the input camera poses. We report the fraction of queries within 16 pixels of the line, averaged over the test scenes. In the examples of Fig. 3 (left), the attended locations in early layers scatter across the target image, whereas in deeper layers they increasingly align with the epipolar lines, and the fraction of aligned locations rises accordingly (right). Both analyses show the same transition around L5, indicating that cross-view reasoning emerges progressively as layers get deeper, where most layers already operate at coarse resolution and global attention is cheap. We take these observations as suggestive motivation for a frame-wise encoder followed by a cross-view decoder, whose merits are ultimately validated empirically in Sec. 3.

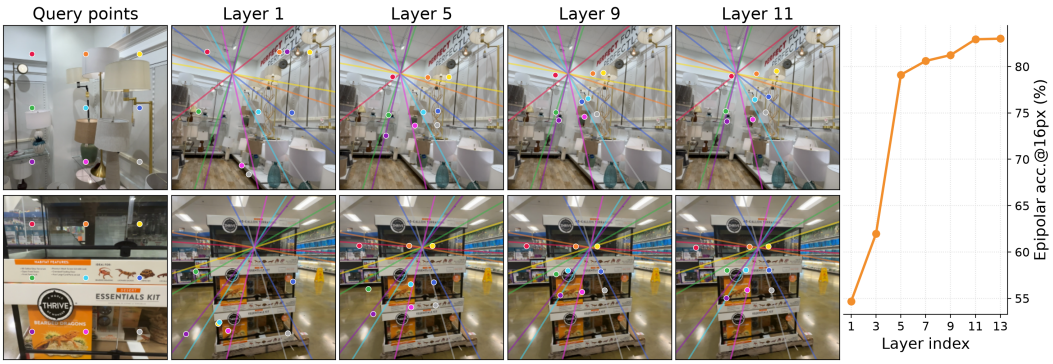

Figure 3: **Cross-view correspondence in global attention.** Left: most attended location for each query point, with its epipolar line (selected examples). Right: fraction of attended locations within 16 pixels of the epipolar line.

2.3 Overall Architecture

Building on the observations in Sec. 2.2, we design U-MVP so that per-view feature extraction and cross-view reasoning take place in separate parts of the network. To this end, U-MVP reorganizes the dual attention hierarchy of MVP into a U-Net [39] topology, as illustrated in Fig. 4. An encoding path processes each view independently with frame-wise self-attention at three resolution levels, and a decoding path introduces cross-view attention as it progressively restores spatial resolution. The two paths share a bottleneck at the coarsest resolution and are connected by skip connections, and a linear head decodes the final tokens into the 3D Gaussians of Eq. 1.

Image and pose encoding. For each input view, we compute a Plücker ray map $P_i \in \mathbb{R}^{H\times W\times 9}$ from the camera π_i, where each pixel's ray is represented by its origin o, direction d, and their cross product $o \times d$. The ray map is concatenated with the normalized image $I_i \in [-1,1]^{H\times W\times 3}$ along the channel dimension, and the resulting 12-channel tensor is divided into non-overlapping patches of size $p_1 \times p_1$ with $p_1 = 8$ and linearly projected into a grid of $h_1 \times w_1$ tokens of dimension d_1, where $h_1 = H/p_1$ and $w_1 = W/p_1$. We prepend four register tokens [11] to each view and encode spatial positions with relative positional embeddings [29, 44], computed at each resolution level.

Frame-wise encoder. The encoding path consists of three blocks $\mathcal{E}_1$, $\mathcal{E}_2$, and $\mathcal{E}_3$ with L_1^{enc}, L_2^{enc}, and L_3^{enc} layers. Block $\mathcal{E}_l$ operates on a grid of $h_l \times w_l$ image tokens per view with embedding dimension d_l, where $h_l = H/p_l$ and $w_l = W/p_l$, and each token covers an image region of $p_l \times p_l$ pixels with $p_l \in \{8, 16, 32\}$ for $l = 1, 2, 3$. All layers in the encoder apply self-attention within each view only, so the image tokens and register tokens of each view are processed independently of the other views. Between consecutive blocks, the image tokens of each view are reshaped into a 2D feature map and downsampled by a factor of 2 in each spatial dimension with a depthwise strided convolution, which also increases the embedding dimension from d_l to d_{l+1}. Each downsampling step therefore doubles the patch size p_l and reduces the number of image tokens per view to one quarter. At each step, the prepended register tokens are projected to d_{l+1} with a linear layer. Before each downsampling step, the feature maps $F_1^{\text{enc}} \in \mathbb{R}^{d_1\times h_1\times w_1}$ and $F_2^{\text{enc}} \in \mathbb{R}^{d_2\times h_2\times w_2}$ are kept for the skip connections to the decoder.

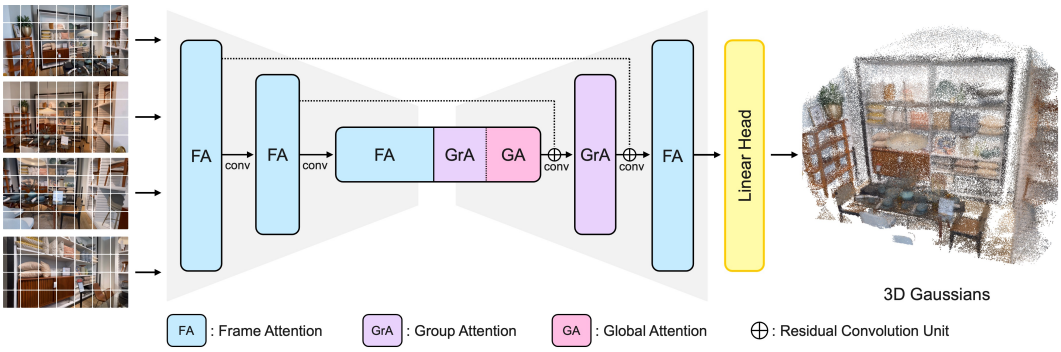

Figure 4: **Overview of U-MVP.** The encoder extracts multi-resolution per-view features with frame-wise attention. The decoder restores spatial resolution while performing group-wise, global, and frame-wise attention, and fuses encoder features through skip connections. A linear head then predicts pixel-aligned 3D Gaussians from the final tokens.

Cross-view decoder. The decoding path starts at the coarsest resolution with the block $\mathcal{D}_3$ of L_3^{dec} layers, which shares the resolution and embedding dimension d_3 of $\mathcal{E}_3$. Together, $\mathcal{E}_3$ and $\mathcal{D}_3$ form the bottleneck of the U-shaped network. $\mathcal{D}_3$ is where cross-view interaction is first introduced. We place it here because each view has only $h_3 \times w_3$ tokens at this level, keeping attention across views affordable. Within $\mathcal{D}_3$, cross-view attention operates at two scopes. Its first three layers apply group-wise attention, in which the N views are partitioned into consecutive groups of M_1 views and attention is computed within each group, so that their cost grows linearly rather than quadratically with N. To let information cross group boundaries, these layers alternate between the standard partition and a view-swapped partition, in which the last four views of each group, except the last, are exchanged with the first four views of the next group before attention and restored to their original order afterward (Fig. 5). The remaining two layers apply global attention across all N views, allowing full cross-view reasoning at the coarsest resolution.

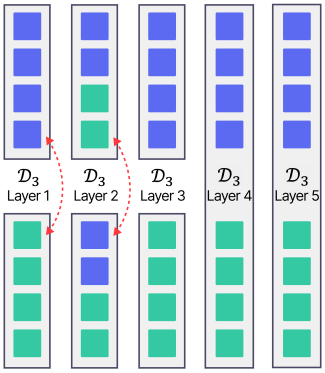

Figure 5: **Cross-view attention in $\mathcal{D}_3$.** In the first three layers of $\mathcal{D}_3$, the view-swapping permutation exchanges boundary views between adjacent groups before attention is computed.

The decoder then discards the register tokens and restores spatial resolution in two steps. We denote the output of decoder block $\mathcal{D}_j$ by F_j^{dec} and the decoder embedding dimensions at the mid and finest levels by d_4 and d_5. At each step, the feature map is upsampled by a factor of 2 with a transposed convolution and fused with the corresponding encoder feature map through a skip connection. The first fusion combines the upsampled bottleneck output F_3^{dec} with F_2^{enc} into $F_2^{\text{fused}} \in \mathbb{R}^{d_4 \times h_2 \times w_2}$, which is processed by the mid-level block $\mathcal{D}_2$ of L_2^{dec} layers with group-wise attention over groups of M_2 views. The second fusion combines the upsampled output F_2^{dec} with F_1^{enc} into $F_1^{\text{fused}} \in \mathbb{R}^{d_5 \times h_1 \times w_1}$, which is processed by the finest-level block $\mathcal{D}_1$ of L_1^{dec} frame-wise attention layers.

Skip fusion. Each skip connection fuses the per-view encoder features F_j^{enc} with the decoder features F_{j+1}^{dec} from the coarser level, which are first upsampled to the resolution of F_j^{enc},

$$F_j^{\text{fused}} = \text{SkipFuse}(F_{j+1}^{\text{dec}}, F_j^{\text{enc}}), \quad j \in \{1, 2\}, \tag{2}$$

$$\text{SkipFuse}(F_{j+1}^{\text{dec}}, F_j^{\text{enc}}) = \text{proj}\Big(\text{RCU}\big(\text{proj}_{\text{dec}}(F_{j+1}^{\text{dec}}) + \text{RCU}(\text{proj}_{\text{enc}}(F_j^{\text{enc}}))\big)\Big), \tag{3}$$

where proj_{dec} and proj_{enc} are 1×1 convolutions that project the two inputs to a common dimension, proj is a final 1×1 output projection, and RCU denotes a residual convolutional unit consisting of two 3×3 convolutions with LayerNorm [2] and GELU [15] activations and a residual connection.

Gaussian prediction. The output tokens of $\mathcal{D}_1$ are passed to a linear head, which maps each of the $h_1 \times w_1$ tokens to p_1^2 Gaussian attribute vectors, one for each pixel in its $p_1 \times p_1$ patch, so that every input pixel receives one Gaussian. Each vector contains the attributes defined in Eq. 1.

Following prior work [19, 24], the opacity α_g and color c_g are parameterized with spherical harmonic coefficients, which makes both of them view-dependent. When a Gaussian is rendered from a given direction, its opacity and color are obtained by evaluating these coefficients in that direction.

2.4 Training Objectives

The $N_{\mathcal{G}}$ predicted Gaussians are rendered into each target view $t \in \mathcal{T}$ with a differentiable rasterizer [27, 56], yielding $\hat{I}_t$. The training objective $\mathcal{L} = \mathcal{L}_{\text{img}} + \gamma \mathcal{R}_\alpha$ combines a rendering loss with an opacity regularizer:

$$\mathcal{L}_{\text{img}} = \sum_{t \in \mathcal{T}} \big(\mathcal{L}_{\text{MSE}}(\hat{I}_t, I_t) + \lambda \, \mathcal{L}_{\text{percept}}(\hat{I}_t, I_t) \big), \quad \mathcal{R}_\alpha = \frac{1}{N_{\mathcal{G}}} \sum_{g=1}^{N_{\mathcal{G}}} \big| \sigma(\alpha_g \cdot \omega_g) \big|, \tag{4}$$

with $\lambda = 0.2$ and $\gamma = 5 \times 10^{-4}$. Since view-dependent opacity can produce artifacts in novel views, we follow [19, 24] and regularize the opacity evaluated in randomly sampled directions: $\sigma(\cdot)$ is the sigmoid function, α_g denotes the spherical harmonic opacity coefficients of the g-th Gaussian, and ω_g is a direction sampled uniformly on the unit sphere.

3 Experiments

3.1 Setup

Implementation details. The full architectural hyperparameters are listed in Tab. 11. Following prior work [22, 24], our transformer layers use multi-head attention [46] with pre-normalization by LayerNorm [2], QK-normalization [16] with RMSNorm [61], and no bias terms for training stability. In cross-view attention layers, we use PRoPE [29], which injects relative camera poses directly into the attention mechanism, while frame-wise layers use only a normalized RoPE [44, 42] along the spatial dimensions. We further apply XSA [60] in all decoder layers.

Training. We train our model on DL3DV [31], a collection of real-world scene videos with camera poses obtained by COLMAP [40, 41]. We use 9,995 scenes in total, excluding those reserved for evaluation. Following MVP [24], training proceeds in three stages. We first train at 480×256 resolution with 32 input views, then move to 960×540 with 32 input views, and finally continue at 960×540 with the number of input views varying from 16 to 192. Training is performed on 16 B200 GPUs, and further details, including the schedule of the bottleneck attention layers across stages, are provided in App. C.1.

Evaluation. We evaluate on the DL3DV benchmark with 16 to 256 input views, and assess cross-dataset generalization on Tanks&Temples [28] and Mip-NeRF360 [4] with 32 to 128 input views. All evaluations are performed at 960×540 resolution, adjusted to 960×544 to align with the patch size; baselines that use a different resolution are noted in the table captions. Following Long-LRM [64], every 8th frame of each sequence is used as a target view, and input views are selected from the remaining frames by K-means clustering on camera poses combined with uniform sampling (see App. C.2), so the target views remain fixed as the number of input views changes. We report PSNR, SSIM, and LPIPS on the target views.

We compare against the feed-forward 3DGS methods Long-LRM [64], iLRM [23], MVP [24], and tttLRM [47], as well as optimization-based 3DGS [27] trained for 30k iterations. Unless otherwise noted, runtimes are measured on a single B200 GPU with FlashAttention-2 [10]. For MVP and ours, we report runtimes both in eager mode and with `torch.compile`, using the same compilation settings for both models and excluding compilation time. Settings that individual baselines cannot support are noted in the table captions.

3.2 Results

Comparison on DL3DV. Tab. 1 and Tab. 2 compare U-MVP with prior methods on the DL3DV benchmark. U-MVP achieves the best PSNR, SSIM, and LPIPS among feed-forward methods at every view count from 16 to 256. In PSNR, it also exceeds optimization-based 3DGS, while 3DGS retains slightly better LPIPS from 64 views and slightly better SSIM from 128 views. With compilation, U-MVP is the fastest among the compared methods at every view count, although it is tied with MVP

Method	16 views				32 views				64 views			
	PSNR↑	SSIM↑	LPIPS↓	Time(s)↓	PSNR↑	SSIM↑	LPIPS↓	Time(s)↓	PSNR↑	SSIM↑	LPIPS↓	Time(s)↓
3D-GS$_{30k}$	21.48	0.753	0.252	8min	24.43	0.827	0.191	8min	27.34	0.883	0.146	8min
Long-LRM	21.05 (22.66)	0.708 (0.740)	0.297 (0.292)	0.19	23.97	0.778	0.267	0.48	23.60	0.789	0.260	1.55
iLRM	21.92 (22.91)	0.748 (0.766)	0.316 (0.295)	0.24	24.30	0.803	0.256	0.71	24.44	0.819	0.240	2.23
MVP	23.76	0.798	0.239	**0.05** / **0.03**	25.96	0.847	0.187	**0.11** / 0.06	27.73	0.881	0.154	**0.22** / 0.13
tttLRM	23.61 (23.60)	0.777 (0.784)	0.253 (0.255)	1.12	25.06 (25.07)	0.818 (0.822)	0.216 (0.215)	2.22	25.95	0.840	0.196 (0.195)	4.54
Ours	**23.91**	**0.803**	**0.238**	0.06 / **0.03**	**26.16**	**0.852**	**0.185**	0.12 / **0.05**	**28.12**	**0.888**	**0.150**	0.23 / **0.11**

Table 1: **Quantitative comparisons on DL3DV (16–64 views).** Best results among feed-forward methods are in **bold**. Numbers in parentheses are those reported in the original papers. Time is reported as eager / compiled (`torch.compile`) for MVP and ours, and in eager mode for other baselines; the best time is marked separately for each mode. tttLRM is evaluated at its native training resolution of 960×536.

Method	128 views				192 views				256 views			
	PSNR↑	SSIM↑	LPIPS↓	Time(s)↓	PSNR↑	SSIM↑	LPIPS↓	Time(s)↓	PSNR↑	SSIM↑	LPIPS↓	Time(s)↓
3D-GS$_{30k}$	29.43	0.914	0.123	8min	30.14	0.924	0.116	8min	30.39	0.926	0.114	8min
Long-LRM	21.24	0.739	0.308	5.58	N/A	N/A	N/A	11.4	N/A	N/A	N/A	19.6
iLRM	22.98	0.807	0.249	7.85	21.68	0.788	0.264	16.9	20.63	0.767	0.281	29.4
MVP	29.02	0.903	0.134	**0.46** / 0.29	29.54	0.912	0.127	**0.74** / 0.50	29.67	0.915	0.125	1.13 / 0.81
tttLRM	25.42	0.848	0.193	9.23	24.18	0.833	0.206	N/A	23.32	0.816	0.219	N/A
Ours	**29.62**	**0.912**	**0.128**	0.48 / **0.23**	**30.23**	**0.921**	**0.119**	**0.74** / **0.38**	**30.46**	**0.925**	**0.117**	**1.03** / **0.55**

Table 2: **Quantitative comparisons on DL3DV (128–256 views).** Bold and time are reported as in Tab. 1. Long-LRM runs at 192 and 256 views but its released checkpoint produces blank renderings, so only its runtime is reported. tttLRM exceeds the memory of a single GPU beyond 128 views and requires multi-GPU inference, so its runtime is not comparable.



Figure 6: **Qualitative comparison on DL3DV.** All methods use 32 input views, which matches the training setting of the feed-forward baselines.

at 16 views at the precision reported in Tab. 1; in eager mode, its runtime is comparable to that of MVP. The gain over MVP also grows with the number of views, from 0.15 dB in PSNR at 16 views to 0.79 dB at 256 views, whereas Long-LRM, iLRM, and tttLRM degrade beyond 32 or 64 views, partly because they are trained with fewer views.

Cross-dataset generalization. Tab. 3 reports results on Tanks&Temples and Mip-NeRF360, which are not used for training. U-MVP achieves the best or tied-best PSNR and the best SSIM on both datasets at all view counts, and the gap to MVP again widens as the number of views increases. MVP shows slightly lower LPIPS at 32 and 64 views, while U-MVP is better at 128 views.

Qualitative results. Fig. 6 and Fig. 7 show qualitative comparisons with 32 input views. The differences between methods concentrate on high-frequency structures and weakly observed regions, where U-MVP recovers sharper details with fewer artifacts.

Compute allocation. To examine whether U-MVP shifts compute from feature aggregation to the transformer blocks, we profile it with the same decomposition and protocol as in Fig. 1. We separate the transformer blocks from the multi-scale aggregation path, which in U-MVP consists of the skip fusion modules and transposed convolutions that replace the PFA module of MVP. All measurements are taken on a single RTX PRO 6000 GPU and averaged over 10 runs after 3 warmup iterations. As shown in Tab. 4, the allocation is largely reversed relative to MVP. The transformer blocks account for roughly 70% of the inference time at every view count, while the aggregation path takes 26–30%.

Method	Tanks&Temples									Mip-NeRF360								
	32 views			64 views			128 views			32 views			64 views			128 views		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Long-LRM	18.59	0.614	0.366	19.44	0.651	0.334	18.47	0.613	0.375	21.08	0.484	0.445	21.30	0.499	0.431	19.82	0.484	0.457
iLRM	18.58	0.631	0.385	19.82	0.692	0.318	19.22	0.696	0.319	21.09	0.495	0.466	21.60	0.522	0.444	21.32	0.551	0.424
MVP	**19.54**	0.708	**0.277**	21.24	0.761	**0.221**	22.36	0.804	0.184	22.21	0.587	**0.355**	23.72	0.656	**0.302**	25.12	0.736	0.248
tttLRM	19.27	0.660	0.304	20.43	0.704	0.270	20.67	0.728	0.256	22.20	0.555	0.373	23.02	0.600	0.344	23.16	0.628	0.327
Ours	**19.54**	**0.711**	0.281	**21.41**	**0.769**	0.223	**22.89**	**0.818**	**0.181**	**22.31**	**0.590**	0.360	**23.89**	**0.658**	0.305	**25.41**	**0.741**	**0.247**

Table 3: **Cross-dataset generalization on Tanks&Temples and Mip-NeRF360.** All models are trained on DL3DV only. Bold indicates the best result.



Figure 7: **Qualitative comparison on Tanks&Temples (top two rows) and Mip-NeRF360 (bottom two rows).** All methods use 32 input views.

3.3 Additional Experiments

Gaussian redundancy. Pixel-aligned Gaussian primitives grow in number proportionally with the number of input views, reaching 133.7M primitives per scene at 256 views. Most of them are redundant: for a median pixel, its opacity is carried by about 125 primitives from about 13 different input views. Models with view-independent opacity can remove such primitives globally by thresholding their opacity [64]. With view-dependent opacity, however, a primitive has no single opacity value, so low-opacity primitives can only be culled per view at render time [24], which leaves the stored set of primitives unchanged.

We instead select a global subset with a per-scene procedure that keeps the network fixed. We attach a gate $m_i \in [0, 1]$ to the opacity of each primitive, render the input views, and score each primitive by its *removal sensitivity* $s_i = \sum_c |\partial \mathcal{L}_c / \partial m_i|$ at $m_i = 1$, where $\mathcal{L}_c$ is the L2 loss between the rendered and the given images of the c-th group of eight input views; backpropagation yields the scores of all primitives at once. Since the importance of a primitive depends on which others remain, removing many low-sensitivity primitives at once would delete redundant copies of the same surface together. We therefore remove primitives over eight rounds on a geometric schedule toward the target keep ratio, re-scoring the survivors after each round. Finally, as the survivors no longer accumulate the opacity of the removed primitives, we add a single scalar b to the DC opacity coefficient of all survivors. It is predicted in closed form from the fraction ρ of primitives kept, $b = 0.40 \ln(1/\rho)$, with the constant fitted once to the input-view grid-search optima of 50 scenes, and after the last round refined by a small grid search around this prediction on the input views. This is applied only after selection: the scores are always computed on the uncompensated survivors, whose brightness deficit is exactly what the selection follows. Neither step updates the network weights or accesses the target views.

Tab. 5 reports the model after round 7, the round closest to a -0.5 dB PSNR drop. At this point, the model keeps 19.0–25.5% of the primitives at a cost of about 0.5 dB PSNR, and this fraction stays nearly constant from 64 views upward. At 256 views, the final round keeps 21.0% of the primitives (133.7M→28.1M) at a cost of 0.57 dB PSNR, while rendering 2.32× faster. Since pruning relies on gradient computation and takes 149 s per scene at 256 views, it serves as an offline compaction step for storage and rendering rather than as part of the feed-forward reconstruction.

Model	Component	16	32	64	128	192
MVP	Transformer blocks	34%	34%	38%	44%	48%
	Aggregation	66%	66%	62%	56%	52%
U-MVP	Transformer blocks	72%	70%	71%	72%	74%
	Aggregation	28%	30%	29%	28%	26%

Table 4: **Share of inference time by component.** The transformer blocks and the aggregation path of MVP and U-MVP are timed separately across input view counts.

# Input views	16	32	64	128	192	256
Full primitives	8.4M	16.7M	33.4M	66.8M	100.3M	133.7M
Retained primitives	1.6M	3.7M	8.2M	16.3M	25.6M	34.1M
Retained ratio	19.0%	22.3%	24.5%	24.5%	25.5%	25.5%
ΔPSNR$\uparrow$	-0.50	-0.52	-0.53	-0.54	-0.49	-0.48
ΔSSIM$\uparrow$	-0.0245	-0.0212	-0.0176	-0.0148	-0.0116	-0.0101
ΔLPIPS$\downarrow$	$+0.0640$	$+0.0584$	$+0.0507$	$+0.0439$	$+0.0355$	$+0.0307$
FPS	202.0	169.8	124.5	112.9	77.0	62.4
Speedup	1.26×	1.61×	1.79×	2.07×	2.03×	1.98×
Pruning time (s)	1.5	4.3	12.3	33.8	74.8	120.4

Table 5: **Pruning by removal sensitivity on DL3DV (140 scenes).** Results after round 7, the round closest to a -0.5 dB PSNR drop, where the compensation scalar is the closed-form prediction (the grid search runs only after the last round). Changes and speedups are relative to the unpruned model of the same scene. FPS and pruning time are measured on a single B200 GPU; pruning time is cumulative up to round 7.

Pose-free reconstruction. We further examine whether U-MVP is compatible with the pose-free setting. Following 2Xplat [20], we pair our backbone with a camera pose regression module [30] and evaluate on DL3DV at 960×540 with 16 to 128 input views, applying the same evaluation-time pose alignment. The predicted poses replace the ground-truth poses as the camera input to our backbone, and the backbone is adapted to them with a short finetuning stage. As shown in Tab. 6, 2Xplat with U-MVP performs comparably to 2Xplat with MVP at sparse view counts and improves upon it as the number of views increases, following the same trend as in the posed setting.

Method	GT pose	16 views			32 views			64 views			128 views		
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
MVP	✓	23.76	0.798	0.239	25.96	0.847	0.187	27.73	0.881	0.154	29.02	0.903	0.134
U-MVP (Ours)	✓	23.91	0.803	0.238	26.16	0.852	0.185	28.12	0.888	0.150	29.62	0.912	0.128
2Xplat (MVP)		23.61	0.786	**0.248**	25.66	**0.832**	**0.198**	27.12	0.859	0.171	28.30	0.880	0.153
2Xplat (U-MVP)		**23.68**	**0.787**	0.249	**25.72**	**0.832**	0.199	**27.37**	**0.863**	**0.170**	**28.69**	**0.885**	**0.151**

Table 6: **Pose-free reconstruction on DL3DV.** The top rows use ground-truth camera poses as input and are shown for reference. The bottom rows estimate poses with the 2Xplat framework [20] using either MVP or U-MVP as the backbone. Bold indicates the better pose-free result.

Pointmap estimation. We also examine how the bottleneck design behaves beyond novel view synthesis by applying it to pointmap estimation, which involves both depth prediction and camera pose estimation. For this experiment, we train on BlendedMVS [54], ScanNet++ [57], and ARKitScenes [5], following the BlendedMVS training and validation splits of MapAnything [26]. We compare three bottleneck variants that use only global attention, only group-wise attention with view swapping, or our hybrid of the two. All variants are trained at a reduced scale under identical settings to allow a controlled comparison.

As shown in Tab. 7 and Tab. 8, using only global attention gives the best accuracy across most metrics, as expected from its unrestricted cross-view interaction. Using only group-wise attention leads to a clear drop, particularly in completeness. The hybrid design recovers much of this gap on several metrics, such as accuracy and completeness on NRGBD, and monocular depth on Bonn, while replacing most global attention layers with cheaper group-wise ones. However, it remains behind global attention in pose accuracy on TUM and normal consistency on NRGBD. The hybrid bottleneck retains most of the accuracy of full global attention at a lower cost, while pose estimation, which relies more on global cross-view reasoning, favors full global attention.

Bottleneck	BMVS [54]			NRGBD [1]		
	Acc.↓	Comp.↓	NC.↑	Acc.↓	Comp.↓	NC.↑
Global	1.76	3.91	0.631	0.154	0.145	0.492
Group swap	2.24	6.85	0.618	0.195	0.283	0.488
Hybrid (Ours)	2.00	6.55	0.618	0.155	0.188	0.464

Table 7: **Pointmap reconstruction** with different bottleneck designs.

Bottleneck	TUM [43] pose			Bonn [36] monocular depth		
	ATE↓	RPE trans.↓	RPE rot.↓	AbsRel↓	RMSE↓	$\delta < 1.25$↑
Global	0.032	0.026	1.91	0.258	0.574	0.718
Group swap	0.039	0.029	2.20	0.265	0.596	0.692
Hybrid (Ours)	0.041	0.028	2.12	0.259	0.588	0.698

Table 8: **Pose and monocular depth estimation** with different bottleneck designs.

Method	Frame-wise Attn.	Multi-view Attn.	Skip	Global	Group swap	vdo	PSNR↑	SSIM↑	LPIPS↓
Baseline	$\mathcal{E}$	$\mathcal{D}$	RCU	o	o	o	24.55	0.801	0.191
Alternating attn.	$\mathcal{E}, \mathcal{D}$	$\mathcal{E}, \mathcal{D}$	RCU	o	o	o	24.38	0.794	0.196
Reversed attn.	$\mathcal{D}$	$\mathcal{E}$	RCU	o	o	o	24.34	0.792	0.195
w/o Skip	$\mathcal{E}$	$\mathcal{D}$	x	o	o	o	24.57	0.779	0.235
Cat. + linear skip	$\mathcal{E}$	$\mathcal{D}$	Cat. + linear	o	o	o	24.47	0.796	0.199
w/o Global	$\mathcal{E}$	$\mathcal{D}$	RCU	x	o	o	24.52	0.802	0.191
+ w/o Group swap	$\mathcal{E}$	$\mathcal{D}$	RCU	x	x	o	24.33	0.797	0.195
w/o View-dependent opacity	$\mathcal{E}$	$\mathcal{D}$	RCU	o	o	x	22.48	0.717	0.283

Table 9: **Ablation studies.**

3.4 Ablation studies

We conduct ablation experiments at a reduced resolution of 256×256 with 16 input views to validate the key architectural components of our encoder-decoder design and view-dependent opacity (Tab. 9). The baseline model shares the same layer architecture as the final model but is scaled down in capacity, providing a reference for each variant.

Multi-view attention placement. We experiment with two variants here, one with alternating attention across both encoder and decoder, and the other placing multi-view attention in the encoder and frame-wise attention in the decoder, which is essentially the inverse of our final model design. Both variants cause a drop in quantitative metrics, which suggests that the assignment of attention types across the encoder and decoder is a critical design choice.

Skip connections. Removing encoder-to-decoder skip connections causes a sharp increase in LPIPS, demonstrating their necessity for recovering fine-grained detail that cannot be reconstructed by the decoder alone. Furthermore, replacing our RCU-based skip connection with simple channel-wise concatenation followed by linear projection underperforms, indicating that the non-linear residual fusion more effectively integrates encoder features into the decoder path.

Decoder bottleneck design. We progressively ablate $\mathcal{D}_3$ blocks by first removing the global attention layers, then additionally removing the view-swapping from the remaining group-wise attention layers. Both steps lead to monotonic degradation, with the larger drop occurring when permutation is also removed, confirming that both global attention and group swap contribute independently to cross-view reasoning at the bottleneck.

View-dependent opacity. Following MVP [24], we adopt the spherical-harmonic opacity of LVT [19], and verify its effectiveness in our encoder-decoder design by replacing it with a view-independent scalar. This yields the largest degradation among all ablations (PSNR -2.07 dB, LPIPS 0.191→0.283), consistent with LVT. We attribute this to the redundancy of pixel-aligned Gaussians: primitives from different views overlap without perfect alignment, and view-dependent opacity allows the model to suppress them from unreliable viewing directions, whereas a constant opacity blends them, causing ghosting and blur.

4 Conclusion

We presented U-MVP, a multi-view transformer for feed-forward 3D Gaussian Splatting reconstruction that revisits the architectural foundations of MVP through a systematic investigation of computational efficiency and geometric understanding in multi-view transformers. Guided by these analyses, we redesign the architecture into a U-Net topology with an explicit separation of per-view feature extraction and cross-view geometric reasoning, enabling more effective use of the compute budget. U-MVP matches or surpasses both feed-forward and optimization-based baselines across a wide range of input views, generalizes robustly to unseen datasets, and, with compilation, is at least as fast as MVP at every tested view count. We hope that our architectural investigation and the insights derived from it provide a useful foundation for future work on scalable, high-quality feed-forward 3D reconstruction.

Acknowledgments and Disclosure of Funding

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) under the following projects: (No. RS-2024-00457882, AI Research Hub Project); (No. RS-2025-25441838, Development of a human foundation model for human-centric universal artificial intelligence and training of personnel); and (No. RS-2026-25517417, Development of Compression, Reconstruction, and Rendering Technologies for Free-viewpoint Media). This research was also supported by the Advanced GPU Utilization Support Program funded by the Government of the Republic of Korea (Ministry of Science and ICT) and by the AI Seoul Tech Research Support Program of the Seoul Future Foundation.

References

[1] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgb-d surface reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6290–6301, 2022.

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

[3] Xingjian Bai, Guande He, Zhengqi Li, Eli Shechtman, Xun Huang, and Zongze Wu. Causality in video diffusers is separable from denoising. *arXiv preprint arXiv:2602.10095*, 2026.

[4] 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.

[5] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, Thomas Gebauer, Brandon Joffe, Daniel Kurz, Arik Schwartz, et al. Arkitscenes: A diverse real-world dataset for 3d indoor scene understanding using mobile rgb-d data. *arXiv preprint arXiv:2111.08897*, 2021.

[6] Jelena Bratulić, Sudhanshu Mittal, Thomas Brox, and Christian Rupprecht. On geometric understanding and learned data priors in vggt. *arXiv preprint arXiv:2512.11508*, 2025.

[7] Hu Cao, Yueyue Wang, Joy Chen, Dongsheng Jiang, Xiaopeng Zhang, Qi Tian, and Manning Wang. Swin-unet: Unet-like pure transformer for medical image segmentation. In *European conference on computer vision*, pages 205–218. Springer, 2022.

[8] 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.

[9] 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.

[10] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning, 2023. *arXiv preprint arXiv:2307.08691*, 2023.

[11] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. *arXiv preprint arXiv:2309.16588*, 2023.

[12] Alejandro Escontrela, Justin Kerr, Arthur Allshire, Jonas Frey, Rocky Duan, Carmelo Sferrazza, and Pieter Abbeel. Gaussgym: An open-source real-to-sim framework for learning locomotion from pixels. *arXiv preprint arXiv:2510.15352*, 2025.

[13] Ali Hatamizadeh, Jiaming Song, Guilin Liu, Jan Kautz, and Arash Vahdat. Diffit: Diffusion vision transformers for image generation. In *European Conference on Computer Vision*, pages 37–55. Springer, 2024.

[14] David Haynes, Steven Corns, and Ganesh Kumar Venayagamoorthy. An exponential moving average algorithm. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–8, 2012.

[15] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[16] Alex Henry, Prudhvi Raj Dachapally, Shubham Shantaram Pawar, and Yuxuan Chen. Query-key normalization for transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4246–4253, 2020.

[17] Sunghwan Hong, Jaewoo Jung, Heeseong Shin, Jisang Han, Jiaolong Yang, Chong Luo, and Seungryong Kim. Pf3plat: Pose-free feed-forward 3d gaussian splatting. *arXiv preprint arXiv:2410.22128*, 2024.

[18] Wenlong Huang, Yu-Wei Chao, Arsalan Mousavian, Ming-Yu Liu, Dieter Fox, Kaichun Mo, and Li Fei-Fei. Pointworld: Scaling 3d world models for in-the-wild robotic manipulation. *arXiv preprint arXiv:2601.03782*, 2026.

[19] Tooba Imtiaz, Lucy Chai, Kathryn Heal, Xuan Luo, Jungyeon Park, Jennifer Dy, and John Flynn. Lvt: Large-scale scene reconstruction via local view transformers. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pages 1–12, 2025.

[20] Hwasik Jeong, Seungryong Lee, Gyeongjin Kang, Seungkwon Yang, Xiangyu Sun, Seungtae Nam, and Eunbyung Park. 2xplat: Two experts are better than one generalist. *arXiv preprint arXiv:2603.21064*, 2026.

[21] 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.

[22] Haian Jin, Hanwen Jiang, Hao Tan, Kai Zhang, Sai Bi, Tianyuan Zhang, Fujun Luan, Noah Snavely, and Zexiang Xu. Lvsm: A large view synthesis model with minimal 3d inductive bias. *arXiv preprint arXiv:2410.17242*, 2024.

[23] Gyeongjin Kang, Seungtae Nam, Seungkwon Yang, Xiangyu Sun, Sameh Khamis, Abdelrahman Mohamed, and Eunbyung Park. ilrm: An iterative large 3d reconstruction model. *arXiv preprint arXiv:2507.23277*, 2025.

[24] Gyeongjin Kang, Seungkwon Yang, Seungtae Nam, Younggeun Lee, Jungwoo Kim, and Eunbyung Park. Multi-view pyramid transformer: Look coarser to see broader. *arXiv preprint arXiv:2512.07806*, 2025.

[25] Gyeongjin Kang, Jisang Yoo, Jihyeon Park, Seungtae Nam, Hyeonsoo Im, Sangheon Shin, Sangpil Kim, and Eunbyung Park. Selfsplat: Pose-free and 3d prior-free generalizable 3d gaussian splatting. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22012–22022, 2025.

[26] Nikhil Keetha, Norman Müller, Johannes Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, et al. Mapanything: Universal feed-forward metric 3d reconstruction. *arXiv preprint arXiv:2509.13414*, 2025.

[27] 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, 2023.

[28] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017.

[29] Ruilong Li, Brent Yi, Junchen Liu, Hang Gao, Yi Ma, and Angjoo Kanazawa. Cameras as relative positional encoding. *arXiv preprint arXiv:2507.10496*, 2025.

[30] 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.

[31] 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.

[32] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[33] 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.

[34] Yifan Lu, Xuanchi Ren, Jiawei Yang, Tianchang Shen, Zhangjie Wu, Jun Gao, Yue Wang, Siheng Chen, Mike Chen, Sanja Fidler, et al. Infinicube: Unbounded and controllable dynamic 3d driving scene generation with world-guided video models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 27272–27283, 2025.

[35] 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. *Communications of the ACM*, 65(1):99–106, 2021.

[36] Emanuele Palazzolo, Jens Behley, Philipp Lottes, Philippe Giguere, and Cyrill Stachniss. Refusion: 3d reconstruction in dynamic environments for rgb-d cameras exploiting residuals. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7855–7862. IEEE, 2019.

[37] Olivier Petit, Nicolas Thome, Clement Rambour, Loic Themyr, Toby Collins, and Luc Soler. U-net transformer: Self and cross attention for medical image segmentation. In *International Workshop on Machine Learning in Medical Imaging*, pages 267–276. Springer, 2021.

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

[39] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[40] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[41] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

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

[43] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580, 2012.

[44] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[45] Qijian Tian, Xin Tan, Yuan Xie, and Lizhuang Ma. Drivingforward: Feed-forward 3d gaussian splatting for driving scene reconstruction from flexible surround-view input. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 7374–7382, 2025.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[47] Chen Wang, Hao Tan, Wang Yifan, Zhiqin Chen, Yuheng Liu, Kalyan Sunkavalli, Sai Bi, Lingjie Liu, and Yiwei Hu. tttlrm: Test-time training for long context and autoregressive 3d reconstruction. *arXiv preprint arXiv:2602.20160*, 2026.

[48] 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.

[49] 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.

[50] Bowen Wen, Shaurya Dewan, and Stan Birchfield. Fast-foundationstereo: Real-time zero-shot stereo matching. *arXiv preprint arXiv:2512.11130*, 2025.

[51] Bowen Wen, Matthew Trepte, Joseph Aribido, Jan Kautz, Orazio Gallo, and Stan Birchfield. Foundationstereo: Zero-shot stereo matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5249–5260, 2025.

[52] 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.

[53] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024.

[54] 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.

[55] Botao Ye, Sifei Liu, Haofei Xu, Li Xueting, Marc Pollefeys, Ming-Hsuan Yang, and Peng Songyou. No pose, no problem: Surprisingly simple 3d gaussian splats from sparse unposed images. *arXiv preprint arXiv:2410.24207*, 2024.

[56] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, et al. gsplat: An open-source library for gaussian splatting. *Journal of Machine Learning Research*, 26(34):1–17, 2025.

[57] 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.

[58] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19447–19456, 2024.

[59] Hongjia Zhai, Xiyu Zhang, Boming Zhao, Hai Li, Yijia He, Zhaopeng Cui, Hujun Bao, and Guofeng Zhang. Splatloc: 3d gaussian splatting-based visual localization for augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 2025.

[60] Shuangfei Zhai. Exclusive self attention. *arXiv preprint arXiv:2603.09078*, 2026.

[61] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

[62] 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.

[63] Tianyuan Zhang, Zhengfei Kuang, Haian Jin, Zexiang Xu, Sai Bi, Hao Tan, He Zhang, Yiwei Hu, Milos Hasan, William T Freeman, et al. Relitlrm: Generative relightable radiance for large reconstruction models. *arXiv preprint arXiv:2410.06231*, 2024.

[64] Chen Ziwen, Hao Tan, Kai Zhang, Sai Bi, Fujun Luan, Yicong Hong, Li Fuxin, and Zexiang Xu. Long-lrm: Long-sequence large reconstruction model for wide-coverage gaussian splats. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4349–4359, 2025.

A Related Works

Feed-Forward 3D Gaussian Splatting. 3D Gaussian Splatting (3DGS) [27] has emerged as an efficient explicit scene representation for novel view synthesis, modeling a scene as a set of anisotropic 3D Gaussian primitives rendered with a differentiable rasterizer. While the original 3DGS formulation relies on per-scene optimization, recent feed-forward methods aim to predict Gaussian primitives directly from sparse input images, enabling generalizable reconstruction without test-time optimization. Early approaches incorporate epipolar attention [8] and plane-sweeping cost volumes [9, 52] to achieve 3D consistency in few-shot novel view synthesis. Furthermore, pose-free methods exploit depth and correspondence priors [17] or self-supervised ego-motion estimation [25] to extend these frameworks to settings with unknown camera poses. More recently, several methods leverage the power of large transformer architectures to reformulate feed-forward 3D reconstruction as a sequence-to-sequence image-to-3DGS translation, including GS-LRM [62] for posed settings and subsequent pose-free extensions [55, 21, 20].

However, the quadratic complexity of transformer self-attention with respect to the number of input views and image resolution poses a significant scalability bottleneck. To address this, subsequent works have explored linear-complexity state-space architectures [64], efficient attention designs [23], and test-time training strategies [47]. Among these, MVP [24] introduces a hierarchical attention mechanism along both the view and resolution axes, enabling scalable, high-quality 3DGS reconstruction. In this work, we build upon MVP and conduct a systematic investigation of its architectural properties, aiming to further improve the quality of feed-forward 3DGS reconstruction.

Hierarchical Dense Prediction Backbones. Dense prediction tasks aim to produce high-quality, per-pixel outputs from input images, which requires capturing both fine spatial detail and long-range context. U-Net [39] has become a de facto backbone across a wide range of dense prediction problems, owing to its encoder–decoder design with skip connections. More recently, DPT [38] adapted this design to vision transformers by reassembling tokens from multiple transformer layers into image-like multi-resolution features and progressively fusing them through residual convolutional blocks, and this hierarchical fusion strategy has since been widely adopted in feed-forward 3D reconstruction methods [49, 48, 30]. In parallel, several works have integrated transformers directly into the U-Net design itself, either at the bottleneck or as a replacement for its convolutional blocks [37, 13, 7]. Inspired by these hierarchical architectures, we explore a U-shaped multi-view transformer backbone with skip connections for feed-forward 3D Gaussian reconstruction, combining low-resolution cross-view reasoning with high-resolution pixel-aligned Gaussian prediction.

B Limitations

Despite its strong performance, U-MVP has several limitations. First, as a pixel-aligned model, the number of 3D Gaussians is directly tied to the input image resolution and the number of input views, which leads to redundant primitives in textureless or low-complexity regions and limits the model's ability to allocate representational capacity adaptively. Second, while U-MVP demonstrates strong performance within its training distribution, it falls short of the generality expected of a real-world foundation model for geometric reconstruction. The model is trained exclusively on the DL3DV dataset, in contrast to recent feed-forward geometric foundation models that leverage over ten diverse datasets, and further assumes a static scene, limiting its applicability to the dynamic and unconstrained conditions encountered in practice. Addressing these gaps through large-scale multi-dataset training and relaxing the static-scene assumption remain important directions for future work. In addition, U-MVP relies on externally estimated poses in the pose-free setting and is not a unified model that handles posed and unposed inputs within a single network.

C Additional Details

C.1 Training

We train our model following the MVP framework with slight modifications, as described below:

- **First stage**: Following prior work, we train at a resolution of 480×256 for 100k iterations for 4 days using a learning rate of 2×10^{-4}. The frame interval is sampled uniformly in the range $[64, 128]$.
- **Second stage**: Following prior work, we increase the resolution to 960×540 and train for 65k iterations for 2 days with a learning rate of 2×10^{-5}. Input views are sampled from the full frame range, and we apply the intrinsic augmentation strategy proposed in [64].
- **Remarks**: For training efficiency, we employ only view-swapping group permutation layers in the bottleneck until stage 2. In stage 3, two of these layers are replaced with global attention layers. Since transformer layers are independent of the number of input images, this transition introduces no additional parameters.
- **Third stage**: Maintaining the resolution at 960×540, we continue training for 60k iterations for about 2.5 days with the learning rate decayed from 2×10^{-5} to 1×10^{-6}. To improve robustness across diverse view configurations, we vary the number of input views in the range $[16, 192]$. Batch size and the number of target views are adjusted based on available GPU memory. Input views are drawn from the full frame range, and the intrinsic augmentation strategy is retained throughout.

# Input views	16	32	64	128	192
Batch size (per GPU)	10	6	3	1	1
# Target views	6	6	6	2	1
Ratio (%)	6.25	12.5	18.75	31.25	31.25

Table 10: Training settings with different numbers of input views.

Furthermore, we adopt a cosine learning rate scheduler and the AdamW optimizer [32] with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and a weight decay of 0.05, which is not applied to normalization or bias parameters. We also employ an exponential moving average [14] of model weights. Notably, to mitigate training instability observed in the first stage, we apply gradient clipping exclusively during this stage.

C.2 Evaluation

Importantly, target view indices are fixed across all input-view configurations for each scene. Target frames are obtained by sampling every 8th frame from the full sequence, with input views sampled from the remaining frames. For input view sampling, we follow prior work [64] and employ a combination of K-means clustering based on camera poses and uniform sampling based on the number of remaining frames. Thus, as the number of input views increases, its effect on novel view synthesis quality at the target views is directly observable, with reconstruction density improving accordingly.

C.3 Implementation Details

All linear and normalization layers in the transformer blocks omit bias terms, and model weights are initialized from a zero-mean normal distribution with a standard deviation of 0.02. Camera poses are normalized to establish a consistent coordinate system and scale. A reference frame is derived from the mean camera position and viewing directions computed across all input extrinsics, and all poses are transformed accordingly. The scene is then uniformly scaled such that the maximum camera distance equals one [62]. We also provide detailed architectural hyperparameters in Tab. 11.

D Additional Results

D.1 Runtime analysis

Fig. 8 and Fig. 9 show how inference time scales with the number of input views. We conduct all measurements at 960×540 resolution on a single NVIDIA B200 GPU using each method's official implementation. Although tttLRM [47] supports multi-GPU inference, we run all methods on a single GPU for a fair comparison. In eager mode (Fig. 8), U-MVP and MVP [24] scale moderately with the number of views and remain below one second up to 192 views, while the latency of iLRM [23] and

Category	Hyperparameter	Value
Token dimensions	$[d_1, d_2, d_3, d_4, d_5]$	[256, 512, 1024, 768, 512]
# Encoder layers	$[L_1^{\text{enc}}, L_2^{\text{enc}}, L_3^{\text{enc}}]$	[1, 2, 4]
# Decoder layers	$[L_3^{\text{dec}}, L_2^{\text{dec}}, L_1^{\text{dec}}]$	[5 (3 group + 2 global), 2, 1]
Attention	[Head dim., M_1, M_2]	[64, 16, 4]
Tokenization	[Patch size p, Register tokens]	[8, 4]
Gaussians	[Color SH deg., Opacity SH deg.]	[1, 2]

Table 11: Architectural hyperparameters of U-MVP.

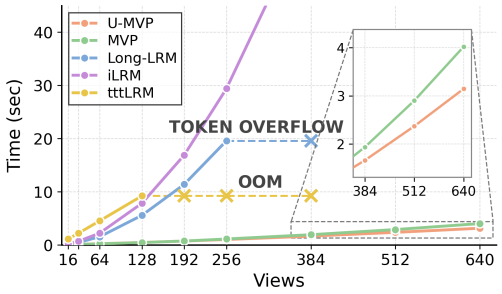

Figure 8: **Inference time** of all methods in eager mode at 960×540 on a single B200 GPU. `OOM` denotes out-of-memory on a single GPU and `TOKEN OVERFLOW` denotes exceeding the maximum token length. Ours becomes faster than MVP beyond 192 views, reaching $1.28\times$ at 640 views.

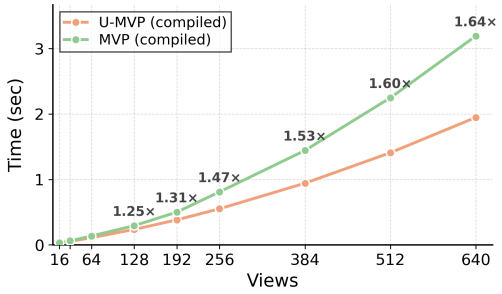

Figure 9: **Inference time with** `torch.compile` for MVP and ours under the same setting. Ours is faster at every view count, and the speedup grows from $1.13\times$ at 16 views to $1.64\times$ at 640 views.

Long-LRM [64] grows rapidly. iLRM approaches 30 seconds at 256 views and exceeds 60 seconds at 384 views, and Long-LRM is constrained by token overflow beyond 256 views. tttLRM also runs out of memory on a single GPU when the number of views exceeds 128. In contrast, our method scales gracefully to 640 views without any memory or token-length issues. U-MVP becomes faster than MVP from 256 views onward, and the gap widens to $1.28\times$ at 640 views (zoomed inset). Since the other baselines are already several times slower than both in eager mode, we further compare U-MVP and MVP with `torch.compile` (Fig. 9). In this setting, U-MVP is faster at every view count, with the speedup increasing from $1.13\times$ at 16 views to $1.64\times$ at 640 views, demonstrating the efficiency benefits of our architectural design.

D.2 Quantitative results

We evaluate both U-MVP and MVP at input view counts never seen during training. Following the evaluation protocol in Sec. C.2, we sample every 8th frame of each scene as a target view, remove these frames from the frame list, and use all remaining frames as input. Since each scene contains a different number of frames, this yields between 272 and 368 input views per scene, roughly $1.4\times$ to $1.9\times$ beyond the largest count used in training. Because group-wise attention requires the number of input views to be a multiple of 16 for U-MVP and 4 for MVP, we duplicate the final frames as needed. Other baselines are excluded, as they either degrade in the dense-view setting or fail to produce meaningful renderings at these view counts, as already observed in Tab. 2.

As shown in Tab. 12, neither model degrades outside the training range. Both improve over their 256-view results, indicating that reconstruction quality keeps increasing with additional views rather than deteriorating once the trained maximum is exceeded. Moreover, the margin of U-MVP over MVP widens from $+0.79$ dB at 256 views to $+0.88$ dB in this setting, consistent with the trend in Tab. 1 and 2 that our gains grow with the number of input views.

Method	256 views			All views (272–368)		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
MVP [24]	29.67	0.915	0.125	29.80	0.918	0.123
Ours	**30.46**	**0.925**	**0.117**	**30.68**	**0.928**	**0.114**

Table 12: **Generalization beyond the trained number of input views on DL3DV.** "All views" uses every frame except the target views as input, resulting in 272 to 368 input views per scene. The 256-view results are taken from Tab. 2.

Bottleneck	PSNR↑	SSIM↑	LPIPS↓
Global	**26.20**	**0.844**	**0.151**
Group swap	26.05	0.841	0.155
Hybrid (Ours)	26.14	0.842	0.153

Table 13: **Novel view synthesis** with different bottleneck designs on DL3DV (256×256, 16 views).

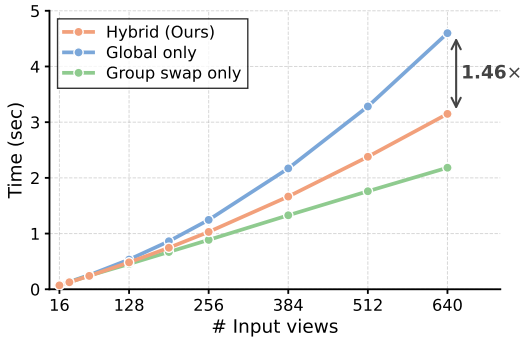

Figure 10: **Inference latency** of the bottleneck designs at 960×540 on a single RTX PRO 6000 GPU. The gap between global-only attention and our hybrid reaches $1.46\times$ at 640 views.

D.3 Trade-off of the bottleneck design

We compare three variants of the bottleneck block $\mathcal{D}_3$ under an identical backbone: global attention only, grouping and swapping only, and our hybrid of both, in posed 3DGS reconstruction. Quality is evaluated on DL3DV at 256×256 with 16 input views, using a larger training batch size than the ablations in Tab. 9. Latency is measured at 960×540 on a single RTX PRO 6000 GPU, averaged over 5 iterations after 3 warmup iterations.

As shown in Tab. 13, global-only attention outperforms our hybrid by 0.06 dB PSNR, 0.002 SSIM, and 0.002 LPIPS, while its latency gap over the hybrid widens with the number of input views, from $1.06\times$ at 16 views to $1.46\times$ at 640 views. Removing global attention entirely is faster still, but reduces PSNR by 0.15 dB, more than twice the gap between global-only attention and our hybrid. Retaining two global attention layers therefore recovers most of the quality of full global attention at a fraction of its scaling cost.

D.4 Feature visualization

To examine what the Gaussian prediction head receives, we visualize the final-layer features of MVP [24] and U-MVP, i.e., the tokens passed to the per-pixel linear Gaussian head. For MVP, these are the outputs of the PFA module; for U-MVP, they are the outputs of the finest-level decoder $\mathcal{D}_1$. For each scene and model, we fit a PCA jointly over the features of all 16 input views and map the first three components to RGB.

As shown in Fig. 11, both models produce spatially coherent features that are consistent across views. MVP's features are spatially smooth, with diffuse object boundaries. U-MVP's features preserve sharper boundaries and finer structures, which is consistent with the skip connections propagating high-frequency per-view detail from the frame-wise encoder to the decoder.

D.5 Qualitative results

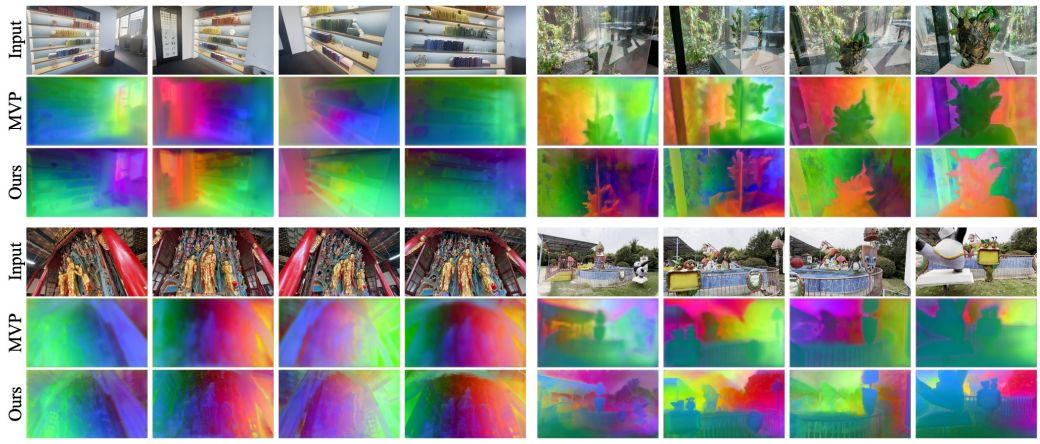

Figure 11: **PCA visualization of final-layer features** (DL3DV, 16 input views). Each block shows one scene, with input views, MVP, and U-MVP (Ours) from top to bottom. PCA colors are not comparable between models.

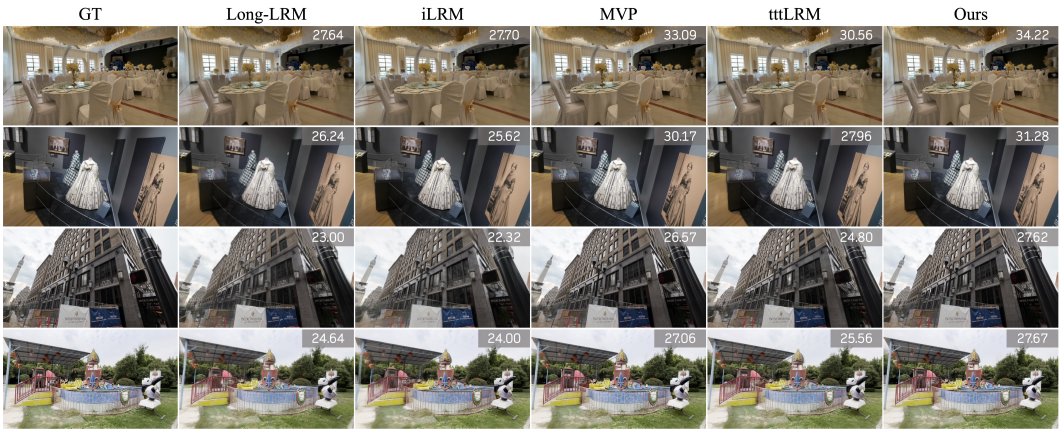

Figure 12: Qualitative results on the DL3DV dataset. All methods are evaluated using 32 input views.



Figure 13: Qualitative results on the Tanks&Temples (top two rows) and Mip-NeRF360 (bottom two rows) datasets. All methods are evaluated using 32 input views.

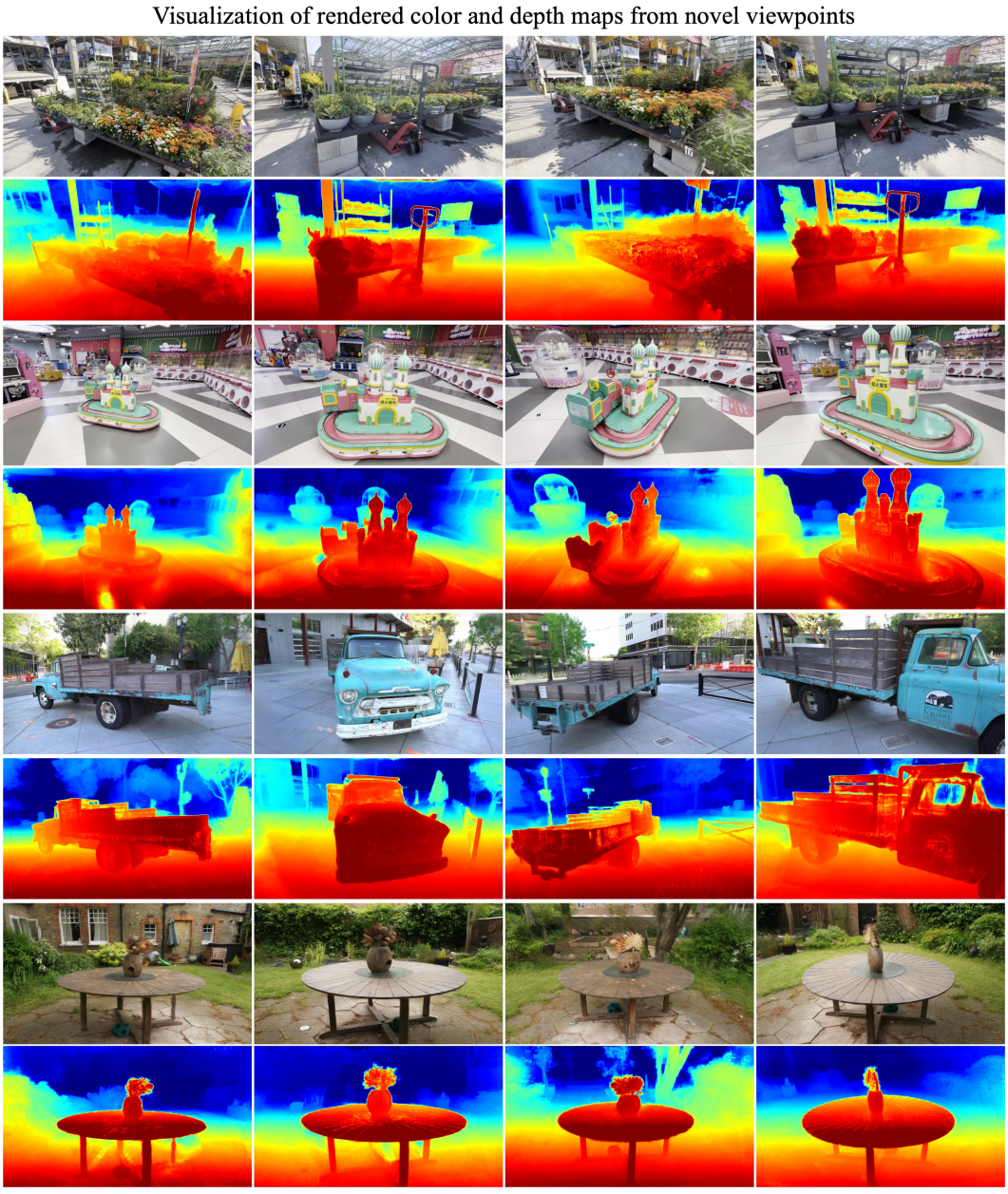

Figure 14: Visualization of rendered color and depth maps from novel viewpoints using 32 input views of the DL3DV dataset.

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 claims made in the abstract and introduction are supported by the experimental results presented in the paper. All architectural contributions and performance improvements are validated through quantitative comparisons and ablation studies.

   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: We discuss the limitations of our work in Appendix B.

   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: This paper does not include theoretical results or proofs. The contributions are empirical and architectural.

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: We describe the architecture clearly and fully, with visual aids. Our training code is based on the open-sourced project, and we described the modifications we made.

   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: [Yes]

Justification: Code, pretrained checkpoints, and instructions to reproduce the main results will be released through our project page at `https://yang-gwon2.github.io/UMVP/`. All datasets we use (DL3DV, Tanks&Temples, Mip-NeRF360) are publicly available.

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: We provide relevant details in Sec. 3.1, App. C, and the code.

   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 did not report error bars for the experiments.

   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: We provide relevant details in Sec. 3.1 and App. C.

   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: Our 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: [N/A]

    Justification: This work proposes a neural architecture for 3D scene reconstruction from multi-view images. We do not foresee any direct positive or negative societal impacts from this work.

    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: This work presents a 3D scene reconstruction model trained on publicly available datasets. We do not anticipate high risk for misuse, and therefore no additional safeguards beyond standard research practices are required.

    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: All datasets, code, and models used in this work are properly credited with citations. We use publicly available datasets and open-source libraries in accordance with their respective licenses and terms of use.

    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: [Yes]

    Justification: We will release our code and pretrained models through our project page at `https://yang-gwon2.github.io/UMVP/`, with documentation covering installation, training, evaluation, and the license. No new datasets are introduced.

    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: We do not involve crowdsourcing or 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: We do not involve crowdsourcing or 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: LLMs were not used as part of the core methodology of this work.

    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.