This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Generative Densification: Learning to Densify Gaussians for High-Fidelity Generalizable 3D Reconstruction

Seungtae Nam[1*] Xiangyu Sun[2*] Gyeongjin Kang[2] Younggeun Lee[2] Seungjun Oh[2] Eunbyung Park[1†]
[1]Yonsei University [2]Sungkyunkwan University

https://stnamjef.github.io/GenerativeDensification/

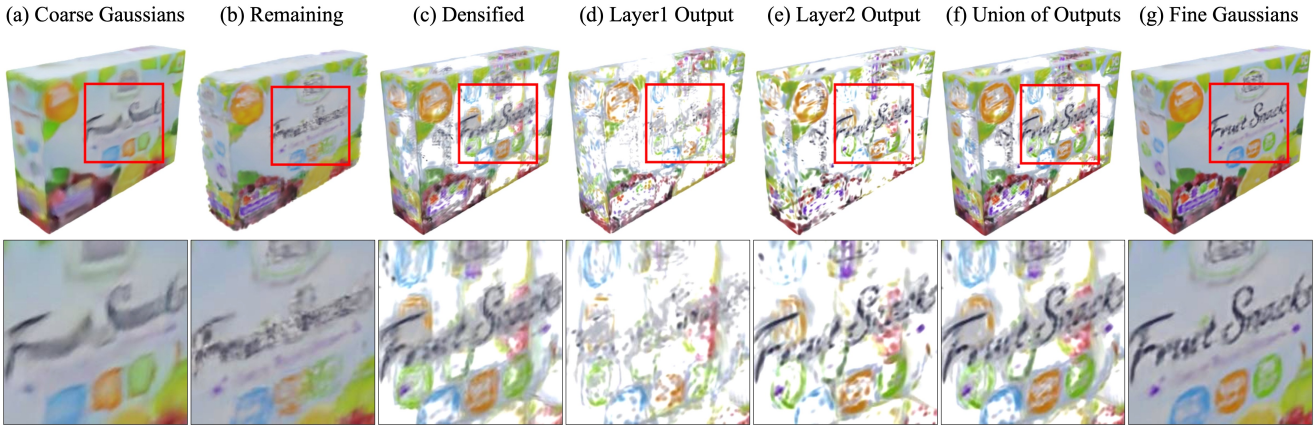


Figure 1. Our method selectively densifies (a) coarse Gaussians from generalized feed-forward models. (c) The top K Gaussians with large view-space positional gradients are selected, and (d-e) their fine Gaussians are generated in each densification layer. (g) The final Gaussians are obtained by combining (b) the remaining (non-selected) Gaussians with (f) the union of each layer's output Gaussians.

Abstract

Generalized feed-forward Gaussian models have made significant strides in sparse-view 3D reconstruction by leveraging prior knowledge from large multi-view datasets. However, these models often struggle to represent high-frequency details primarily due to the limited number of Gaussians. While the densification strategy from per-scene 3D Gaussian splatting can be adapted for feed-forward models, it typically requires tens of thousands of optimization steps to reconstruct fine details and can easily lead to overfitting in sparse-view scenarios. In this paper, we propose Generative Densification, an efficient and generalizable densification strategy specifically tailored for feed-forward models. Instead of iteratively splitting and cloning raw Gaussian parameters, our method up-samples feature representations produced by feed-forward models and generates their corresponding fine Gaussians in a single forward pass, leveraging learned prior knowledge for enhanced generalization. Experimental results on both object-level and scene-level reconstruction tasks demonstrate that our method outperforms state-of-the-art approaches with comparable or smaller model sizes, achieving notable improvements in representing fine details.

1. Introduction

3D Gaussian splatting (3D-GS) [18] has been a massive success in high-quality 3D scene reconstruction and real-time novel view synthesis, representing a 3D scene with a set of learnable Gaussian primitives and exploiting a fast differentiable rasterization pipeline. Since its introduction, numerous studies have explored 3D-GS as a versatile 3D representation with various applications extending beyond the per-scene optimization. One notable example is generalized feed-forward Gaussian models [3, 7, 49], which generate Gaussian primitives to reconstruct 3D scenes or objects in a single forward pass. By leveraging prior knowledge learned from large multi-view datasets (e.g., Objaverse [9, 10], RE10k [50]), these models can reconstruct 3D scenes or objects from just a single or a few images.

*Equal contribution
†Corresponding author

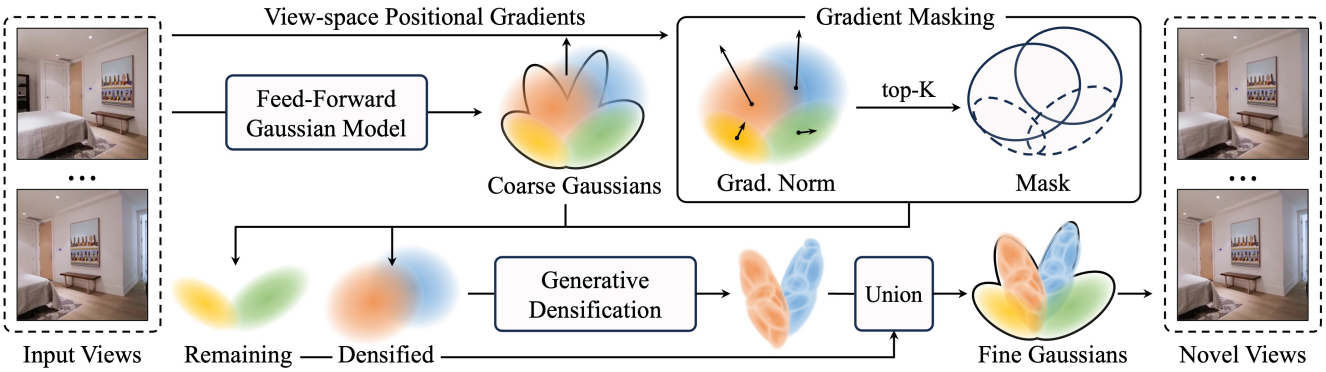


Figure 2. Generative Densification overview. Our method selectively densifies Gaussians with large view-space positional gradients.

While effective and promising, the feed-forward Gaussian models often face challenges in capturing high-frequency details primarily due to the limited number of Gaussians and the lack of a densification strategy tailored for these approaches. One possible solution is to generate more Gaussians, but this does not fully address the problem. For example, in the pixel-aligned Gaussian model [3], a widely adopted architecture in the feed-forward models (see Sec. 2 for details), we can predict multiple Gaussians per pixel. However, this strategy uniformly increases the number of Gaussians across the entire 3D space, resulting in an excess of unnecessary small Gaussians that may degrade rendering quality and speed. Ideally, the scene details should be represented by numerous small Gaussians, while smoother regions can be covered by a few large Gaussians.

In per-scene 3D-GS optimization, adaptive densification strategy selectively increases the number of Gaussians to better reconstruct complex 3D geometry and fine details. Specifically, every few optimization steps, it densifies only the Gaussians with large view-space positional gradients, replicating the existing Gaussians or splitting them into smaller ones when more Gaussians are required. Through repeated optimization and densification processes, the Gaussians gradually become non-uniformly distributed, with numerous small Gaussians concentrated in detailed areas and a few large Gaussians scattered across smooth regions. This simple heuristic algorithm has demonstrated its effectiveness across various real-world scenes and objects.

A straightforward approach to extend the densification strategy to the feed-forward Gaussian models is to fine-tune the output Gaussians from the feed-forward models using the original 3D-GS optimization and densification procedure. However, directly applying the densification strategy used in per-scene optimization scenarios presents several challenges. For example, it demands tens and thousands of optimization steps to converge and reconstruct the fine details, which significantly reduces generation speed. Additionally, the per-scene 3D-GS optimization usually assumes many input images from various viewpoints, whereas the feed-forward models typically take just a few views or even a single view image. Thus, the densification and extensive optimization to a few view images can easily lead to overfitting, compromising the feed-forward models' ability to generalize and synthesize novel views effectively.

In this paper, we propose *Generative Densification (GD)*, an efficient generative densification strategy for generalized feed-forward Gaussian models, which is designed in accordance with the following principles: 1) adaptability, allowing the model to distinguish Gaussians that requires densification and those that do not; and 2) generalizability, enabling the model to learn and leverage prior knowledge from large multi-view datasets. Inspired by the 3D-GS densification strategy, our approach leverages view-space positional gradients to identify where additional Gaussians are needed. We selectively densify the top K Gaussians with large gradients, and the remaining (non-selected) Gaussians are used together with the densified ones to render high-fidelity images (Fig. 2). Unlike the 3D-GS densification strategy, the proposed *GD* densifies the selected Gaussians based on the learned prior knowledge for the purpose of generalizable 3D reconstruction. More specifically, *GD* densifies feature representations from feed-forward models instead of raw Gaussian parameters, embedding prior knowledge in these features for improved generalization.

We further propose to utilize an efficient point-level transformer [37] to implement *GD*. Due to the quadratic increase in memory and the unstructured nature of point-level 3D data, it is infeasible to apply naive self-attention on hundreds of thousands of input Gaussians. While local attention can be applied within groups of neighboring Gaussians, finding the neighbors by calculating and comparing distances between the Gaussians is computationally expensive. Instead, we sort the Gaussians in traversing order of space-filling curves [28] to rearrange them into non-overlapping groups and apply group-wise attention, enabling efficient operations directly in 3D space. Additionally, while we selectively densify the Gaussians with large view-space positional gradients, multiple rounds of densification are unnec-

essary for all of them. To optimize this process, we predict a confidence mask for each densification layer to filter out the Gaussians that do not require further densification.

We integrated *GD* into two recent feed-forward Gaussian models: LaRa [5] for object-level reconstruction and MVSplat [7] for scene-level reconstruction. On the large-scale Gobjaverse [38] and RE10k [50] datasets, two models incorporating our method achieved the best performance, significantly improving reconstruction quality compared to their respective baselines. Qualitative analysis further highlights that the fine Gaussians generated by *GD* effectively capture thin structures and intricate details, which are often challenging for the coarse Gaussians to represent accurately (Fig. 1). Additionally, cross-dataset evaluations confirm that *GD* consistently improves image quality, demonstrating its robust generalizability across different datasets.

2. Related Work

Feed-forward Gaussian Models. Feed-forward Gaussian models learn a mapping from a single or a set of few input images to a set Gaussian representations that can be rendered into any viewpoint. One common approach combines a pre-trained image encoder [2, 14, 27] with a learnable embedding decoder, where the decoder generates 3D features via a series of cross-attention between 3D embeddings and image features. These 3D features are commonly represented as voxels [5, 45], point-cloud [24], or a hybrid of point-cloud and triplane [52], and Gaussians are typically obtained by decoding the features using an MLP.

Another approach involves pixel-aligned Gaussian representations [3, 33], where each Gaussian is assumed to be on a ray of each pixel and distanced from the ray origin by a depth. The depth is either directly regressed from pixel-wise image features [33] or determined by the probability that a Gaussian exists at a depth along a ray [3]. Since its introduction, substantial efforts have been made to estimate the depth more accurately [7, 23, 32, 40, 46], scale up models for better generalizability [34, 41, 47], and generate 3D scenes leveraging learned prior [15, 21, 36]. Recent works further leverage these representations to cover wide-ranging views [6, 8, 35, 51] or reconstruct 4D scenes [20, 30].

While effective, both approaches often struggle to reconstruct fine details, mainly due to the limited number of Gaussians and the lack of a densification strategy specifically designed for feed-forward models. We aim to develop a unified densification framework that can be seamlessly applied to both of these representative approaches.

Adaptive Density Control of Gaussians. Adaptive densification [18] plays a crucial role in capturing fine details and controlling the number of Gaussians. However, it relies heavily on well-initialized Gaussians, and its core operations (splitting, cloning, and pruning) are not sufficiently effective, often leading to blurry reconstructions, overfitting, and slow training. A line of studies [17, 19] tackles the reliance on accurate initialization, achieving comparable or superior rendering quality even with random initialization. Another line of studies [1, 13, 26, 42, 48] explores alternative densification criteria beyond view-space positional gradients to better capture Gaussians in blurry regions. Recently, overfitted Gaussian pruning [11] and aggressive densification [25] have been introduced, with the former enhancing generalization and the latter accelerating training.

These methods typically require more than thousands of optimization and densification steps, and the discussions are limited to per-scene scenarios. Instead of directly applying them to generalized settings, we learn from large multi-view datasets to generate fine Gaussians in a single forward pass.

3. Method

3.1. Generative Densification

A feed-forward Gaussian model is a function Φ that maps a set of images $\mathcal{I} = \{I_v\}_{v=1}^{V}$ and camera poses $\mathcal{C} = \{C_v\}_{v=1}^{V}$ to a set of Gaussians $\mathcal{G} = \{(\mu_i, \alpha_i, \Sigma_i, c_i)\}_{i=1}^{N}$ and their associated features $\mathcal{F} = \{f_i\}_{i=1}^{N}$:

$$\mathcal{G}, \mathcal{F} = \Phi(\mathcal{I}, \mathcal{C}), \tag{1}$$

where V is the number of input views, and N is the number of Gaussians. Each feature f_i is the penultimate feature extracted from the feed-forward model (see Sec. 3.3), which can be decoded into Gaussian parameters, including center position μ_i, opacity α_i, covariance Σ_i, and spherical harmonics (SH) coefficient c_i.

Our goal is to learn a model Ψ that adaptively densifies Gaussians generated by a variety of feed-forward models:

$$\tilde{\mathcal{G}} = \Psi(\mathcal{G}, \mathcal{F}, \mathcal{I}, \mathcal{C}). \tag{2}$$

First, we calculate gradient scores to split the Gaussians into two groups, $\mathcal{G}_{\text{den}}$ and $\mathcal{G}_{\text{rem}}$, the Gaussians requires densification and the remaining Gaussians, respectively. More specifically, we compute the scores as the norms of the gradients with respect to view-space coordinates of Gaussian positions and average the norms across the V input views:

$$s_i = \frac{1}{V} \sum_{v=1}^{V} \| \nabla_{p(\mu_i, v)} \mathcal{L}_{\text{MSE}}(I_v, \hat{I}_v) \|_2, \tag{3}$$

where $\mathcal{L}_{\text{MSE}}(I_v, \hat{I}_v)$ is a mean squared error loss between ground truth images (I_v) and rendered images ($\hat{I}_v$), and $p(\mu_i, v) \in \mathbb{R}^2$ denotes the projected coordinate of i-th Gaussian position onto input view v. Then, we select the top K scoring Gaussians ($\mathcal{G}_{\text{den}}$), and their positions and features ($\mathcal{X}_{\text{den}} \in \mathbb{R}^{K \times 3}$, $\mathcal{F}_{\text{den}} \in \mathbb{R}^{K \times D}$) are passed to the densification module, which consists of L layers of serialized attention (`SA`), up-sampling (`UP`), splitting via learnable masking

(SPLIT), and Gaussian head (HEAD) blocks (Fig. 3-(a)):

$$(\tilde{\mathcal{X}}^{(l)}, \tilde{\mathcal{F}}^{(l)}) = \texttt{SA}(\mathcal{X}_{\text{den}}^{(l-1)}, \mathcal{F}_{\text{den}}^{(l-1)}), \quad (4)$$

$$(\mathcal{X}^{(l)}, \mathcal{F}^{(l)}) = \texttt{UP}(\tilde{\mathcal{X}}^{(l)}, \tilde{\mathcal{F}}^{(l)}), \quad (5)$$

$$(\mathcal{X}_{\text{den}}^{(l)}, \mathcal{F}_{\text{den}}^{(l)}, \mathcal{X}_{\text{rem}}^{(l)}, \mathcal{F}_{\text{rem}}^{(l)}) = \texttt{SPLIT}(\mathcal{X}^{(l)}, \mathcal{F}^{(l)}), \quad (6)$$

$$\mathcal{G}^{(l)} = \texttt{HEAD}(\mathcal{X}_{\text{rem}}^{(l)}, \mathcal{F}_{\text{rem}}^{(l)}), \quad (7)$$

for $l \in \{1, ..., L-1\}$, where $\mathcal{X}_{\text{den}}^{(0)} = \mathcal{X}_{\text{den}}$ and $\mathcal{F}_{\text{den}}^{(0)} = \mathcal{F}_{\text{den}}$. Here, we initialize the input positions as the Gaussian center positions. $\texttt{SA}(\cdot,\cdot)\colon \mathbb{R}^{K\times 3} \times \mathbb{R}^{K\times D} \rightarrow \mathbb{R}^{K\times 3} \times \mathbb{R}^{K\times D}$, efficiently aggregates scene information through serialized attention. $\texttt{UP}(\cdot,\cdot)\colon \mathbb{R}^{K\times 3} \times \mathbb{R}^{K\times D} \rightarrow \mathbb{R}^{\tilde{K}\times 3} \times \mathbb{R}^{\tilde{K}\times D}$ ($\tilde{K} >> K$), up-samples the Gaussian positions and features. $\texttt{SPLIT}(\cdot,\cdot)\colon \mathbb{R}^{\tilde{K}\times 3} \times \mathbb{R}^{\tilde{K}\times D} \rightarrow \mathbb{R}^{\tilde{K}_{\text{den}}\times 3} \times \mathbb{R}^{\tilde{K}_{\text{den}}\times D} \times \mathbb{R}^{\tilde{K}_{\text{rem}}\times 3} \times \mathbb{R}^{\tilde{K}_{\text{rem}}\times D}$ ($\tilde{K} = \tilde{K}_{\text{den}} + \tilde{K}_{\text{rem}}$), divides the up-sampled positions and features into those requiring further densification in the next layer and the remaining ones (see Appendix C for further details). By using the remaining ones, we generate each layer's Gaussians $\mathcal{G}^{(l)}$ using $\texttt{HEAD}(\cdot,\cdot)$ block. For the L-th layer, we generate the Gaussians by $\mathcal{G}^{(L)} = \texttt{HEAD}(\texttt{UP}(\texttt{SA}(\mathcal{X}_{\text{den}}^{(L-1)}, \mathcal{F}_{\text{den}}^{(L-1)})))$, and the final set of fine Gaussians is obtained as follows:

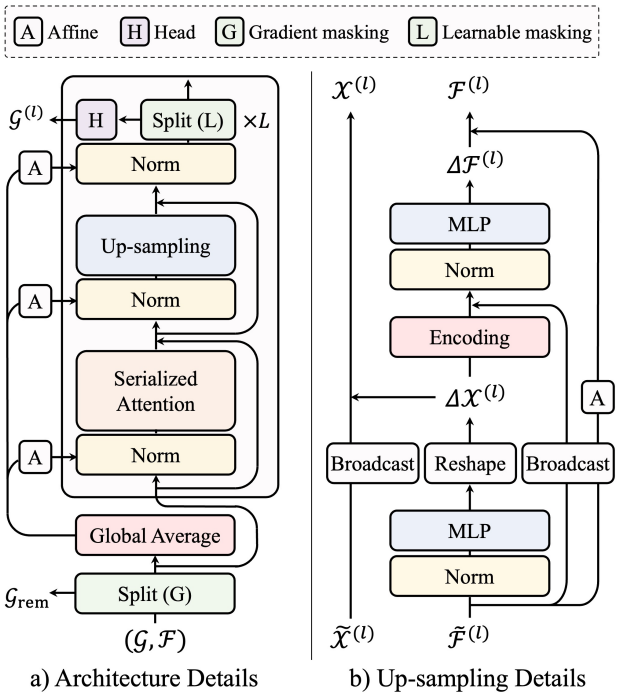
$$\tilde{\mathcal{G}} = \mathcal{G}_{\text{rem}} \cup \{\bigcup_{l=1}^{L} \mathcal{G}^{(l)}\}. \quad (8)$$

3.2. Architecture Details

Serialized Attention. Due to the quadratic increase in memory requirements and the unstructured nature of the Gaussian representations, it is infeasible to apply self-attention directly to all input Gaussians. One alternative is to group the Gaussians by searching for their neighbors and apply group-wise attention. While this approach allows the model to consider the relative positions of the neighbors, searching for neighbors across hundreds of thousands of Gaussians is memory and computationally expensive.

Serialized attention [37] enables efficient operations on unstructured point clouds by sorting them in traversing order of given space-filling curves [28]. Following its design principles, we first encode each Gaussian position into a serialized code, an integer value that reflects its order in the space-filling curves [37]. Then, the Gaussian features are structured by sorting the serialized codes and divided into non-overlapping groups, where self-attention is applied within the same group. By adapting serialized attention, our method efficiently embeds prior knowledge and local scene context into the features, allowing the output features to be used as sources for generating fine Gaussians.

Up-sampling. In the UP block, we predict residuals for each Gaussian position and feature (Fig. 3-(b)). We generate R offsets for each Gaussian position by passing the input



a) Architecture Details b) Up-sampling Details

Figure 3. (a) Given Gaussians and features, out method selects K Gaussians and densifies their positions and features through densification module. (b) Predicted positional and feature residuals are added to their respective inputs after a broadcasting operation.

features through an MLP parameterized by $\theta_x^{(l)}$. These predicted offsets are then positionally encoded, concatenated with the input features, and transformed to residual features using another MLP parameterized by $\theta_f^{(l)}$. More formally,

$$\Delta x_i^{(l)} = \texttt{MLP}(\tilde{f}_i^{(l)}; \theta_x^{(l)}), \quad (9)$$

$$\Delta f_{i,j}^{(l)} = \texttt{MLP}(\gamma(\Delta x_{i,j}^{(l)}) \oplus \tilde{f}_i^{(l)}; \theta_f^{(l)}), \quad (10)$$

where $\tilde{f}_i^{(l)} \in \mathbb{R}^D$ is the feature of i-th Gaussian after the l-th serialized attention block. $\Delta x_i^{(l)} \in \mathbb{R}^{R\times 3}$ is the predicted R offsets per Gaussian, and $\Delta x_{i,j}^{(l)} \in \mathbb{R}^3$ denotes each predicted offset. Similarly, $\Delta f_i^{(l)} \in \mathbb{R}^{R\times D}$ is the predicted R residuals per Gaussian feature, and $\Delta f_{i,j}^{(l)} \in \mathbb{R}^D$ denotes each predicted residual. $\gamma(\cdot)$ is a positional encoding function, and $\oplus$ represents a concatenation operator. The output positions and features are obtained by adding the residuals to their respective inputs, following the broadcasting of the input positions and features to align the dimensions.

Splitting via Learnable Masking. In the SPLIT block, we use learnable masking to filter out Gaussians not requiring further densification in the next layer. While gradient masking initially selects Gaussians ($\mathcal{G}_{\text{den}}$) for densification, computing gradients in every layer introduces considerable computational overhead. Instead, we predict a confidence score $s_i^{(l)} \in [0, 1]$ for each Gaussian using an MLP parame-

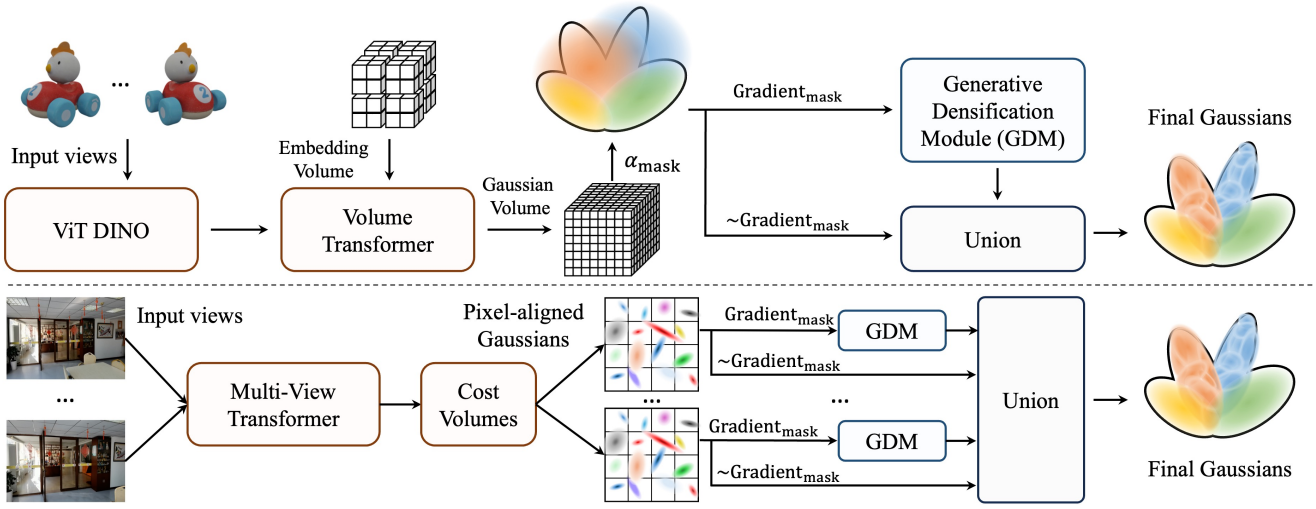


Figure 4. Overview of the Generative Densification pipelines for object-level (top) and scene-level (bottom) reconstruction tasks.

terized by $\theta_s^{(l)}$ and a sigmoid function $\sigma(\cdot)$. More formally,

$$s_i^{(l)} = \sigma(\texttt{MLP}(f_i^{(l)}; \theta_s^{(l)})), \tag{11}$$

where $f_i^{(l)} \in \mathbb{R}^D$ is the feature for the i-th Gaussian. Based on these predicted scores, the top $\tilde{K}_{\text{den}}$ scoring Gaussians are selected for further densification, while the remaining $\tilde{K}_{\text{rem}}$ Gaussians are decoded into Gaussian parameters.

Since the top-k operation is non-differentiable, we attach the computational graph of the score $s_i^{(l)}$ to the input features $f_i^{(l)}$ while keeping the feature values intact:

$$f_i^{(l)} = \texttt{SG}(f_i^{(l)} - f_i^{(l)} s_i^{(l)}) + f_i^{(l)} s_i^{(l)}, \tag{12}$$

where $\texttt{SG}(\cdot)$ is a stop-gradient operator. Analogous to the straight-through estimator [43], the gradients with respect to the output features $\partial\mathcal{L}/\partial f_i^{(l)}$ propagate back to the second term of the Eq. (12), hence, which enables us to learn the confidence scoring function and do end-to-end training.

Gaussian Head. In the Gaussian `HEAD` block, the positions and features of the remaining Gaussians $(\mathcal{X}_{\text{rem}}^{(l)}, \mathcal{F}_{\text{rem}}^{(l)})$, filtered out by the learnable masking block, are decoded into a set of Gaussian parameters $(\mathcal{G}^{(l)})$. Their positions are set to match the input positions $(\mathcal{X}_{\text{rem}}^{(l)})$, while the remaining attributes (opacities, covariances, SH coefficients) are generated from the input features $(\mathcal{F}_{\text{rem}}^{(l)})$ using an MLP.

3.3. Applying Generative Densification

We present two models incorporating generative densification, based on LaRa [5] and MVSplat [7]. For object-level reconstruction, we use the Gaussians and the volume features generated by the LaRa as input, from which we generate fine Gaussians (top row of Fig. 4). For scene-level reconstruction, fine Gaussians are generated per view by utilizing the pixel-aligned Gaussians and image features extracted from the MVSplat (bottom row of Fig. 4). Additionally, residual learning is incorporated into the scene-level model to better reconstruct complex real-world scenes. Further details on the residual learning and the pipelines are provided in Appendix B and Appendix C, respectively.

4. Experiment

4.1. Experimental Setup

Implementation Details. We jointly trained the generative densification module with the LaRa backbone for 30 epochs, and then fine-tuned for 20 epochs to achieve further improvements in rendering quality. The training was conducted on four A6000-48G GPUs over 3.5 days, with a batch size of 4 per GPU. Similarly, the MVSplat backbone was trained jointly with our densification module for 300,000 iterations, followed by fine-tuning for 150,000 iterations. The scene-level training was conducted on four H100-80G GPUs over 6.5 days, also with a batch size of 4 per GPU. For a detailed description of the network and training hyperparameters, please refer to Appendix D.

Datasets. We train and evaluate our object-level model on Gobjaverse [38] dataset, a large-scale multi-view dataset with an image resolution of 512×512. To further demonstrate the cross-domain capability, we evaluate our method on Google Scanned Objects [12] dataset and a subset of Co3D [29] test set, following LaRa [5]. Similarly, we train and evaluate our scene-level model on RE10k [50] with an image resolution of 256×256, and further evaluate it on two cross-domain datasets, ACID [22] and DTU [16], following MVSplat [7]. For both models, we use training and testing splits provided in each dataset without any modifications.

Baselines. We compare our object-level model against the original LaRa [5], as well as LGM [34] and GS-LRM [47].

Table 1. Quantitative comparisons of our object-level models against their baselines. 'Our-fast' is trained on the Gobjaverse [38] training set for 30 epochs, and 'Ours' is further fine-tuned for 20 epochs. 'Ours (w/ residual)' is trained on the same set for 50 epochs with residual learning (Appendix B). LaRa is re-evaluated using the publicly available checkpoint and our view-sampling method (Appendix D).

Method	#Param. (M)	Gobjaverse [38]			GSO [12]			Co3D [29]		
		PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
MVSNeRF [4]	0.52	14.48	0.896	0.185	15.21	0.912	0.154	12.94	0.841	0.241
MuRF [39]	15.7	14.05	0.877	0.301	12.89	0.885	0.279	11.60	0.815	0.393
LGM [34]	415	19.67	0.867	0.157	23.67	0.917	0.063	13.81	0.739	0.414
GS-LRM [47]	300	-	-	-	30.52	0.952	**0.050**	-	-	-
LaRa [5]	125	27.49	0.938	0.093	29.70	0.959	0.060	21.18	0.862	0.216
Ours-fast	134	28.23	0.943	0.084	30.62	0.964	0.062	21.67	0.864	0.209
Ours	134	28.58	0.945	0.080	31.06	0.966	0.058	21.72	0.865	0.209
Ours (w/ residual)	134	**28.75**	**0.946**	**0.078**	**31.23**	**0.967**	0.058	**22.08**	**0.867**	**0.206**



Figure 5. Qualitative comparisons of our object-level model trained for 50 epochs against the original LaRa. The zoomed-in parts within the red boxes are shown on the right side of the second and third columns, focusing on the comparison of fine detail reconstruction. The two images in the rightmost column present the Gaussians input to and output from our generative densification module, respectively.

Additionally, we include two feed-forward NeRF models, MVSNeRF [4] and MuRF [39], for reference. For comparing our scene-level model, we include feed-forward NeRF models (pixelNeRF [48], GNPR [31], and MuRF [39]) and pixel-aligned Gaussian models (pixelSplat [3], DepthSplat [40], and the original MVSplat [7]) as baselines.

4.2. Results

Object-level Reconstruction. Fig. 5 compares the rendered images from the baseline LaRa against those from our object-level model. Our model clearly outperforms LaRa in reconstructing fine details, such as the white dots on the red ribbon (first row of Fig. 5) and the intricate floral patterns on the socks (second row of Fig. 5). It is also evident that our densification strategy effectively captures detailed areas and generate fine Gaussians, as shown in the rightmost column of Fig. 5. While LaRa quickly learns the object's general shape by leveraging explicit volume representations, it struggles to accurately represent edges or contours.

The quantitative comparison provided in Tab. 1 further demonstrates the ability of our model to capture and reconstruct fine details. Our model achieves the highest PSNR in both in-domain reconstruction and cross-dataset generalization tasks. It even outperforms GS-LRM, the current state-of-the-art model for object-level reconstruction, using significantly fewer number of parameters (300M vs. 134M). This highlights that selectively generating Gaussians in the detailed areas can be more effective than uniformly generating numerous Gaussians across the entire object.

Scene-level Reconstruction. Fig. 6 and Fig. 7 compare the rendered images from the baseline MVSplat against those from our scene-level model. Our model better reconstructs thin structures and fine details, such as the guardrail (first row of Fig. 6) and the wave (second row, first column of Fig. 7). Moreover, we empirically found that our method reduces the opacity of Gaussians in empty space, making them less visible in the rendered images. For example, it makes the floaters more transparent near the water pipe (second row of Fig. 6) and the iron fence (first row, first column of Fig. 7), removing artifacts in the images.

In the quantitative comparisons, our model not only outperforms the baselines including DepthSplat [40] on the in-

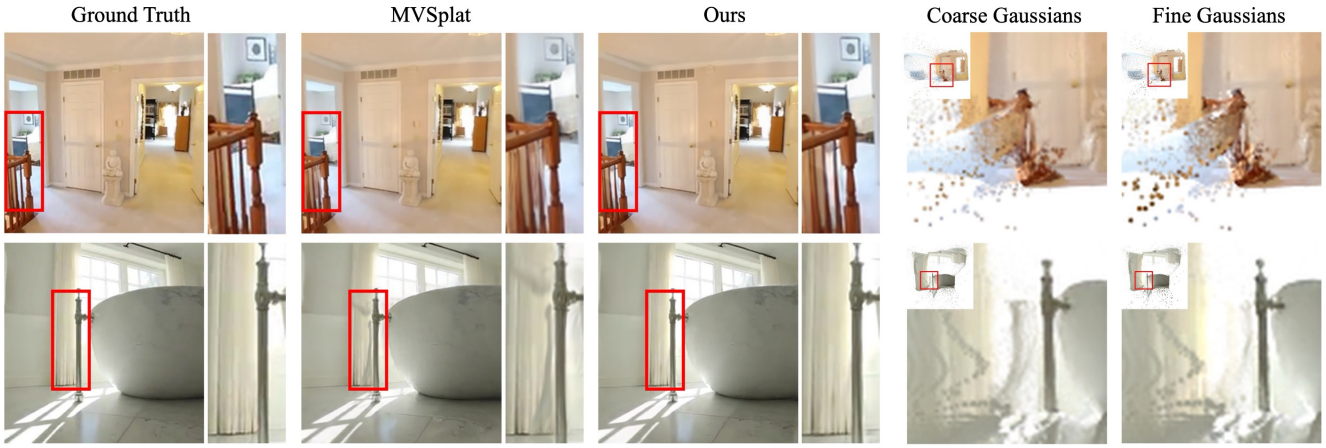


Figure 6. Qualitative comparisons of our scene-level model against the original MVSplat on the RE10k [50] dataset.

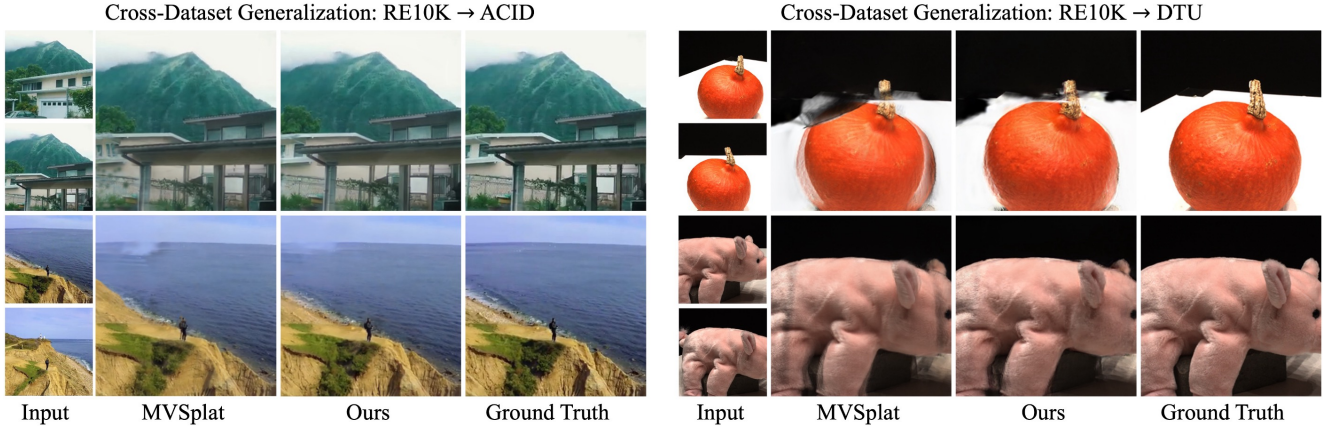


Figure 7. Qualitative comparisons of our scene-level model against the original MVSplat on the ACID [22] and DTU [16] datasets.

domain reconstruction task (Tab. 2), but also consistently achieves the best performance on the cross-dataset generalization task (Tab. 3). While DepthSplat is concurrent work, we include it as a baseline for the reader's reference. Our model outperforms DepthSplat with fewer parameters (37M vs. 28M) and a smaller batch size (32 vs. 16). Additionally, unlike DepthSplat, which relies on a pretrained depth predictor, our model is solely trained with multi-view images.

4.3. Analysis and Ablations

We begin by providing an intuitive example comparing coarse and fine Gaussians, with an analysis of their positions, opacities, and scales. Here, coarse Gaussians refer to the selected input Gaussians for densification, while fine Gaussians represent the union of each layer's output. Next, we present ablation results to demonstrate: 1) the superiority of our method over naive densification, 2) the adaptiveness of gradient masking and its role in improving rendering quality, 3) the effectiveness of learnable masking in reducing redundant Gaussians and accelerating model inference, and 4) the necessity of combining both masking strategies to balance rendering quality and inference speed.

Table 2. Quantitative comparisons of our scene-level model against its baselines on the RE10k [50] dataset. The MVSplat-finetune model is fine-tuned for 150,000 iterations from the MVSplat checkpoint at 300,000 iterations. * indicates concurrent work.

Method	#Param. (M)	PSNR ↑	SSIM ↑	LPIPS ↓
pixelNeRF [44]	250	20.43	0.589	0.550
GPNR [31]	27	24.11	0.793	0.255
MuRF [39]	15.7	26.10	0.858	0.143
pixelSplat [3]	125.1	25.89	0.858	0.142
MVSplat [7]	12	26.39	0.869	0.128
MVSplat-finetune	12	26.46	0.870	0.127
DepthSplat* [40]	37	26.76	0.877	0.123
Ours	27.8	**27.08**	**0.879**	**0.120**

Table 3. Cross-dataset generalization results on the ACID [22] and DTU [16] datasets. All models are trained on the RE10k [50] training set and evaluated on each dataset without fine-tuning.

Method	ACID [22]			DTU [16]		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
pixelSplat [3]	27.64	0.830	0.160	12.89	0.382	0.560
MVSplat [7]	28.15	0.841	0.147	13.94	0.473	0.385
Ours	**28.61**	**0.847**	**0.141**	**14.05**	**0.477**	**0.380**

Comparison of Coarse and Fine Gaussians. The left side of Fig. 8 illustrates that reconstructing contours is chal-

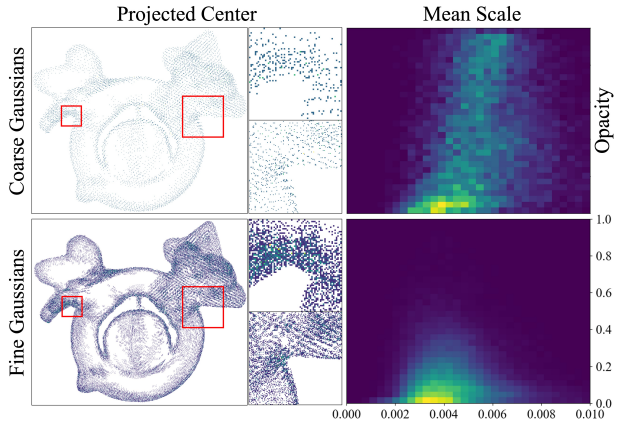


Figure 8. 2D histograms of Gaussian attributes. Each pixel represents a histogram bin, with brighter colors for higher counts.

lenging with insufficient number of Gaussians. Although these areas are covered by large Gaussians with no visible holes in the rendered image, it still appears blurry, with the object not clearly separated from the background (Fig. 5). As shown on the right side of Fig. 8, fine Gaussians have smaller scales and lower opacities compared to coarse Gaussians, indicating that our method reconstructs details by accumulating partially overlapping small Gaussians.

Effectiveness of Generative Densification. Increasing the per-voxel Gaussian count in LaRa raises the total number of Gaussians by 68%, but only results in a slight PSNR improvement (second row of Tab.4). In contrast, our method achieves a significantly higher PSNR with a comparable Gaussian count (third row of Tab.4). Moreover, although LaRa cannot be trained with more than 80k Gaussians due to memory overhead, our approach handles over 110k Gaussians on the same GPU. The scene-level model shows similar results, further illustrating the efficiency and effectiveness of our adaptive strategy (last row of Tab. 4).

Gradient Masking. The first row of Tab. 5 presents the re-evaluation of the 'Ours-fast' model from Tab. 1, where all masking blocks are removed. In comparison to the model with a single gradient masking block (second row of Tab. 5), there is a notable increase in the number of Gaussians, and the average inference time for reconstructing a single scene rises by 32%. Additionally, the PSNR decreases by 0.52 dB, highlighting that the adaptive densification enabled by gradient masking improves rendering quality compared to densely producing an excess of Gaussians across the scene.

Learnable Masking. Learnable masking plays a pivotal role in reducing computational and memory demands by controlling the number of Gaussians generated in each densification layer. Ideally, it should filter out Gaussians not requiring further densification without compromising image quality. As shown in the second and third rows of Tab. 5, although masking leads to a minor decrease in PSNR, it results in a substantial 25% reduction in the number of Gaussians and notably accelerates model inference. This indicates that the efficiency gains from reducing the number of Gaussians outweigh the slight drop in image quality, emphasizing the significance of learnable masking.

Table 4. Comparison against naïve densification. Object and scene models are evaluated on the GSO [12] and RE10k [50] datasets.

Method	#Gauss. ↓	PSNR ↑	SSIM ↑	LPIPS ↓
LaRa (2 Gaussians per voxel)	**46,878**	29.70	0.959	0.060
LaRa (3 Gaussians per voxel)	79,111	29.77	0.961	0.068
Ours ($K = 6,000$)	78,084	**30.95**	**0.965**	**0.060**
MVSplat (1 Gaussian per pixel) [7]	**65,000**	26.39	0.869	0.280
MVSplat (3 Gaussians per pixel) [7]	195,000	26.54	0.872	0.127
Ours ($K = 30,000$)	173,000	**27.08**	**0.879**	**0.120**

Table 5. Ablation results on two masking strategies with varying configurations, evaluated on the GSO [12] dataset. 'Grad.' and 'Learn.' refer to gradient and learnable masking (Sec. 3), respectively, while 'None' indicates no masking is applied. Inference time ('Time') is measured in milliseconds on a single A6000 GPU.

Masking Configuration		#Gauss. ↓	Time ↓	PSNR ↑
1st Masking	2nd Masking			
None	None	413,066	255	30.21
Grad. ($K = 12$k)	None	151,968	193	**30.73**
Grad. ($K = 12$k)	Learn. ($\tilde{K}_{\text{den}} = 19$k)	**114,351**	**187**	30.62
Grad. ($K = 15$k)	Learn. ($\tilde{K}_{\text{den}} = 24$k)	130,056	198	30.68
Grad. ($K = 30$k)	Learn. ($\tilde{K}_{\text{den}} = 48$k)	189,322	208	30.63
Grad. ($K = 12$k)	Grad. ($\tilde{K}_{\text{den}} = 19$k)	114,371	199	30.12
Learn. ($K = 12$k)	Learn. ($\tilde{K}_{\text{den}} = 19$k)	114,565	188	29.31

Combining Gradient and Learnable Masking. Both masking strategies can be used interchangeably to reduce the number of Gaussians and accelerate inference. However, gradient masking is primarily designed to select detailed regions for adaptive densification, in addition to reducing overhead. The last row of Tab. 5 shows a significant PSNR drop when replacing gradient masking with learnable masking, highlighting that the former is better suited for selecting to-be-densified Gaussians. Conversely, learnable masking can be replaced with gradient masking, but rendering quality still lags behind that of the combined model (second last row of Tab. 5), which demonstrates the necessity of using a combination of both masking strategies.

5. Conclusion

We proposed generative densification, an efficient densification strategy tailored for generalized feed-forward Guassian models. We integrated the proposed method into LaRa and MVSplat, providing practical guidance for real-world applications. Extensive experiments demonstrate that our method effectively reconstruct high-frequency details by generating fine Gaussians, establishing new benchmarks in both object-level and scene-level reconstruction tasks. We believe our work paves the way for future research on feed-forward adaptive densification leveraging learned prior.

Acknowledgements

This research was supported by the Ministry of Science and ICT (MSIT) of Korea, under the National Research Foundation (NRF) grant (RS-2024-00337548), the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea Government (MSIT) (RS-2020-II201361, Artificial Intelligence Graduate School Program (Yonsei University)), and Samsung Research Funding & Incubation Center of Samsung Electronics under Project Number SRFC-IT2401-01, and the Korea-US Collaborative Research Fund (KUCRF), funded by the Ministry of Science and ICT and Ministry of Health & Welfare, Republic of Korea (grant number: RS-2024-00468417).

References

[1] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. Revising densification in gaussian splatting. *arXiv preprint arXiv:2404.06109*, 2024. 3

[2] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 3

[3] 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*, 2024. 1, 2, 3, 6, 7

[4] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 6

[5] Anpei Chen, Haofei Xu, Stefano Esposito, Siyu Tang, and Andreas Geiger. Lara: Efficient large-baseline radiance fields. In *European Conference on Computer Vision*, 2025. 3, 5, 6

[6] Yuedong Chen, Chuanxia Zheng, Haofei Xu, Bohan Zhuang, Andrea Vedaldi, Tat-Jen Cham, and Jianfei Cai. Mvsplat360: Feed-forward 360 scene synthesis from sparse views. *arXiv preprint arXiv:2411.04924*, 2024. 3

[7] 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*, 2025. 1, 3, 5, 6, 7, 8

[8] Zheng Chen, Chenming Wu, Zhelun Shen, Chen Zhao, Weicai Ye, Haocheng Feng, Errui Ding, and Song-Hai Zhang. Splatter-360: Generalizable 360 gaussian splatting for wide-baseline panoramic images. *arXiv preprint arXiv:2412.06250*, 2024. 3

[9] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 1

[10] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36, 2024. 1

[11] Xiaobin Deng, Changyu Diao, Min Li, Ruohan Yu, and Duanqing Xu. Efficient density control for 3d gaussian splatting. *arXiv preprint arXiv:2411.10133*, 2024. 3

[12] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *International Conference on Robotics and Automation*, 2022. 5, 6, 8

[13] Guangchi Fang and Bing Wang. Mini-splatting: Representing scenes with a constrained number of gaussians. In *European Conference on Computer Vision*, 2024. 3

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016. 3

[15] Paul Henderson, Melonie de Almeida, Daniela Ivanova, and Titas Anciukevičius. Sampling 3d gaussian scenes in seconds with latent diffusion models. *arXiv preprint arXiv:2406.13099*, 2024. 3

[16] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2014. 5, 7

[17] Jaewoo Jung, Jisang Han, Honggyu An, Jiwon Kang, Seonghoon Park, and Seungryong Kim. Relaxing accurate initialization constraint for 3d gaussian splatting. *arXiv preprint arXiv:2403.09413*, 2024. 3

[18] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 2023. 1, 3

[19] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Jeff Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3d gaussian splatting as markov chain monte carlo. *arXiv preprint arXiv:2404.09591*, 2024. 3

[20] Hanxue Liang, Jiawei Ren, Ashkan Mirzaei, Antonio Torralba, Ziwei Liu, Igor Gilitschenski, Sanja Fidler, Cengiz Oztireli, Huan Ling, Zan Gojcic, et al. Feed-forward bullet-time reconstruction of dynamic scenes from monocular videos. *arXiv preprint arXiv:2412.03526*, 2024. 3

[21] Chenguo Lin, Panwang Pan, Bangbang Yang, Zeming Li, and Yadong Mu. Diffsplat: Repurposing image diffusion models for scalable gaussian splat generation. *arXiv preprint arXiv:2501.16764*, 2025. 3

[22] Andrew Liu, Richard Tucker, Varun Jampani, Ameesh Makadia, Noah Snavely, and Angjoo Kanazawa. Infinite nature: Perpetual view generation of natural scenes from a sin-

gle image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 5, 7

[23] Tianqi Liu, Guangcong Wang, Shoukang Hu, Liao Shen, Xinyi Ye, Yuhang Zang, Zhiguo Cao, Wei Li, and Ziwei Liu. Mvsgaussian: Fast generalizable gaussian splatting reconstruction from multi-view stereo. In *European Conference on Computer Vision*, 2025. 3

[24] Longfei Lu, Huachen Gao, Tao Dai, Yaohua Zha, Zhi Hou, Junta Wu, and Shu-Tao Xia. Large point-to-gaussian model for image-to-3d generation. In *Proceedings of the ACM International Conference on Multimedia*, 2024. 3

[25] Tao Lu, Ankit Dhiman, R Srinath, Emre Arslan, Angela Xing, Yuanbo Xiangli, R Venkatesh Babu, and Srinath Sridhar. Turbo-gs: Accelerating 3d gaussian fitting for high-quality radiance fields. *arXiv preprint arXiv:2412.13547*, 2024. 3

[26] Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Francisco Vicente Carrasco, Markus Steinberger, and Fernando De La Torre. Taming 3dgs: High-quality radiance fields with limited resources. *arXiv preprint arXiv:2406.15643*, 2024. 3

[27] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023. 3

[28] Giuseppe Peano and G Peano. *Sur une courbe, qui remplit toute une aire plane*. Springer, 1990. 2, 4

[29] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 5, 6

[30] Jiawei Ren, Cheng Xie, Ashkan Mirzaei, Karsten Kreis, Ziwei Liu, Antonio Torralba, Sanja Fidler, Seung Wook Kim, Huan Ling, et al. L4gm: Large 4d gaussian reconstruction model. *Advances in Neural Information Processing Systems*, 2024. 3

[31] Mohammed Suhail, Carlos Esteves, Leonid Sigal, and Ameesh Makadia. Generalizable patch-based neural rendering. In *European Conference on Computer Vision*, 2022. 6, 7

[32] Stanislaw Szymanowicz, Eldar Insafutdinov, Chuanxia Zheng, Dylan Campbell, João F Henriques, Christian Rupprecht, and Andrea Vedaldi. Flash3d: Feed-forward generalisable 3d scene reconstruction from a single image. *arXiv preprint arXiv:2406.04343*, 2024. 3

[33] Stanislaw Szymanowicz, Chrisitian Rupprecht, and Andrea Vedaldi. Splatter image: Ultra-fast single-view 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 3

[34] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. In *European Conference on Computer Vision*, 2025. 3, 5, 6

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

[36] Christopher Wewer, Kevin Raj, Eddy Ilg, Bernt Schiele, and Jan Eric Lenssen. latentsplat: Autoencoding variational gaussians for fast generalizable 3d reconstruction. In *European Conference on Computer Vision*, 2024. 3

[37] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler faster stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 2, 4

[38] Chao Xu, Yuan Dong, Qi Zuo, Junfei Zhang, Xiaodan Ye, Wenbo Geng, Yuxiang Zhang, Xiaodong Gu, Lingteng Qui, Zhengyi Zhao, Qing Ran, Jiayi Jiang, Zilong Dong, and Liefeng Bo. G-buffer objaverse: High-quality rendering dataset of objaverse. https://aigc3d.github.io/gobjaverse/. 3, 5, 6

[39] Haofei Xu, Anpei Chen, Yuedong Chen, Christos Sakaridis, Yulun Zhang, Marc Pollefeys, Andreas Geiger, and Fisher Yu. Murf: Multi-baseline radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 6, 7

[40] Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. Depthsplat: Connecting gaussian splatting and depth. *arXiv preprint arXiv:2410.13862*, 2024. 3, 6, 7

[41] Yinghao Xu, Zifan Shi, Wang Yifan, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation. *arXiv preprint arXiv:2403.14621*, 2024. 3

[42] Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. Absgs: Recovering fine details in 3d gaussian splatting. In *Proceedings of the ACM International Conference on Multimedia*, 2024. 3

[43] Penghang Yin, Jiancheng Lyu, Shuai Zhang, Stanley Osher, Yingyong Qi, and Jack Xin. Understanding straight-through estimator in training activation quantized neural nets. *arXiv preprint arXiv:1903.05662*, 2019. 5

[44] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 7

[45] Chubin Zhang, Hongliang Song, Yi Wei, Yu Chen, Jiwen Lu, and Yansong Tang. Geolrm: Geometry-aware large reconstruction model for high-quality 3d gaussian generation. *arXiv preprint arXiv:2406.15333*, 2024. 3

[46] Chuanrui Zhang, Yingshuang Zou, Zhuoling Li, Minmin Yi, and Haoqian Wang. Transplat: Generalizable 3d gaussian splatting from sparse multi-view images with transformers. *arXiv preprint arXiv:2408.13770*, 2024. 3

[47] 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*, 2025. 3, 5, 6

[48] Zheng Zhang, Wenbo Hu, Yixing Lao, Tong He, and Hengshuang Zhao. Pixel-gs: Density control with pixel-

aware gradient for 3d gaussian splatting. *arXiv preprint arXiv:2403.15530*, 2024. 3, 6

[49] Shunyuan Zheng, Boyao Zhou, Ruizhi Shao, Boning Liu, Shengping Zhang, Liqiang Nie, and Yebin Liu. Gps-gaussian: Generalizable pixel-wise 3d gaussian splatting for real-time human novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 1

[50] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018. 1, 3, 5, 7, 8

[51] 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. *arXiv preprint arXiv:2410.12781*, 2024. 3

[52] Zi-Xin Zou, Zhipeng Yu, Yuan-Chen Guo, Yangguang Li, Ding Liang, Yan-Pei Cao, and Song-Hai Zhang. Triplane meets gaussian splatting: Fast and generalizable single-view 3d reconstruction with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 3