

Matryoshka Gaussian Splatting

Zhilin Guo¹, Boqiao Zhang¹, Hakan Aktas¹,
Kyle Fogarty¹, Jeffrey Hu¹, Nursena Koprucu Aslan¹, Wenzhao Li¹,
Canberk Baykal¹, Albert Miao¹, Josef Bengtson², Chenliang Zhou¹,
Weihao Xia^{1*}, Cristina Nader Vasconcelos³, Cengiz Oztireli^{1,3}

¹University of Cambridge ²Chalmers University of Technology ³Google
<https://ZhilinGuo.github.io/MGS>

Abstract. The ability to render scenes at adjustable fidelity from a single model, known as level of detail (LoD), is crucial for practical deployment of 3D Gaussian Splatting (3DGS). Existing discrete LoD methods expose only a limited set of operating points, while concurrent continuous LoD approaches enable smoother scaling but often suffer noticeable quality degradation at full capacity, making LoD a costly design decision. We introduce *Matryoshka Gaussian Splatting (MGS)*, a training framework that enables continuous LoD for standard 3DGS pipelines without sacrificing full-capacity rendering quality. MGS learns a single ordered set of Gaussians such that rendering any prefix, the first k splats, produces a coherent reconstruction whose fidelity improves smoothly with increasing budget. Our key idea is *stochastic budget training*: each iteration samples a random splat budget and optimises both the corresponding prefix and the full set. This strategy requires only two forward passes and introduces no architectural modifications. Experiments across four benchmarks and six baselines show that MGS matches the full-capacity performance of its backbone while enabling a continuous quality–speed trade-off from a single model. Extensive ablations on ordering strategies, training objectives, and model capacity further validate the designs.

Keywords: 3D Gaussian splatting · continuous level of detail · nested representations · budgeted rendering · stochastic training

1 Introduction

Real-time neural rendering is governed by a fundamental tension between scene fidelity and the computational budget available [25, 30]. This budget varies by orders of magnitude across the hardware spectrum, from high-end GPU workstations to mobile devices and mixed-reality headsets [28, 37]; and fluctuates dynamically at runtime by viewpoint and scene complexity [9, 45]. Level of detail (LoD) techniques address this tension by scaling the rendered representation to match available resources and have long been a cornerstone of interactive graphics [24]. The dominant paradigm, discrete LoD, precomputes a set of quality levels and

* Corresponding author: wx258@cam.ac.uk



Fig. 1: Continuous LoD need not sacrifice full-capacity quality to enable budget trade-off. Our method, MGS (top), learns an ordered set of Gaussian primitives whose prefixes yield coherent reconstructions at any splat budget. Compared to CLoD-3DGS [26] (mid) and CLoD-GS [4] (bot), MGS achieves the highest fidelity at every operating point with quality degrading gracefully under budget reduction. Scene: *Garden* [1].

switches between them at runtime [5]. A coarse set of fixed levels, however, cannot smoothly track a budget that shifts continuously with scene content and viewpoint, and the abrupt transitions between levels produce visible pop-in and pop-out artifacts [9, 10].

3D Gaussian Splatting (3DGS) [15] achieves photorealistic novel view synthesis by rasterizing millions of anisotropic Gaussian primitives at real-time frame rates, with computational cost scaling directly with the number of primitives [8]. In principle, this primitive-based nature of the representation offers continuous budget control, since omitting any subset of splats yields an immediate speedup [11, 15]. Yet a conventionally trained 3DGS model has no ordering among its primitives, so quality collapses rapidly as splats are removed [21].

Existing approaches attempt to introduce LoD to 3DGS through several strategies. Discrete LoD methods [16, 19, 31, 33] build hierarchical structures over Gaussian primitives, exposing a limited number of quality levels but requiring auxiliary index structures and offering only coarse budget granularity. Compression and pruning pipelines [6, 22, 36] can approximate multi-budget behaviour, but each operating point is typically obtained independently without ensuring that subsets are nested or globally coherent. Concurrent continuous LoD methods [4, 26] enable smoother scaling within a single model, but often incur substantial quality degradation at full capacity. As a result, adopting LoD in 3DGS remains a costly design choice that often sacrifices reconstruction quality.

In this work, we introduce *Matryoshka Gaussian Splatting* (MGS), a training framework that enables continuous LoD control without degrading full-capacity quality (Fig. 1). Where Matryoshka Representation Learning [20] nests several embedding dimensions, MGS transfers this principle to scene primitives at arbitrary granularity. Concretely, MGS learns an ordered set of Gaussian primitives such that any prefix (the first k splats) forms a coherent scene representation,

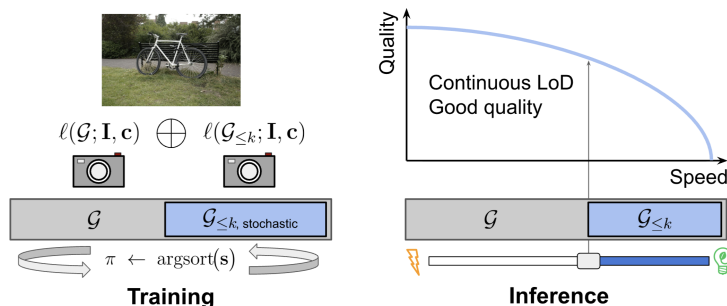


Fig. 2: Framework of Matryoshka Gaussian Splatting.

with reconstruction fidelity improving smoothly as k increases. Rendering at different budgets is then achieved by truncating the ordered sequence, producing a continuous spectrum of quality–speed operating points from a single model.

The key mechanism is *stochastic budget training*: each iteration samples a random splat budget uniformly and optimises both the corresponding prefix and the full set, requiring only two forward passes each training step. Because this procedure modifies only the training objective and not the model architecture, it integrates into most existing 3DGS pipelines with minimal implementation effort. At deployment, one adjusts k to match available resources—no per-budget retraining, no auxiliary data structures.

Extensive experiments across four benchmarks [1, 12, 18, 39] and six baselines [4, 16, 22, 26, 31, 36] spanning discrete LoD and continuous LoD methods demonstrate that MGS enables a continuous quality–speed frontier from a single model while matching its backbone’s full-capacity reconstruction quality.

Our contributions are summarised as follows:

- We introduce *Matryoshka Gaussian Splatting* (MGS), which learns a nested Gaussian primitive representation where any prefix yields a coherent scene reconstruction, enabling continuous LoD control.
- We propose *stochastic budget training*, a simple yet effective model-agnostic training procedure that optimises across a continuous budget range with only two renders per iteration.
- We provide extensive evaluation on four benchmarks against six baselines, demonstrating state-of-the-art quality–speed performance, with ablations on importance scoring, budget sampling, and training objective.

2 Related Work

2.1 3D Gaussian Splatting

3D Gaussian Splatting (3DGS) [15] replaces implicit volumetric representations in Neural Radiance Fields [25] with explicit anisotropic Gaussian primitives

rendered via differentiable rasterization. This formulation enables photorealistic novel view synthesis at real-time frame rates, with rendering cost scaling directly with the number of primitives, making primitive count a natural axis for controlling computational budget. Subsequent work [7, 17, 40, 43] has extended 3DGS along several directions. For capacity control, the original densification heuristics offer no direct control over the final primitive count. 3DGS-MCMC [17] addresses this via Langevin-dynamics sampling under a fixed budget. Mini-Splatting [7] further improves densification under constrained capacities. For multi-scale rendering, Mip-Splatting [43] and Multi-Scale 3DGS [40] introduce anti-aliasing filters that improve quality when viewed at varying observation scales. Our method MGS builds upon these advances by learning a primitive ordering that maintains coherent reconstructions across a continuous range of rendering budgets.

2.2 Level-of-Detail Rendering

Level-of-detail (LoD) rendering enables adaptive control of rendering cost by varying scene complexity according to available computational budgets. Discrete LoD [16, 19, 23, 31, 33, 34] and progressive streaming [3, 35] methods construct hierarchical or layered structures over Gaussians to expose multiple quality levels. However, these methods require auxiliary index structures and typically provide only a small number of discrete operating points [16, 31]. Compression and pruning pipelines [6, 22, 36] can approximate multi-budget behaviour by applying importance-based pruning or quantization at varying targets, but each operating point is typically obtained independently without ensuring nested or globally consistent primitive subsets [27, 29]. Concurrent continuous LoD methods [4, 26] target smoother budget scaling within a single model, either by learning view-dependent opacity decay [14] or by training on random primitive subsets. However, both incur noticeable quality degradation at full capacity and rapid quality collapse at reduced budgets, making continuous LoD a costly design choice. In contrast, MGS learns a single ordered set of Gaussian primitives such that *every* prefix yields a coherent reconstruction. This produces a dense continuum of valid budgets while still closely matching the backbone’s full-capacity quality.

2.3 Nested (Matryoshka) Representations

Nested representations learn ordered structures in which every prefix of the representation remains independently usable. This idea was introduced by nested dropout [32] and scaled to high-dimensional embeddings by Matryoshka Representation Learning (MRL) [20]. Related ideas of packing multiple capacity levels into a single model have also been explored in slimmable networks [2, 42], which train neural networks that operate at different channel widths, although these methods typically require separate forward passes for each width configuration. In classical graphics, progressive meshes [13] provide an analogous concept by representing geometry as an ordered sequence of refinement operations that enables continuous mesh simplification.

These representations nest along an order given *a priori*, such as embedding dimensions, network channels, or a mesh-refinement index. But 3D Gaussians possess no intrinsic ordering, and removing an arbitrary subset of Gaussians destructively reorders the α -compositing sequence of every pixel. Our MGS introduces the nested representation principle to Gaussian scene primitives. Critically, we show that an ordering compatible with the rasterizer exists and can be discovered in place, yielding a coherent prefix at every budget.

3 Method

Our MGS consists of two components: (i) an *ordered Gaussian representation* where importance-ranked prefixes serve as nested scene representations at different fidelity levels (Sec. 3.2), and (ii) a *stochastic training procedure* that efficiently optimises across all prefix lengths using a single random prefix and the full set per iteration (Sec. 3.3). We first review the necessary background in Sec. 3.1.

3.1 Preliminaries on 3DGS

3D Gaussian Splatting (3DGS) [15] represents a scene as a set of N anisotropic 3D Gaussians $\mathcal{G} = \{g_i\}_{i=1}^N$. Each Gaussian g_i carries a mean $\boldsymbol{\mu}_i \in \mathbb{R}^3$, a covariance $\boldsymbol{\Sigma}_i$, an opacity $\sigma_i \in [0, 1]$, and view-dependent colour parameters (*e.g.*, spherical harmonics). Given a camera $\mathbf{c}$, a pixel colour is obtained via front-to-back α -compositing of depth-sorted Gaussians:

$$\hat{\mathbf{I}}(\mathbf{u}) = \sum_{i=1}^N \mathbf{C}_i(\mathbf{u}) \alpha_i(\mathbf{u}) \prod_{j=1}^{i-1} (1 - \alpha_j(\mathbf{u})), \quad (1)$$

where $\mathbf{u}$ is a pixel coordinate, $\mathbf{C}_i(\mathbf{u})$ the view-dependent colour, and $\alpha_i(\mathbf{u})$ the screen-space opacity obtained by evaluating the projected 2D Gaussian at $\mathbf{u}$. Given posed training images $\mathcal{D} = \{(\mathbf{I}_n, \mathbf{c}_n)\}_{n=1}^M$, the model minimises a per-image reconstruction loss:

$$\ell(\mathcal{G}; \mathbf{I}, \mathbf{c}) := (1 - \lambda) \|\mathcal{R}(\mathcal{G}; \mathbf{c}) - \mathbf{I}\|_1 + \lambda \mathcal{L}_{\text{D-SSIM}}(\mathcal{R}(\mathcal{G}; \mathbf{c}), \mathbf{I}), \quad (2)$$

where $\mathcal{R}$ denotes the differentiable splatting renderer and λ is the mixing weight.

3.2 Matryoshka Gaussian Splatting

We construct a nested Gaussian representation that enables continuous control over rendering budgets from a single model. Specifically, we (i) define an importance score to rank Gaussian primitives, (ii) organise them into an ordered prefix representation that supports variable-budget rendering, and (iii) adopt a capacity control mechanism to maintain a fixed Gaussian budget during training.

Importance Score. To construct a nested Gaussian representation, we assign each Gaussian primitive g_i a scalar score $s(g_i)$ based on a per-primitive property. Gaussians are then sorted by this score so that the most important primitives appear first in the sequence.

The MGS formulation is agnostic to the choice of $s(g)$; any statistic that induces a meaningful importance ordering is admissible. Empirically, we find that sorting Gaussians by opacity in descending order yields stable behaviour across budgets. This is likely because opacity reflects the visibility and radiance contribution of each primitive, making it a natural criterion for ordering primitives so that early prefixes already capture the dominant scene structure. We further observe that MGS training reshapes the opacity distribution toward the high-opacity end, reinforcing this ordering (analysed in Sec. 4.2). Therefore, we use primitive’s opacity as importance score in implementation:

$$s(g_i) = \sigma_i, \quad (3)$$

where σ_i denotes the opacity parameter of Gaussian g_i . Alternative scoring criteria are evaluated in ablation studies.

Nested Primitive Representation. Let π be a permutation of $\{1, \dots, N\}$ that ranks Gaussians by a scalar importance score $s(g)$ in non-increasing order:

$$s(g_{\pi(1)}) \geq s(g_{\pi(2)}) \geq \dots \geq s(g_{\pi(N)}). \quad (4)$$

The k -prefix is the subset of the k highest-ranked Gaussians:

$$\mathcal{G}_{\leq k} := \{g_{\pi(1)}, \dots, g_{\pi(k)}\}, \quad k \in \{1, \dots, N\}. \quad (5)$$

Rendering with budget k produces $\hat{\mathbf{I}}_{\leq k} = \mathcal{R}(\mathcal{G}_{\leq k}; \mathbf{c})$. Because prefixes are nested ($\mathcal{G}_{\leq k} \subset \mathcal{G}_{\leq k'}$ for $k < k'$) and rasterization cost scales with the Gaussian count, varying k traces a continuous quality–speed curve from a single trained model without any per-budget retraining or model switching.

3.3 Stochastic Budget Training

Training a single representation to perform well at every budget level is the central challenge of LoD. Optimising all N possible budgets per step would be prohibitively expensive. We instead propose a stochastic procedure that covers a dense continuum of prefix sizes with only two forward passes per iteration.

Budget Sampling. At each training step we draw a random keep ratio and compute the corresponding prefix size k such that:

$$k = \lceil r N \rceil, \quad r \sim \text{Unif}(r_{\min}, 1), \quad (6)$$

where $r_{\min} \in (0, 1]$ is the smallest prefix fraction seen during training; $r_{\min} = 1$ recovers standard 3DGS training. Uniform sampling ensures that every budget in $[\lceil r_{\min} N \rceil, N]$ is visited with equal probability over training, yielding unbiased coverage of the full budget spectrum.

Training Objective. At each step, we sample a training view $(\mathbf{I}, \mathbf{c}) \sim \mathcal{D}$ and prefix size $k = \lceil rN \rceil$, render both the prefix and the full set, and minimise:

$$\ell_{\text{MGS}} := \ell(\mathcal{G}_{\leq k}; \mathbf{I}, \mathbf{c}) + \gamma \ell(\mathcal{G}_{\leq N}; \mathbf{I}, \mathbf{c}), \quad (7)$$

where $\gamma \geq 0$ balances the prefix and full-set terms. The prefix term pressures the model to produce strong reconstructions from partial subsets, while the full-set term anchors full-quality performance. Over training, this stochastic procedure estimates the expected multi-budget objective across all budget fractions $r \in [r_{\min}, 1]$ and all training views, with prefix membership determined by the ordering in Eq. (4). Each step incurs exactly two renders regardless of N .

Dynamic Reordering. Because gradient updates modify the Gaussian parameters at every step, the importance scores $s(g_i)$ evolve throughout training. After each training iteration, we recompute the permutation so that the ordering in Eq. (4) holds under the current parameters:

$$\pi \leftarrow \text{argsort}(\mathbf{s}), \quad \mathbf{s} = (s(g_1), \dots, s(g_N)), \quad (8)$$

in non-increasing order. This ensures every prefix $\mathcal{G}_{\leq k}$ always contains the k most important primitives under the current parameters.

4 Experiments

4.1 Experimental Setup

Benchmarks. We evaluate MGS on four standard 3DGS benchmarks: MipNeRF 360 [1], Tanks & Temples [18], Deep Blending [12], and BungeeNeRF [39]. For all datasets, we follow the standard evaluation protocol and use the every-8th-image split, where images with indices 0, 8, 16, ... are reserved for testing.

Baselines. We compare MGS against six representative Gaussian LoD baselines. *Discrete* LoD methods expose a fixed set of quality levels: H3DGS [16] (τ -threshold hierarchy), Octree-GS [31] (anchor-based octree LoD), MaskGaussian [22] (learnable existence mask), and FlexGaussian [36] (training-free pruning/quantization), each at the discrete operating points from the respective papers. For MaskGaussian with inference-time masking, we sweep the published mask threshold to trace its operating points, preserving the single-model property under comparison. *Continuous* LoD methods support rendering at arbitrary splat budgets, including CLoD-GS [4] (distance-dependent opacity decay) and CLoD-3DGS [26] (learned importance ordering), are evaluated at the same prefix ratios as MGS: 100%, 90%, ..., 20%, 10%, 5%, and 1%.

Table 1: Quantitative comparison on four benchmarks. PSNR, SSIM [38], LPIPS [44] reported at highest splat budget per baseline. AUC_{fps} summarises quality–FPS trade-off frontier, and AUC_{splats} encapsulates quality/splat efficiency. The **best** and **second best** results are highlighted. $\uparrow$ higher is better, $\downarrow$ lower is better.

Method		MipNeRF 360					Tanks & Temples				
		PSNR↑	SSIM↑	LPIPS↓	AUC _{fps} ↑	AUC _{splats} ↑	PSNR↑	SSIM↑	LPIPS↓	AUC _{fps} ↑	AUC _{splats} ↑
Disc.	3DGS-MCMC [17]	28.40	0.843	0.133	–	–	24.76	0.877	0.083	–	–
	H3DGS [16]	25.41	0.780	0.230	11.62	60.51	20.41	0.808	0.186	8.23	58.84
	FlexGaussian [36]	26.86	0.793	0.237	30.04	62.05	22.89	0.823	0.198	<u>52.38</u>	64.93
	MaskGaussian [22]	27.42	<u>0.815</u>	0.218	34.11	67.10	23.66	0.846	0.181	45.15	68.89
	Octree-GS [31]	<u>27.62</u>	0.813	0.221	21.61	<u>68.43</u>	24.59	<u>0.865</u>	<u>0.157</u>	23.80	<u>67.81</u>
Cont.	CLoD-GS [4]	26.55	0.803	0.233	19.39	63.43	22.78	0.832	0.193	23.39	65.37
	CLoD-3DGS [26]	27.44	0.814	<u>0.215</u>	<u>46.38</u>	56.56	23.51	0.845	0.176	50.97	61.89
	MGS (Ours)	28.20	0.841	0.130	64.81	77.79	<u>24.56</u>	0.874	0.086	70.16	77.54
Method		Deep Blending					BungeeNeRF				
		PSNR↑	SSIM↑	LPIPS↓	AUC _{fps} ↑	AUC _{splats} ↑	PSNR↑	SSIM↑	LPIPS↓	AUC _{fps} ↑	AUC _{splats} ↑
3DGS-MCMC [17]		27.63	0.893	0.204	–	–	27.04	0.903	0.091	–	–
Disc.	H3DGS [16]	27.77	0.882	0.277	10.39	73.82	27.74	0.910	0.114	8.47	62.57
	FlexGaussian [36]	29.25	0.899	0.251	43.61	75.53	25.87	0.872	0.129	12.89	53.58
	MaskGaussian [22]	<u>29.69</u>	<u>0.907</u>	<u>0.245</u>	54.96	78.77	<u>27.80</u>	<u>0.917</u>	0.099	21.58	71.18
	Octree-GS [31]	30.35	0.910	0.252	29.54	<u>83.21</u>	28.23	0.922	<u>0.088</u>	19.69	<u>73.70</u>
Cont.	CLoD-GS [4]	29.03	0.901	0.249	26.09	77.34	25.35	0.859	0.148	13.83	65.40
	CLoD-3DGS [26]	29.54	0.903	0.247	<u>69.36</u>	72.33	26.57	0.892	0.131	<u>24.45</u>	29.72
	MGS (Ours)	28.41	0.902	0.176	82.63	83.22	27.13	0.906	0.088	62.99	79.93

Metrics. We report image-quality metrics PSNR, SSIM [38], LPIPS [44]. In addition, a key requirement to evaluating LoD methods is comparing quality–speed trade-off at different operating points. To this end, we first define a composite *quality* score calculated at each operating point:

$$\bar{Q} = \frac{1}{3}(\tilde{p} + \tilde{s} + (1 - \tilde{\ell})), \quad (9)$$

where $\tilde{p}$, $\tilde{s}$, $\tilde{\ell}$ are PSNR, SSIM, and LPIPS linearly clamped to $[0, 1]$ using fixed ranges $(14, 32)$, $(0.35, 0.92)$, and $(0.06, 0.60)$, respectively. We then summarise the full quality–speed trade-off with two area-under-the-curve (AUC) scores.

$AUC_{fps}(\uparrow)$: we construct a monotone envelope of quality versus FPS. Noting that an operating point can be replicated at any lower speed, the envelope extends leftward from every achieved operating point along lowering throughput. We clip FPS to $[0, 500]$ and compute the normalised area under this envelope.

$AUC_{splats}(\uparrow)$: we construct a monotone envelope of quality vs splat count. Noting that any operating point can be replicated with additional splats, the envelope extends rightward from all operating points along increasing splat count. Also noting that quality diminishes as splat count approaches 0, we connect the origin $(0, 0)$ to the lowest-budget operating point. We clip budget to $[0, 5M]$ and compute the normalised area under this envelope.

Both AUC_{fps} and AUC_{splats} are scaled by 100 for enhanced readability.

Implementation. We implement MGS on *gsplat* [41] codebase using 3DGS-MCMC [17] training strategy. Unless otherwise noted, we order splats by opacity

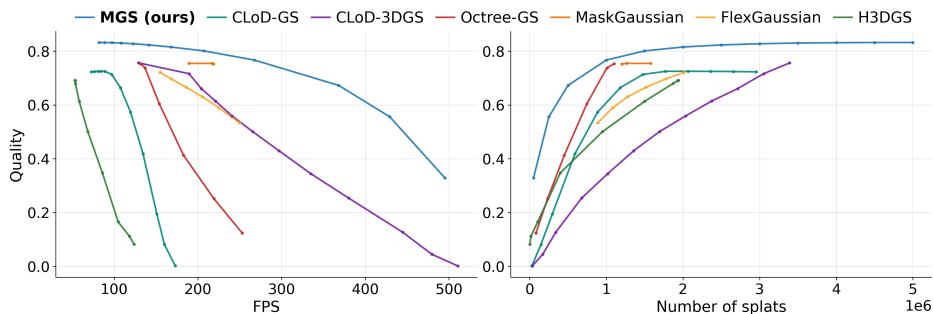


Fig. 3: Quality–budget trade-off on MipNeRF 360 [1], averaged across all nine scenes. **Left:** Quality ($\bar{Q}$, Eq. (9)) vs. FPS. **Right:** Quality vs. number of Gaussian splats. Curves trace continuous LoD models across prefix ratios 1%–100%, and trace discrete LoD models at their recommended operating points respectively. MGS (ours, dark blue) achieves the highest quality at every speed and splat budget, while spanning a much wider FPS range than any baseline.

in descending order (Eq. (4)), use equal prefix/full weights ($\gamma=1$, Eq. (7)), set the full-scene capacity to $N=5\text{M}$, and train for 50k iterations.

We adopt 3DGS-MCMC [17] because it enforces a fixed splat-count budget, essential for a controlled quality–budget comparison. A control with MGS on a vanilla 3DGS (reported in supplementary) shows the same continuous-LoD behaviour. CLoD-GS, CLoD-3DGS, H3DGS, MaskGaussian, and FlexGaussian build on vanilla 3DGS, while Octree-GS builds on Scaffold-GS [23]. All experiments are conducted on identical Ubuntu servers with NVIDIA A100 GPU.

4.2 Results and Comparisons

Full-splat Quality Comparison. Tab. 1 reports each method at its highest splat-count operating point, averaged per benchmark. On MipNeRF 360 [1], MGS achieves the best PSNR (28.20 dB), SSIM (0.841), and LPIPS (0.130), beating the next-best LoD baseline (Octree-GS [31], 27.62 dB) by +0.58 dB at far lower perceptual error (LPIPS 0.130 vs. 0.221); on Tanks & Temples [18] it trails Octree-GS by only 0.03 dB PSNR (24.56 vs. 24.59) with the best SSIM and LPIPS. On Deep Blending [12] and BungeeNeRF [39], Octree-GS has higher PSNR at its single highest-quality level, but its coarser levels degrade sharply, yielding far lower AUC than MGS. MGS consistently attains the lowest LPIPS across all four benchmarks.

Quality–Speed Trade-off. MGS outperforms all baselines on both AUC_{fps} and $\text{AUC}_{\text{splats}}$ by wide margins across the four benchmarks, sustaining high fidelity across varying speed and budget constraints (Fig. 3). By sweeping the prefix ratio from 1% to 100%, it traces a smooth, dense Pareto frontier from a single trained model, with no per-budget retraining or model switching. Qualitatively, MGS preserves coherent scene structure at aggressive budgets (5–10%

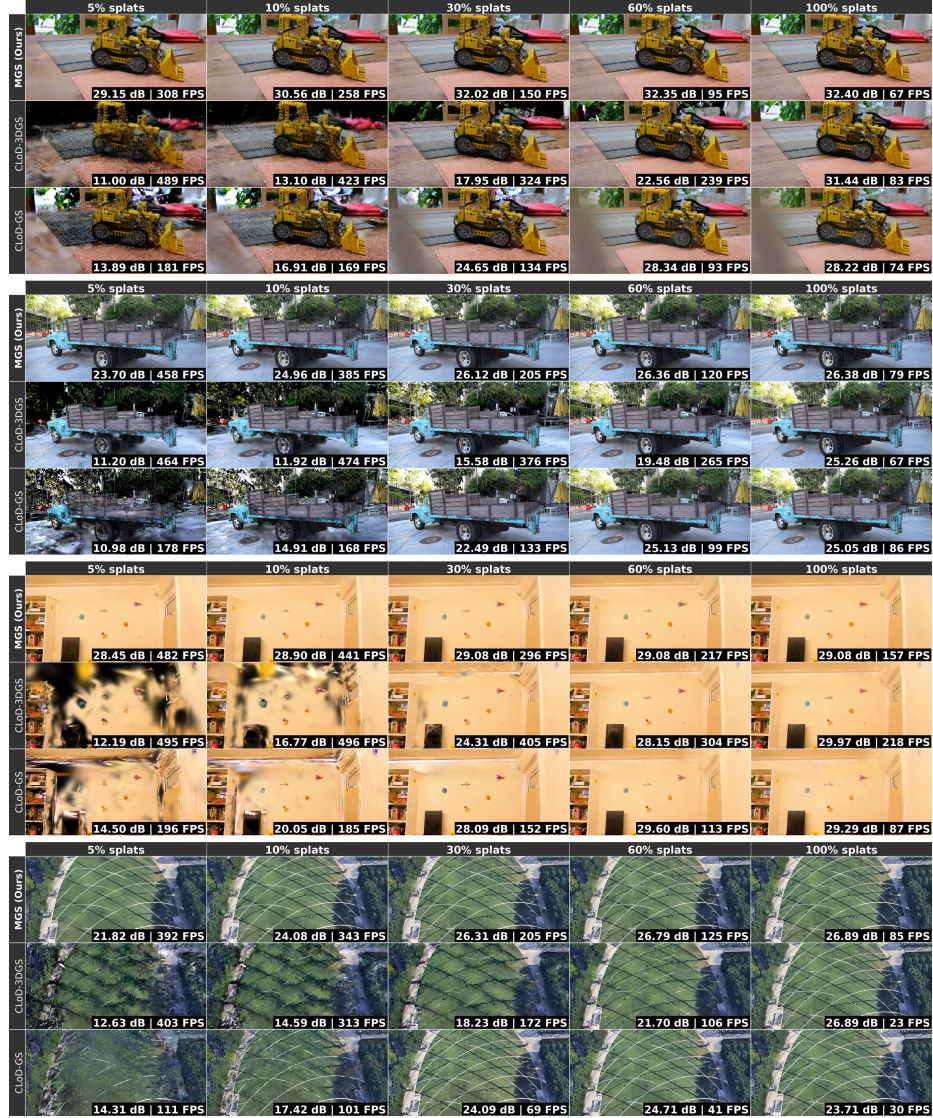


Fig. 4: Qualitative comparison of continuous LoD methods across four benchmarks. Renderings are shown at 5%, 10%, 30%, 60%, and 100% of the full splat budget. We compare MGS with CLoD-3DGS [26] and CLoD-GS [4]. Under highly constrained budgets (5–10%), MGS maintains coherent reconstructions with PSNR of 21–28 dB, while both baselines suffer from severe artifacts and quality collapse (11–17 dB).

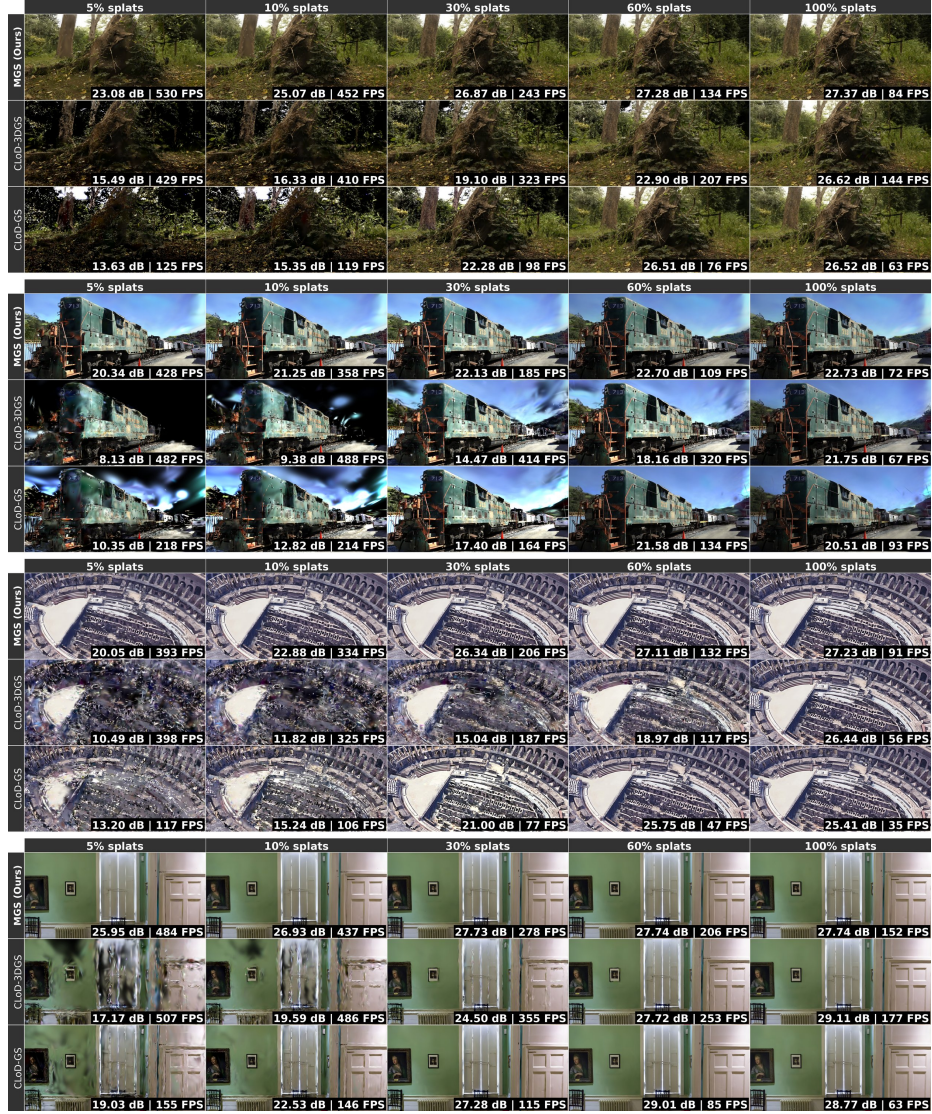


Fig. 5: Additional qualitative results including a failure case. On stump, train, and rome, MGS maintains higher fidelity across all budget levels. On DrJohnson (bottom), CLoD-3DGS [26] achieves higher peak PSNR at 100% budget (29.1 vs. 27.7 dB); however, MGS still degrades more gracefully at reduced budgets (5–30%).

splats; Figs. 4 and 5), whereas CLoD-3DGS and CLoD-GS show severe artifacts; the lone exception is DrJohnson, where CLoD-3DGS has higher peak PSNR yet degrades more sharply at reduced budgets. Among continuous-LoD competitors, CLoD-3DGS [26] is closest on MipNeRF 360 (AUC_{fps} 46.38 vs. 64.81 for MGS) but at lower image quality (27.44 vs. 28.20 dB PSNR, 0.215 vs. 0.130 LPIPS), and CLoD-GS [4] spans a narrower speed range (19.39). Discrete-LoD methods such as Octree-GS [31] and H3DGS [16] score lower (AUC_{fps} 21.61 and 11.62) as their coarse levels incur severe quality drops, and FlexGaussian [36] attains competitive full quality but exposes only a handful of discrete operating points.

Continuous vs. Discrete Operating Points. A key practical advantage of MGS is the *density* of its operating-point frontier: it produces a coherent rendering for every integer splat budget $k \in \{1, \dots, N\}$, and evaluating 12 budget ratios already yields a smooth quality–speed curve (Fig. 3). In contrast, Octree-GS [31] provides 3–6 LoD levels, H3DGS [16] offers 9 τ -thresholds, and FlexGaussian [36] exposes 2–6 compression targets, each needing separate configurations. This matters for deployment: MGS responds to per-frame or per-device budgets by simply truncating the splat array, with no extra data structures, mode switches, or latency spikes. This protocol is deliberately conservative toward discrete methods: their AUC_{fps} step-envelope is exact, whereas the envelope for MGS is a strict lower bound. MGS still attains the best AUC throughout.

Quality vs 3DGS-MCMC. Although MGS trains for *all* splat budgets, its full-quality performance closely matches the stand-alone 3DGS-MCMC [17] backbone, which trains only for the full set with no LoD capability: MGS trails by just 0.20 dB on MipNeRF 360 and Tanks & Temples (28.20 vs. 28.40; 24.56 vs. 24.76) and slightly exceeds it on Deep Blending and BungeeNeRF (28.41 vs. 27.63; 27.13 vs. 27.04 dB). We note the full-capacity gap is comparable to seed-level variation: a three-seed study on *bicycle* places the full-budget difference within one standard deviation (25.85 ± 0.02 vs. 25.83 ± 0.01 dB) and its sign flips across scenes. Thus continuous LoD need not cost full-capacity quality.

Training Cost. Relative to the backbone, for each training step MGS adds one full-set render and a per-iteration opacity re-sort, costing roughly $2\times$ the backbone (Tab. 2). Crucially, this is a one-time cost for the entire continuous-LoD frontier: covering the same 12 operating points with separate single-budget models would need about $12\times$ the backbone training, making MGS roughly $6\times$ more efficient with no loss in full-budget quality.

Opacity Distribution. Finally, we examine *why* prefix truncation degrades so gracefully. Against the identical 3DGS-MCMC backbone, MGS training reshapes the learned opacity distribution (Fig. 6): it suppresses the low-opacity tail and shifts mass toward the high-opacity end (high-opacity peak $+1.9$ pp). A prefix therefore drops primitives the ordering has already driven to low opacity, which explains the smooth quality–budget curves of Fig. 3.

Table 2: Wall-clock training time (in seconds) comparison for vanilla MCMC vs. MGS. Our MGS roughly doubles training time, a one-time cost amortised across the entire continuous-LoD frontier. Single A100, identical settings, averaged per benchmark.

Benchmark	MCMC (s)	MGS-MCMC (s)	Overhead
MipNeRF 360	2592	5325	$2.05\times$
Tanks & Temples	2445	5186	$2.12\times$
Deep Blending	2286	5409	$2.37\times$
BungeeNeRF	3294	6794	$2.06\times$

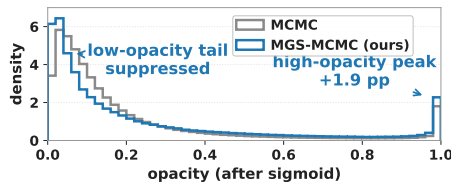


Fig. 6: MGS reshapes the opacity distribution toward the high-opacity end. Prefix truncation thus discards primitives the ordering has already deprioritised, producing the smooth quality–budget curves and enabling its graceful low-budget LoD.

4.3 Ablation Studies

Unless otherwise noted, we ablate on a single scene (bicycle, MipNeRF 360 [1]) with $N=1\text{M}$ splats and 50k steps, varying one design choice at a time while keeping the others at their defaults (Tab. 3).

Importance Score. We evaluate seven importance scores from four per-splat criteria $s(g)$ (opacity, volume, SH energy, colour variance; Eq. (4)) in ascending and descending order, plus orders not determined by splat characteristics (Fig. 7). Sorting by opacity in descending order consistently produces the most effective multi-budget performance among all strategies: at 10% of the splat budget, it achieves 22.2 dB PSNR at 493 FPS, whereas the next-best score-based ordering (SH-energy descending) reaches only 17.6 dB under the same constraints. Ascending orders perform poorly throughout.

We also compare against fixed insertion orders that bypass importance-based sorting: the **append**, **prepend**, and **fixed-random** baselines insert newly densified Gaussian at the end, the start, or a fixed random index, respectively, and never re-sort, isolating the effect of opacity-descending resort. These fixed orders underperform opacity-descending at low prefix ratios, confirming the importance of a semantically meaningful coarse-to-fine ordering for preserving early-prefix quality. Full-budget **append/prepend** metrics are in Tab. 3; **fixed-random** and full-capacity ($N=5\text{M}$) results are in supplementary.

We deliberately use opacity rather than a dedicated learnable importance scalar. Opacity is the key per-primitive scalar that already receives a coherent

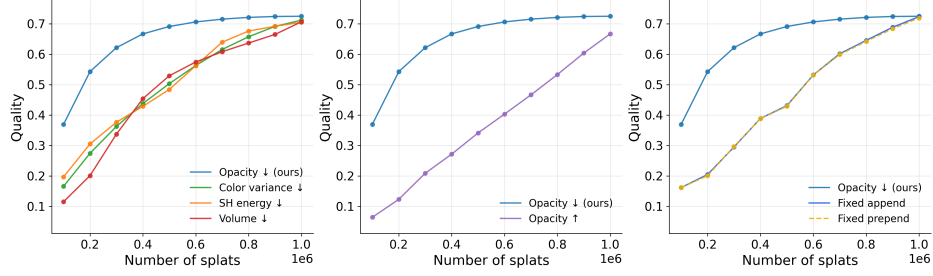


Fig. 7: Ablations on the importance score. The opacity-descending dominates other importance scores across the full budget range.

Table 3: Ablation study on importance score and budget training strategy. All metrics reported at split=1.0 (full budget). PSNR, SSIM and LPIPS represent raw quality; AUC_{fps} and AUC_{splats} summarise quality–efficiency frontiers. The **best** and **second best** results per column are highlighted. $\uparrow$ higher is better, $\downarrow$ lower is better.

		PSNR $\uparrow$	SSIM $\uparrow$	LPIPS $\downarrow$	$AUC_{fps}\uparrow$	$AUC_{splats}\uparrow$
Importance Score	Opacity $\downarrow$ (ours)	<u>25.47</u>	<u>0.776</u>	<u>0.174</u>	<u>68.62</u>	<u>61.05</u>
	Opacity $\uparrow$	24.91	0.743	0.219	30.53	33.48
	Colour variance $\downarrow$	25.25	0.770	0.183	51.94	46.26
	SH energy $\downarrow$	25.17	0.766	0.187	<u>59.39</u>	<u>47.12</u>
	Volume $\downarrow$	25.19	0.764	0.183	52.15	44.73
	Fixed append	25.51	<u>0.773</u>	<u>0.175</u>	46.41	43.09
	Fixed prepend	25.32	0.772	0.176	49.65	42.90
Budget Training	Prefix + full (ours)	25.43	0.776	0.175	66.87	60.96
	Prefix	24.97	0.734	0.253	56.58	57.09
	MRL	<u>25.37</u>	<u>0.771</u>	<u>0.183</u>	<u>66.57</u>	<u>60.35</u>

per-pixel gradient under every prefix via α -compositing, so it ranks primitives at no extra cost, whereas a separate importance head would require a synthetic gradient and forfeit the drop-in nature of MGS. Consistent with this observation, all non-opacity criteria trail opacity-descending; view-dependent importance is discussed under the view-adaptive extension of Sec. 5.

Budget Training. Fixing the opacity-descending score, we compare three budget-training strategies: **prefix + full** (default; Eq. (7)), **prefix only** (no full-set term), and **MRL nesting** [20] (deterministic, fixed prefix sizes). **Prefix + full** gives the best low-budget/full-quality trade-off; **prefix only** marginally improves early prefixes but degrades higher budgets; and **MRL nesting** restricts the model to discrete budget sizes, sacrificing the fine-grained coverage of stochastic sampling.

Prefix/Full Loss Weight. Sweeping the prefix : full weight ratio from 9:1 to 1:9 reveals a clear trade-off (Fig. 8): at the 10% prefix, 9:1 attains 22.2 dB whereas

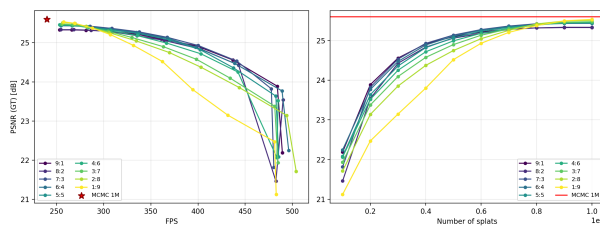


Fig. 8: Ablations on Prefix/Full Loss Weight distribution. While prefix-heavy or full-heavy weighting can marginally improve their respective endpoints, a 6:4 ratio provides the best area under the curve. For the sake of simplicity and balanced performance, we utilise a 5:5 ratio for our MGS approach and for all other ablation experiments.

1:9 drops to 21.1dB, while at full quality 1:9 is best (PSNR 25.5dB, LPIPS 0.169), surpassing 9:1 by +0.2dB. We use **1:1** (5:5) by default, as it performs comparably (25.4dB at full quality, 22.1 dB at the 10% prefix).

5 Conclusion

We present Matryoshka Gaussian Splatting (MGS), a continuous budget-control framework for 3DGS. By learning an ordered, prefix-closed set of Gaussian primitives, MGS renders at arbitrary splat budgets by truncating the set, yielding a dense spectrum of quality–speed operating points; a stochastic budget-training strategy lets a single representation perform well at every prefix length and integrates into existing 3DGS pipelines without architectural changes. Experiments on four standard benchmarks show that MGS matches the full-capacity quality of 3DGS while enabling continuous LoD from a single model, and ablations confirm the proposed ordering, objective, and budget-sampling designs. These results suggest that nested primitive representations are a promising direction for scalable neural scene representations. While our global ordering is deliberately view-agnostic, our preliminary experiments suggest it is a strong base for view- and spatially-adaptive extensions. Adaptive budget scheduling and device-aware rendering are also promising further directions.

Acknowledgements

This work was supported by a UKRI Future Leaders Fellowship [grant number G127262].

References

1. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: CVPR. pp. 5470–5479 (2022)
2. Cai, H., Gan, C., Wang, T., Zhang, Z., Han, S.: Once-for-all: Train one network and specialize it for efficient deployment. In: ICLR (2020)
3. Chen, Y., Li, M., Wu, Q., Lin, W., Harandi, M., Cai, J.: Pcgs: Progressive compression of 3d gaussian splatting. In: AAAI (2026)
4. Cheng, Z., Sun, M., Liu, Y., Ge, Z., Tang, L., Xu, M., Li, Y., Pan, P.: Clod-gs: Continuous level-of-detail via 3d gaussian splatting. In: ICLR (2026)
5. Clark, J.H.: Hierarchical geometric models for visible surface algorithms. *Communications of the ACM* **19**(10), 547–554 (1976)
6. Fan, Z., Wang, K., Wen, K., Zhu, Z., Xu, D., Wang, Z.: Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. In: NeurIPS. vol. 37, pp. 140138–140158 (2024)
7. Fang, G., Wang, B.: Mini-splatting: Representing scenes with a constrained number of gaussians. In: ECCV. pp. 165–181. Springer (2024)
8. Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. In: CVPR. pp. 5501–5510 (2022)
9. Funkhouser, T.A., Séquin, C.H.: Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In: Proceedings of the 20th annual conference on Computer graphics and interactive techniques. pp. 247–254 (1993)
10. Giegl, M., Wimmer, M.: Unpopping: Solving the image-space blend problem for smooth discrete lod transitions. *Computer Graphics Forum* **26**(1), 46–49 (2007)
11. Hanson, A., Tu, A., Singla, V., Jayawardhana, M., Zwicker, M., Goldstein, T.: Pup 3d-gs: Principled uncertainty pruning for 3d gaussian splatting. In: CVPR. pp. 5949–5958 (2025)
12. Hedman, P., Philip, J., Price, T., Frahm, J.M., Drettakis, G., Brostow, G.: Deep blending for free-viewpoint image-based rendering. *TOG* **37**(6), 1–15 (2018)
13. Hoppe, H.: Progressive meshes. In: SIGGRAPH. pp. 99–108 (1996)
14. Hoppe, H.: View-dependent refinement of progressive meshes. In: SIGGRAPH. pp. 189–198 (1997)
15. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. *TOG* **42**(4), 139–1 (2023)
16. Kerbl, B., Meuleman, A., Kopanas, G., Wimmer, M., Lanvin, A., Drettakis, G.: A hierarchical 3d gaussian representation for real-time rendering of very large datasets. *TOG* **43**(4), 1–15 (2024)
17. Kheradmand, S., Rebain, D., Sharma, G., Sun, W., Tseng, Y.C., Isack, H., Kar, A., Tagliasacchi, A., Yi, K.M.: 3d gaussian splatting as markov chain monte carlo. In: NeurIPS. vol. 37, pp. 80965–80986 (2024)
18. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. *TOG* **36**(4), 1–13 (2017)
19. Kulhanek, J., Rakotosaona, M.J., Manhardt, F., Tsalicoglou, C., Niemeyer, M., Sattler, T., Peng, S., Tombari, F.: Lodge: Level-of-detail large-scale gaussian splatting with efficient rendering. In: NeurIPS (2025)
20. Kusupati, A., Bhatt, G., Rege, A., Wallingford, M., Sinha, A., Ramanujan, V., Howard-Snyder, W., Chen, K., Kakade, S., Jain, P., et al.: Matryoshka representation learning. In: NeurIPS. vol. 35, pp. 30233–30249 (2022)

21. Lee, Y., Zhang, Z., Fan, D.: Safeguards: 3d gaussian primitive pruning while avoiding catastrophic scene destruction. In: WACV. pp. 8479–8489 (2026)
22. Liu, Y., Zhong, Z., Zhan, Y., Xu, S., Sun, X.: Maskgaussian: Adaptive 3d gaussian representation from probabilistic masks. In: CVPR. pp. 681–690 (2025)
23. Lu, T., Yu, M., Xu, L., Xiangli, Y., Wang, L., Lin, D., Dai, B.: Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In: CVPR. pp. 20654–20664 (2024)
24. Luebke, D., Reddy, M., Cohen, J.D., Varshney, A., Watson, B., Huebner, R.: Level of detail for 3D graphics. Elsevier (2002)
25. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: ECCV. pp. 405–421 (2020)
26. Milef, N., Seyb, D., Keeler, T., Nguyen-Phuoc, T., Božič, A., Kondguli, S., Marshall, C.: Learning fast 3d gaussian splatting rendering using continuous level of detail. *Computer Graphics Forum* **44**(2), e70069 (2025)
27. Navaneet, K., Pourahmadi Meibodi, K., Abbasi Koohpayegani, S., Pirsivash, H.: Compigs: Smaller and faster gaussian splatting with vector quantization. In: ECCV. pp. 330–349. Springer (2024)
28. Neff, T., Stadlbauer, P., Parger, M., Kurz, A., Mueller, J.H., Chaitanya, C.R.A., Kaplanyan, A., Steinberger, M.: Donerf: Towards real-time rendering of compact neural radiance fields using depth oracle networks. *Computer Graphics Forum* **40**(4), 45–59 (2021)
29. Papantonakis, P., Kopanas, G., Kerbl, B., Lanvin, A., Drettakis, G.: Reducing the memory footprint of 3d gaussian splatting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* **7**(1), 1–17 (2024)
30. Reiser, C., Peng, S., Liao, Y., Geiger, A.: Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In: ICCV. pp. 14315–14325 (2021)
31. Ren, K., Jiang, L., Lu, T., Yu, M., Xu, L., Ni, Z., Dai, B.: Octree-gs: Towards consistent real-time rendering with lod-structured 3d gaussians. *TPAMI* pp. 1–15 (2025)
32. Rippel, O., Gelbart, M., Adams, R.: Learning ordered representations with nested dropout. In: ICML. pp. 1746–1754. PMLR (2014)
33. Seo, Y., Choi, Y.S., Son, H.S., Uh, Y.: Flod: Integrating flexible level of detail into 3d gaussian splatting for customizable rendering. In: SIGGRAPH. pp. 1–16 (2025)
34. Shen, J., Qian, Y., Zhan, X.: Lod-gs: Achieving levels of detail using scalable gaussian soup. In: CVPR. pp. 671–680 (2025)
35. Shi, Y., Morin, G., Gasparini, S., Ooi, W.T.: Lapisgs: Layered progressive 3d gaussian splatting for adaptive streaming. In: 3DV. pp. 991–1000. IEEE (2025)
36. Tian, B., Gao, Q., Xianyu, S., Cui, X., Zhang, M.: Flexgaussian: Flexible and cost-effective training-free compression for 3d gaussian splatting. In: ACM MM. pp. 7287–7296 (2025)
37. Wang, L., Shi, X., Liu, Y.: Foveated rendering: A state-of-the-art survey. *Computational Visual Media* **9**(2), 195–228 (2023)
38. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. *TIP* **13**(4), 600–612 (2004)
39. Xiangli, Y., Xu, L., Pan, X., Zhao, N., Rao, A., Theobalt, C., Dai, B., Lin, D.: Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering. In: ECCV. pp. 106–122. Springer (2022)
40. Yan, Z., Low, W.F., Chen, Y., Lee, G.H.: Multi-scale 3d gaussian splatting for anti-aliased rendering. In: CVPR. pp. 20923–20931 (2024)

- 41. Ye, V., Li, R., Kerr, J., Turkulainen, M., Yi, B., Pan, Z., Seiskari, O., Ye, J., Hu, J., Tancik, M., et al.: gsplat: An open-source library for gaussian splatting. *Journal of Machine Learning Research* **26**(34), 1–17 (2025)
- 42. Yu, J., Yang, L., Xu, N., Yang, J., Huang, T.: Slimmable neural networks. In: *ICLR* (2019)
- 43. Yu, Z., Chen, A., Huang, B., Sattler, T., Geiger, A.: Mip-splatting: Alias-free 3d gaussian splatting. In: *CVPR*. pp. 19447–19456 (2024)
- 44. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: *CVPR*. pp. 586–595 (2018)
- 45. Zhang, Y., Ortín-Obón, M., Arellano, V., Wang, R., Gutierrez, D., Bao, H.: On-the-fly power-aware rendering. *Computer Graphics Forum* **37** (2018)