NECROMANTIC GAUSSIAN: RAISING THE RETENTION CONFIDENCE FOR CONTINUOUS LEVEL OF DETAIL IN 3D GAUSSIAN SPLATTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Level of detail (LoD) for 3D Gaussian Splatting trades rendering cost against fidelity, but reducing a trained field requires deciding which primitives to retain. Heuristics based on size, distance, or opacity can discard essential coverage, while hard selection removes the reconstruction feedback that could reverse a bad decision. We introduce Necromantic Gaussian, which learns retention confidence from the rendering loss itself. Built on Octree-GS, a lightweight router predicts per-anchor confidence from base and dedicated side features. Rendering supervision teaches which anchors are kept, while a direct count objective calibrates how many decoded Gaussians are selected at a requested budget. An extended rasterizer backward pass evaluates local reconstruction sensitivity at zero opacity, allowing rejected gates to receive image evidence without contributing to the hard forward image. Scale compensation and joint optimization of the base representation help the retained subset preserve coverage. The resulting confidence ordering lets a single checkpoint serve budgets specified at inference. We evaluate its quality–workload trade-offs on Tanks&Temples and Mip-NeRF 360, including controlled mechanism ablations.

1 INTRODUCTION

3D Gaussian Splatting (3DGS) represents scenes with explicit anisotropic primitives and differentiable rendering Kerbl et al. (2023). While densification improves quality, it increases rendering cost. Level of detail (LoD) addresses this gap by adapting a single scene representation to different rendering budgets without retraining.

The central challenge is deciding which Gaussians to retain. Unlike meshes, Gaussian fields lack an explicit simplification order: overlapping primitives jointly explain coverage, boundaries, and appearance. Therefore, heuristics based on size, distance, or opacity may discard essential primitives, making retention ordering as important as the budget itself.



Figure 1: One checkpoint rendered at progressively smaller budgets on Mip-NeRF 360 room.

Existing methods expose reduced capacity through hierarchies, pruning, masks, continuous scale control, or nested subsets (Ren et al., 2024; Kerbl et al., 2024; Fan et al., 2024; Liu et al., 2025; **?**; Guo et al., 2026), but the retained primitives are typically determined by predefined structures or importance rules. This leaves two coupled questions unresolved: **which** primitives should occupy a given budget and **how many** primitives the budget should permit. A budget alone specifies only the amount of retained work, while a fixed importance rule does not guarantee that the retained subset preserves the image-space coverage needed for reconstruction. Meanwhile hard selection introduces an additional optimization difficulty: once a primitive is excluded from rasterization, the standard backward pass no longer provides the reconstruction evidence needed to reconsider that decision.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We introduce **Necromantic Gaussian**, a continuous LoD framework that learns retention confidence directly from rendering supervision. Built on the anchor-structured representation of Octree-GS, a lightweight network predicts anchor-level confidence from base and learned side features. Reconstruction gradients determine which anchors should retain confidence, while a direct decoded-Gaussian count objective provides global pressure toward how many Gaussians the requested workload permits. To keep hard routing recoverable, we extend the rasterizer backward pass to evaluate rejected Gaussians at zero opacity, allowing inactive gates to receive signed image-space evidence without affecting the hard forward image. The resulting confidence field defines a shared retention ordering that can be queried at different workloads from a single checkpoint.

At inference, a continuous budget selects a workload-compatible prefix of the confidence-ordered anchors. Because the ordering is learned rather than fixed, one checkpoint supports the full range of LoD subsets without retraining. We evaluate this quality–workload trade-off on Tanks and Temples and Mip-NeRF 360.

Our contributions are:

1. We formulate Gaussian LoD as confidence-guided workload allocation. A lightweight anchor-level predictor learns a rendering-driven retention ordering, decoupling primitive selection from workload size without hand-crafted scores.
2. We make hard Gaussian selection recoverable by extending rasterizer backward to evaluate the local alpha-compositing derivative of rejected Gaussians at zero opacity. Combined with a straight-through gate, this provides signed reconstruction evidence to inactive anchors without changing the hard forward image.
3. We realize single-checkpoint continuous budget control by combining reconstruction-driven confidence learning with direct decoded-Gaussian count supervision. The former determines which anchors should occupy the available budget, while the latter calibrates the confidence population to the requested workload.

2 Related Work

Gaussian representations and compression. Neural radiance fields model scenes as continuous functions (Mildenhall et al., 2021; Barron et al., 2021), whereas 3D Gaussian Splatting (3DGS) uses explicit anisotropic primitives and a differentiable rasterizer whose work scales with submitted Gaussians (Kerbl et al., 2023). Mip-Splatting improves scale consistency through anti-aliasing (Yu et al., 2024), and Scaffold-GS and Octree-GS decode Gaussians from sparse anchors or voxel structures (Lu et al., 2024; Ren et al., 2024). Compression methods such as LightGaussian and Compact3DGS learn pruning or masks, while Mini-Splatting and budgeted densification constrain capacity during optimization (Fan et al., 2024; Lee et al., 2024; Fang & Wang, 2024). These methods reduce stored capacity, usually at a fixed operating point or over a finite set of points.

Level of detail. Runtime LoD selects a quality–workload operating point, classically by cutting a progressive hierarchy (Hoppe, 2023; 1997). Gaussian methods build related hierarchies or nested subsets: Octree-GS uses voxel scales, LODGE uses camera distance, GoDe freezes a sensitivity ordering, and FLoD trains discrete levels (Ren et al., 2024; Kulhanek et al., 2025; Di Sario et al., 2025; Seo et al., 2024). CLoD-GS learns distance-dependent opacity decay, while Matryoshka Gaussian Splatting learns nested capacities (**?**Guo et al., 2026). Their controls are set by distance, hierarchy, or global capacity and do not directly supervise the decoded Gaussian count. We learn the ordering from rendering feedback and calibrate that count for each requested workload.

Learned selection and sensitivity. Pruning methods use complexity-constrained gates and straight-through estimators (Louizos et al., 2017; Bengio et al., 2013). MaskGaussian keeps probabilistic masks differentiable through rasterization, RAP predicts importance from intrinsic statistics, and LeGS uses rendering sensitivity as a densification reward (Liu et al., 2025; Yang et al., 2026; Ning et al., 2026). We instead combine rasterizer feedback with a decoded-count objective for continuous per-view LoD, and evaluate the zero-opacity alpha-compositing derivative so excluded primitives remain recoverable. The resulting confidence ordering supports multiple workloads from one checkpoint without conflating selection with the amount of rendering work.

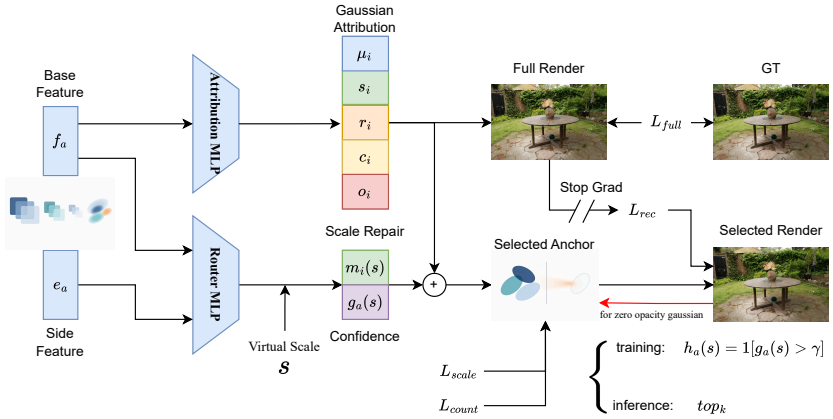

Figure 2: Architecture and training paths of Necromantic Gaussian. Multi-resolution anchors are decoded into Gaussian slots, routed by a shared confidence network, and trained with reconstruction and workload objectives before target-budget prefix export.

3 METHOD

3.1 PRELIMINARIES AND OVERVIEW

We use Scaffold-GS anchors (Lu et al., 2024) with Octree-GS's multi-resolution voxel sizes (Ren et al., 2024). Routing decisions are made per anchor and broadcast to its decoded slots, while rendering workload is counted after expansion. A virtual scale $s \geq 1$ requests fraction $q = 1/s$; $s = 1$ uses all candidates.

We turn LoD selection into a learned confidence ordering over multi-scale anchors. Section 3.2 learns which anchors matter by connecting confidence to rendering loss, and section 3.3 restores image gradients to candidates rejected by the hard gate. Section 3.4 calibrates how many decoded Gaussians the budget permits, while section 3.5 repairs lost coverage and jointly adapts the base representation. Finally, section 3.6 converts the learned ordering into a workload-aware prefix, compacts the retained slots, and renders multiple LoD levels from one checkpoint. The pipeline is shown in fig. 2.

3.2 LEARNING CONFIDENCE THROUGH OPACITY

We learn an LoD ordering from rendering error rather than geometric heuristics. Because overlapping Gaussians jointly explain the image, a primitive's size or opacity is an unreliable importance proxy. A learned anchor confidence lets the reconstruction loss itself determine retention.

We let confidence act through opacity, so routing is optimized in the same image-space compositing process that determines reconstruction quality. Multiplying a slot's base opacity α_i^0 by confidence allows rendering gradients to distinguish anchors whose attenuation increases reconstruction error from those that can be suppressed with little visual consequence. The count objective at section 3.4 provides a global pressure on the confidence population toward the requested workload, but does not determine which individual anchor should be retained. Thus, reconstruction supervision answers which anchors should occupy the budget, whereas count supervision controls how many decoded Gaussians the budget should admit.

We predict confidence from the anchor feature f_a and a learned side feature e_a. The base feature describes scene content and is shared with the scene decoder. A dedicated learned side feature e_a gives the LoD objective its own per-anchor representation, decoupling the retention decision from the reconstruction task and preventing LoD gradients from overly distorting scene features. A shared MLP maps the concatenation $[f_a, e_a]$ to a scalar logit z_a and per-slot scale-compensation parameters $t_{a,k}$:

$$(z_a, \{t_{a,k}\}_k) = F_\theta([f_a, e_a]), \qquad g_a(s) = \sigma(\kappa - \log_2(s)\ \mathrm{softplus}(\delta_0 + z_a))\,,$$
$$h_a(s) = \mathbf{1}[g_a(s) > \gamma], \qquad s > 1. \tag{1}$$

Here g_a is an anchor-level soft confidence broadcast to all its decoded slots, and the hard gate h_a binarizes it for the forward pass. δ_0 controls the shared decay trend while z_a provides anchor-specific adaptation. The monotonic confidence function ensures that different budgets correspond to cuts over the same learned ordering.

After densification, training first uses soft routing to establish confidence ordering and then enables hard selection. Count supervision is gradually introduced during this transition.

3.3 Gradient Recovery for Rejected Gaussians

Once a hard gate rejects a Gaussian, boolean compaction removes both its image contribution and gradient. A backward extension is therefore needed to let an inactive gate recover.

During training, a rejected slot at pixel p has $\alpha(p) = 0$, so it leaves the forward rendered color $C(p)$ unchanged. However, its local alpha-compositing derivative is nonzero:

$$C(p) = \sum_i T_i(p)\alpha_i(p)c_i + T_{\text{end}}(p)c_{\text{bg}}, \qquad T_i(p) = \prod_{j<i}(1-\alpha_j(p)),$$
$$\left.\frac{\partial C(p)}{\partial \alpha_i(p)}\right|_{\alpha_i(p)=0} = T_i(p)\big(c_i - C_{\text{behind},i}(p)\big). \tag{2}$$

Combined with image gradients, this derivative provides signed reconstruction feedback for rejected candidates. We connect the binary decision to soft confidence directly in the opacity:

$$\bar{\alpha}_i^{\text{ST}} = \alpha_i^0 g_{a(i)} \left[h_{a(i)} + g_{a(i)} - \text{stopgrad}(g_{a(i)})\right]. \tag{3}$$

In the forward pass, $h_{a(i)}$ is binary: selected anchors contribute $\alpha_i^0 g_{a(i)}$, while rejected candidates remain in the training rasterizer input with zero opacity and therefore do not change the rendered image. In the backward pass, the straight-through $g_{a(i)}$ term carries confidence gradients, and the analytic zero-opacity derivative supplies reconstruction feedback to the rejected candidates. They are compacted out at inference.

3.4 Workload Correction

Rendering gradients determine which anchors matter, but do not specify how many Gaussians the current budget permits. Moreover, equal numbers of anchors can decode unequal workloads. We therefore supervise the selected Gaussian fraction after expansion. Let $K_a(v)$ be the number of valid slots decoded from anchor a under camera v, and let $N_{\text{cand}}(v) = \sum_a K_a(v)$. We define

$$\eta_h = \frac{\sum_a K_a(v) h_a}{N_{\text{cand}}(v)}, \qquad \eta_g = \frac{\sum_a K_a(v) g_a}{N_{\text{cand}}(v)},$$
$$\eta_{\text{ST}} = \eta_h + \eta_g - \text{stopgrad}(\eta_g), \qquad L_{\text{count}} = (s-1)^2 \ell_\epsilon(\eta_{\text{ST}} - 1/s). \tag{4}$$

Here η_h is the fraction of decoded slots selected by the hard gate, η_g is the corresponding confidence-weighted soft fraction, and η_{ST} is their straight-through estimator. After hard routing is enabled, η_h is the workload fraction used in the forward pass, while η_g supplies its differentiable correction. Unlike the reconstruction loss, L_{count} does not pass through the rasterizer and therefore contains no information about which individual anchor is visually preferable. Instead, it shifts the confidence distribution toward the requested decoded workload. Rendering gradients then determine which anchors resist this global suppression pressure.

The reconstruction straight-through estimator (STE) acts per slot: its forward value follows the hard gate, while its backward path uses the soft confidence to carry gradients. In contrast, η_h and η_g aggregate decoded slots for workload calibration. Before the route is activated, this branch is absent. During the soft warmup, the hard mask is disabled and every slot remains submitted, so the count loss uses η_g to drive the soft activation rate toward $1/s$. Once the warmup ends, the forward count uses η_h, while $\eta_g - \text{stopgrad}(\eta_g)$ supplies its gradient.

Importantly, this count supervision is a training-time calibration objective; the target-budget prefix used at deployment in Section 3.6 is a separate mechanism that converts the learned confidence ordering into an explicit workload constraint.

3.5 Coverage Compensation and Joint Learning

Opacity gating can weaken retained spatial support. We therefore let each retained slot enlarge its scale as workload decreases:

$$m_i(s) = \exp(\log(s)\,\sigma(t_i))\,, \qquad \widetilde{\boldsymbol{\ell}}_i = \boldsymbol{\ell}_i^0 \odot m_i(s), \\ \bar{\alpha}_i = h_{a(i)}(s) g_{a(i)}(s) \alpha_i^0. \tag{5}$$

Scale compensation enlarges retained footprints within a bounded range to recover coverage lost from selection. A full branch reconstructs the ground-truth image with all candidates. The route branch reconstructs the detached full-branch image using the selected and compensated representation. Their objective is

$$L = L_{\text{full}} + w(s)\left[L_{\text{rec}} + \beta(t) L_{\text{count}} + \lambda_{\text{scale}} L_{\text{scale}}\right], \\ L_{\text{scale}} = \text{mean}_{i,d}\big(\log m_{i,d}\big)^2. \tag{6}$$

The regularizer prevents compensation from becoming a shortcut. Only the full-branch teacher is detached; decoded attributes and anchor features remain differentiable, so image gradients jointly shape confidence and the representation. Hard selection starts after the soft transition forms the ordering.

3.6 Inference

Given a requested fraction q, we set $s = 1/q$ and evaluate the learned confidence over the anchors in the current visible candidate pool. We then rank these candidates by confidence and retain the shortest prefix whose decoded Gaussian workload reaches the target. The selected slots are compacted and rasterized. The prefix does not learn a new ordering at inference: it only converts the confidence ordering learned under reconstruction and count supervision into an explicit workload constraint.

4 Experiments

4.1 Experimental Setup

We use nine Mip-NeRF 360 scenes at quarter resolution and the train and truck scenes of Tanks&Temples at full resolution (Barron et al., 2022; Knapitsch et al., 2017). We compare 3DGS, Octree-GS, CLoD-GS, Matryoshka-GS, FlexGaussian, and MaskGaussian. We report PSNR, SSIM, VGG-LPIPS, and the mean per-view effective Gaussian count, with equal weight for every scene. We train Ours and 3DGS for 30k iterations.

4.2 Implementation Details

The router is a lightweight MLP operating on base and side anchor features. Training follows a 15k densification phase and a 5k soft routing transition before hard selection. Octree-GS and Ours are trained for 30k to match the standard 3DGS training budget. After the 15k-iteration densification phase, training applies a soft route for 5k iterations with a ramped count loss before enabling the hard gate; the target fraction is $q = 1/s$. We sample $s \in [1, 5]$ and report Ours at $s = 1, 2, 4$, corresponding to 100%, 50%, and 25% target workloads.

The router and compensation heads use Adam with learning rate 5×10^{-3}, exponentially decayed to 5×10^{-4} over the route schedule. The factor group uses the Octree-GS optimizer with learning rate 10^{-2}, decayed to 10^{-3} over the 15k route steps. Reconstruction and count losses have unit weight; scale compensation uses a 2×10^{-3} log-scale regularizer and the count Huber transition is 0.05. The full branch is stepped before the route branch; further ordering details are given in section A.2.

4.3 Comparison

The comparison distinguishes the scope of selection. Octree-GS, CLoD-GS, and Ours determine the submitted representation from the candidates available to each view, whereas Matryoshka-GS, FlexGaussian, and MaskGaussian use global/static retained subsets. We denote the former setting

Table 1: Quality at native LoD operating points. Budget percentages are method-specific and do not imply matched workloads across methods. N denotes the mean per-view number of Gaussians with positive rasterizer radii, measured consistently for all methods. We distinguish view-dependent selection (V), global/static subsets (G), and full/native references (F). Cross-method quality–workload comparisons should be read from Fig. 3 on the common N axis. Values are averaged equally across scenes; $\pm$ denotes the population standard deviation of scene-level means.

		Tanks&Temples				Mip-NeRF 360			
Budget	Method	PSNR	SSIM	LPIPS	N (k)	PSNR	SSIM	LPIPS	N (k)
Full/native	3DGS^{F}	23.754	0.8530	0.2036	713.1 ± 38.0	28.111	0.8341	0.1993	724.2 ± 345.6
	Octree-GS^{V}	24.189	0.8589	0.1927	450.7 ± 81.5	28.019	0.8305	0.2055	419.6 ± 229.4
	CLoD-GS^{V}	23.427	0.8405	0.2218	524.4 ± 54.1	27.436	0.8233	0.2157	563.5 ± 272.5
	Matryoshka-GS^{G}	24.645	0.8788	0.1472	2883.8 ± 147.3	28.475	0.8489	0.1635	2805.6 ± 631.5
	FlexGaussian^{G}	23.754	0.8530	0.2036	713.1 ± 38.0	28.111	0.8341	0.1993	724.2 ± 345.6
	MaskGaussian^{G}	23.787	0.8483	0.2179	287.3 ± 45.3	27.987	0.8331	0.2035	390.6 ± 206.8
	Ours^{V}	23.971	0.8563	0.1937	483.3 ± 85.6	27.709	0.8248	0.2107	437.7 ± 230.3
50%	CLoD-GS^{V}	21.366	0.8133	0.2492	262.2 ± 27.0	22.140	0.7397	0.2851	281.8 ± 136.2
	Matryoshka-GS^{G}	24.573	0.8752	0.1551	1410.4 ± 74.1	28.319	0.8434	0.1730	1316.7 ± 318.7
	FlexGaussian^{G}	22.792	0.8233	0.2339	414.5 ± 63.8	26.238	0.7983	0.2337	457.8 ± 207.0
	Ours^{V}	23.884	0.8499	0.2159	243.2 ± 43.0	27.404	0.8146	0.2402	219.1 ± 115.0
33.3%	Ours^{V}	23.762	0.8410	0.2370	162.4 ± 28.6	27.103	0.8011	0.2679	146.2 ± 76.7
25%	CLoD-GS^{V}	15.947	0.6963	0.3401	131.1 ± 13.5	16.218	0.5644	0.4078	140.9 ± 68.1
	Matryoshka-GS^{G}	24.102	0.8559	0.1965	666.3 ± 49.9	27.719	0.8167	0.2128	593.9 ± 179.8
	FlexGaussian^{G}	20.120	0.7378	0.3077	225.5 ± 47.4	22.083	0.6935	0.3162	248.2 ± 116.2
	Ours^{V}	23.599	0.8280	0.2612	121.9 ± 21.5	26.697	0.7789	0.3000	109.7 ± 57.5
20%	Ours^{V}	23.164	0.8035	0.2941	97.5 ± 17.2	25.615	0.7400	0.3392	87.7 ± 46.0

by V and the latter by G; for Ours, V refers to per-view selection over the visible candidate pool rather than an explicitly camera-conditioned confidence predictor. All methods are compared on the same reported per-view effective Gaussian-count axis.

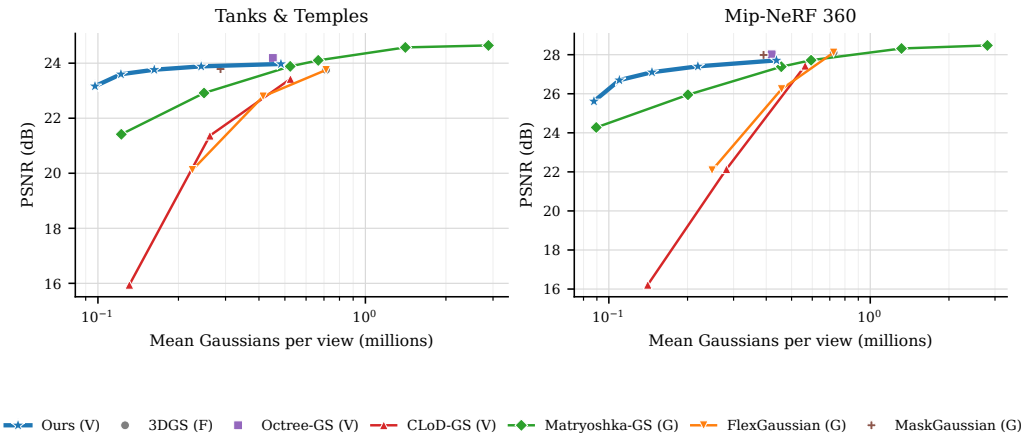

Figure 3: Quality as a function of the reported mean effective Gaussian count.

table 1 groups methods by LoD ratio; equal labels are method-specific operating settings rather than matched workloads. The same ratio therefore does not mean the same rendering budget. For example, at the nominal 50% point on Mip-NeRF 360, Matryoshka-GS submits 1316.7k effective Gaussians versus 219.1k for Ours, and at 25% it submits 593.9k versus 109.7k. Its higher PSNR at those rows is consequently obtained with substantially more primitives, whereas Ours is designed to preserve quality at a given workload. We therefore treat the ratio rows as native operating points and use the common-N horizontal axis in fig. 3 for the fair comparison: in the reduced-workload region, Ours provides the stronger quality–workload trade-off against the reduction baselines. For CLoD-GS, we bisect its distance-decay control for each view until the positive-radii count reaches 50% or 25% of its full count. At 50%, Ours uses fewer effective Gaussians and exceeds CLoD-GS by 2.52 dB on Tanks&Temples and 5.26 dB on Mip-NeRF 360. At 25%, the gaps increase to 7.65

and 10.48 dB, respectively. fig. 3 shows the corresponding gradual quality–workload trade-off. The corresponding visual comparisons are shown in fig. 4.

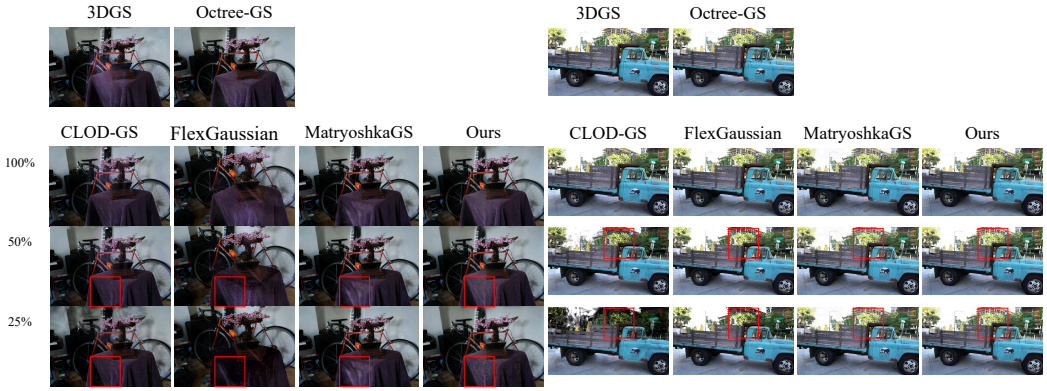

Figure 4: Qualitative comparison on Tanks&Temples (left) and Mip-NeRF 360 (right) at the 50% and 25% operating points in table 1.

4.4 Ablation Study

4.4.1 Confidence Learning and Workload Calibration

Table 2: Ablations of the routing design at two LoD budgets. Entries report PSNR changes relative to the full model under target-budget prefix evaluation. Training top-k replaces confidence-calibrated routing with direct top-k allocation during training, and is therefore an alternative training strategy rather than a component-wise ablation.

Variant	Mip-NeRF 360		Tanks&Temples	
	50%	25%	50%	25%
Full model	0.000	0.000	0.000	0.000
Training top-k	-1.558	-2.474	-0.803	-2.322
No analytic backward	-0.055	-0.101	+0.055	+0.044
No warmup	-0.182	+0.022	-0.055	+0.014
Both stabilizers removed	-0.374	-0.120	-0.192	-0.106
No scale compensation	-0.066	+0.095	-0.036	+0.028

We first evaluate whether the learned confidence field correctly couples retention preference with workload control. Training top-k replaces the confidence-based hard routing and count calibration with direct selection during training. It causes the largest degradation among all variants, decreasing PSNR by 1.6–2.5 dB on Mip-NeRF 360 and 0.8–2.3 dB on Tanks&Temples at 50%/25% budgets. This result is consistent with our earlier observation that naive hard selection removes reconstruction feedback from rejected candidates and prevents the model from learning a recoverable retention ordering.

The count objective provides a complementary signal to reconstruction supervision. Reconstruction gradients determine which anchors should survive, while count supervision controls the population of retained decoded Gaussians.

4.4.2 Recoverable Hard Routing

We next study whether hard routing remains reversible after a Gaussian is temporarily rejected. Removing the analytic zero-opacity backward changes the final PSNR by less than 0.10 dB, because reconstruction and workload objectives still provide sufficient supervision before candidates become inactive.

However, the purpose of analytic backward is not to directly increase the final metric. Instead, it prevents a rejected candidate from becoming permanently unreachable due to the loss of rasterizer gradients. The controlled toy experiment in section A.1 isolates this behavior: standard hard masking produces zero gradient for the rejected useful Gaussian and never recovers, while the proposed zero-opacity derivative provides signed feedback and restores the correct selection without changing the hard forward image.

Warmup and analytic backward address different stages of the same failure mode. The former reduces poor decisions during the initial transition to discrete routing, while the latter preserves local recoverability after rejection. Removing both produces a larger degradation than removing either component alone, confirming their complementary stabilization effects.

4.4.3 Coverage Adaptation and Budget Export

Table 3: Native threshold inference compared with target-budget top-k prefix export. The first two rows report threshold-minus-top-k PSNR; the last two report the realized fraction of the target-budget candidate pool.

Dataset	Quantity	$s = 2$	$s = 3$	$s = 4$	$s = 5$
Mip-NeRF 360	ΔPSNR (dB)	-0.031	-0.008	$+0.043$	-0.919
Tanks&Temples	ΔPSNR (dB)	-0.014	$+0.009$	-0.001	$+0.030$
Mip-NeRF 360	Threshold fraction	0.483	0.331	0.256	0.174
Tanks&Temples	Threshold fraction	0.482	0.338	0.250	0.201

We evaluate the mechanisms used after confidence ordering is learned. Removing scale compensation causes only minor reconstruction changes, confirming that coverage adaptation serves as a secondary refinement rather than the source of the learned retention ordering.

The learned confidence field can be queried either through native thresholding or through target-budget prefix export. As shown in table 3, threshold inference closely follows the requested workload over the stable range of $s = 2$–4 while maintaining comparable reconstruction quality. This demonstrates that count supervision gives the confidence distribution a meaningful workload interpretation, while prefix export provides an explicit deployment-time budget constraint.

4.5 Large-Scale Scene Evaluation

Table 4: Large-scale scene evaluation on MatrixCity under different workload budgets. The learned confidence ordering provides continuous LoD control from a single checkpoint.

Matrix City	$s = 1$	$s = 2$	$s = 3$	$s = 4$	$s = 5$
PSNR	28.05	27.86	27.61	27.07	25.84
SSIM	0.880	0.869	0.857	0.837	0.804
LPIPS	0.171	0.199	0.225	0.254	0.294

We further evaluate whether the learned retention confidence remains effective when scaling to larger scene representations. Following the partitioning strategy of CityGS Liu et al. (2024), we apply the same confidence-based routing strategy for workload control without introducing scene-specific modifications.

table 4 reports the reconstruction quality under different workload levels on MatrixCity. The results show that the confidence ordering learned from local anchor decisions remains stable as the representation scale increases. A single checkpoint provides a continuous quality–workload trade-off, reducing the effective rendering workload while preserving high-fidelity reconstruction.

4.6 Threshold Gate Rendering Evaluation

We further evaluate the native threshold-gated inference mode, where the learned confidence is directly thresholded at a requested virtual scale s. As s increases, the gate progressively removes low-confidence anchors, reducing the effective Gaussian workload and increasing rendering throughput, while reconstruction quality degrades gradually. On Mip-NeRF 360, FPS improves from 299.73 at $s = 1$ to 352.91 at $s = 4$, while PSNR decreases from 27.71 to 26.74 dB. On Tanks&Temples, throughput increases from 274.28 to 330.23 FPS, with PSNR changing only from 23.97 to 23.60 dB over the same range. SSIM and LPIPS follow the same gradual trend. Together with the corresponding effective Gaussian counts, these results show that native threshold gating exposes a continuous quality–workload–speed trade-off from a single checkpoint. FPS is computed as the reciprocal of the mean per-view wall-clock rendering time after warm-up, and is measured as end-to-end rendering throughput at a fixed resolution.

Table 5: Rendering throughput under different LoD budgets on a single RTX 6000D GPU. Lower retained Gaussian workload leads to higher rendering FPS.

Dataset	metrics	$s = 1$	$s = 2$	$s = 3$	$s = 4$
Mip-NeRF 360	PSNR	27.71	27.37	27.10	26.74
	SSIM	0.825	0.814	0.800	0.782
	LPIPS	0.175	0.201	0.225	0.251
	FPS	299.73	337.32	347.85	352.91
	N	437741	212275	144374	112721
Tanks&Temples	PSNR	23.97	23.87	23.77	23.60
	SSIM	0.856	0.849	0.841	0.828
	LPIPS	0.161	0.181	0.198	0.222
	FPS	274.28	312.08	323.02	330.23
	N	483292	234348	164502	121924

4.7 Representation-Agnostic Routing on Vanilla 3DGS

Table 6: Routing adaptation on vanilla 3DGS with frozen Gaussian attributes. The adapter only learns primitive retention ordering under different workload budgets.

Dataset	origin	$s = 2$	$s = 3$	$s = 4$	$s = 5$
Mip-NeRF 360	28.111	27.104	26.169	25.091	22.787
Tanks&Temples	23.754	22.930	22.382	21.690	20.933

Our main implementation adopts Octree-GS anchors. To examine whether the proposed retention confidence is inherently tied to this hierarchical representation, we graft the routing mechanism onto a vanilla 3DGS representation.

Starting from a trained 3DGS checkpoint, each Gaussian is treated as an independent routing unit. All Gaussian attributes are frozen, and the adapter only predicts which primitives should be submitted to the rasterizer. A small autoencoder compresses non-positional Gaussian attributes into a 32-dimensional latent representation, which is frozen after reconstruction training.

We impose the workload constraint through sampled budgets with exact top-k selection, allowing reconstruction supervision to solely determine which Gaussians should be retained. table 6 shows that the adapter preserves competitive reconstruction quality across different budgets, despite having no ability to modify Gaussian attributes.

5 Conclusion and Limitations

We introduced Necromantic Gaussian, a continuous LoD framework that learns how to allocate a rendering budget across a 3D Gaussian representation. Rendering supervision determines which anchors should retain confidence, while direct decoded-Gaussian count supervision calibrates how much workload a requested budget should preserve. A soft transition stabilizes the ordering before hard routing, and a zero-opacity backward path keeps rejected anchors recoverable afterward without changing the hard forward image. Scale compensation provides additional coverage repair at reduced workloads, while joint optimization adapts the base representation across capacities. The resulting confidence ordering supports a continuous range of workload-controlled subsets from a single checkpoint.

The method remains limited by the coverage of the base representation and the anchor granularity of selection. Zero-opacity sensitivity can weaken under occlusion, and discrete threshold crossings can introduce temporal changes near a selection boundary. Future work will study temporally coherent confidence and finer workload control.

AI use statement

In this work, we used generative AI tools for the literature review, namely locating and reading related work; for writing the research code; and for producing figures. We have not used generative AI tools to generate research ideas, hypotheses, or the scientific claims of this work, and no other task requiring disclosure applies to this work. Additionally, we used generative AI tools for manuscript

editing and language revision. We have reviewed all AI-assisted work: every suggested reference was read and checked against the original publication, and we confirmed through a manual literature survey that no AI-assisted text reproduces prior work. AI-assisted code was reviewed and its behaviour validated against independent references and the evaluation path was checked against previously recorded baseline metrics. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Our method is implemented as a modification of the publicly released Octree-GS codebase (commit 9db64fe), and the supplementary material contains a patch against that release. This is a partial release: it covers the proposed mechanism, the modified rasterizer, and our training and evaluation entry points. It does not include the drivers for the ablation study, the large-scale partitioned-scene study (Section 4.5), the representation-agnostic study on vanilla 3DGS (Section 4.7), or the controlled toy analysis (Section A.1). More detailed configurations and the set of experimental will be released upon acceptance.

REFERENCES

Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5835–5844. IEEE, 2021.

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5460–5469. IEEE, 2022.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Francesco Di Sario, Riccardo Renzulli, Marco Grangetto, Akihiro Sugimoto, and Enzo Tartaglione. Gode: Gaussians on demand for progressive level of detail and scalable compression. *arXiv preprint arXiv:2501.13558*, 2025.

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *Advances in neural information processing systems*, 37:140138–140158, 2024.

Guangchi Fang and Bing Wang. Mini-splatting: Representing scenes with a constrained number of gaussians. In *European conference on computer vision*, pp. 165–181. Springer, 2024.

Zhilin Guo, Boqiao Zhang, Hakan Aktas, Kyle Fogarty, Jeffrey Hu, Nursena Koprucu Aslan, Wenzhao Li, Canberk Baykal, Albert Miao, Josef Bengtson, et al. Matryoshka gaussian splatting. *arXiv preprint arXiv:2603.19234*, 2026.

Hugues Hoppe. View-dependent refinement of progressive meshes. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 189–198, 1997.

Hugues Hoppe. Progressive meshes. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pp. 111–120. 2023.

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

Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A hierarchical 3d gaussian representation for real-time rendering of very large datasets. *ACM Transactions On Graphics (TOG)*, 43(4):1–15, 2024.

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017.

Jonas Kulhanek, Marie-Julie Rakotosaona, Fabian Manhardt, Christina Tsalicoglou, Michael Niemeyer, Torsten Sattler, Songyou Peng, and Federico Tombari. Lodge: Level-of-detail large-scale gaussian splatting with efficient rendering. *arXiv preprint arXiv:2505.23158*, 2025.

Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian representation for radiance field. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21719–21728. IEEE, 2024.

Yang Liu, Chuanchen Luo, Lue Fan, Naiyan Wang, Junran Peng, and Zhaoxiang Zhang. Citygaussian: Real-time high-quality large-scale scene rendering with gaussians. In *European Conference on Computer Vision*, pp. 265–282. Springer, 2024.

Yifei Liu, Zhihang Zhong, Yifan Zhan, Sheng Xu, and Xiao Sun. Maskgaussian: Adaptive 3d gaussian representation from probabilistic masks. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 681–690. IEEE, 2025.

Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.

Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20654–20664. IEEE, 2024.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Zhenhua Ning, Xin Li, Jun Yu, Guangming Lu, Yaowei Wang, and Wenjie Pei. Beyond heuristics: Learnable density control for 3d gaussian splatting. *arXiv preprint arXiv:2605.00408*, 2026.

Kerui Ren, Lihan Jiang, Tao Lu, Mulin Yu, Linning Xu, Zhangkai Ni, and Bo Dai. Octree-gs: Towards consistent real-time rendering with lod-structured 3d gaussians. *arXiv preprint arXiv:2403.17898*, 2024.

Yunji Seo, Young Sun Choi, Hyun Seung Son, and Youngjung Uh. Flod: Integrating flexible level of detail into 3d gaussian splatting for customizable rendering. *arXiv preprint arXiv:2408.12894*, 2024.

Kaifa Yang, Qi Yang, Yiling Xu, and Zhu Li. Rap: Fast feedforward rendering-free attribute-guided primitive importance score prediction for efficient 3d gaussian splatting processing. *arXiv preprint arXiv:2602.19753*, 2026.

Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19447–19456. IEEE, 2024.