ACD-GS: Asymmetric Curvature-aware Densification for 3D Gaussian Splatting

Houqiang Zhong[1*] **Tianchi Zhu**[1,3*] **Qiang Hu**[2†] **Xiaoyun Zhang**[2]
Li Song[1†] **Wenjun Zhang**[1]
[1] School of Information Science and Electronic Engineering
[2] Cooperative Medianet Innovation Center
[3] SJTU Paris Elite Institute of Technology
Shanghai Jiao Tong University, Shanghai, China
{zhonghouqiang,zhutianchi,qiang.hu,xiaoyun.zhang}@sjtu.edu.cn
{song_li,zhangwenjun}@sjtu.edu.cn

Abstract

3D Gaussian Splatting (3DGS) has emerged as a powerful paradigm for real-time photorealistic novel view synthesis. However, its training-time densification remains largely heuristic, often resulting in redundant Gaussian proliferation and inaccurate geometric reconstruction. Existing methods predominantly rely on first-order screen-space gradients to trigger cloning and splitting, which fail to distinguish true geometric discontinuities from high-frequency textures or noisy residuals. To address this limitation, we propose **ACD-GS**, an Asymmetric Curvature-aware Densification framework for compact and geometry-aware 3D Gaussian optimization. Our key insight is that second-order depth responses provide a more reliable structural prior for density control, enabling asymmetric regulation of cloning and splitting to better capture geometric discontinuities. Building on this, we introduce a multi-view photometric consistency constraint to suppress transient single-view errors, together with a capacity-aware adaptive regulation strategy that balances densification and pruning during training. Unlike post-hoc compression or quantization-based approaches, ACD-GS directly prevents redundant Gaussian generation during training, yielding inherently compact representations. Extensive experiments demonstrate that our method reduces nearly **60%** of Gaussian count and storage while maintaining competitive rendering fidelity, consistently outperforming existing methods across multiple datasets. Our code is available in https://waveviewer.github.io/ACD-GS/.

1 Introduction

3D Gaussian Splatting (3DGS) [1] has rapidly emerged as a powerful paradigm for real-time photorealistic novel view synthesis. Unlike implicit neural radiance fields, 3DGS represents a scene as an explicit collection of anisotropic Gaussian primitives and renders them through differentiable rasterization, achieving both high rendering fidelity and remarkable efficiency. Benefiting from its explicit representation and fast optimization, 3DGS has inspired extensive advances in anti-aliasing [2–4], surface reconstruction [5–9], dynamic scene modeling [10–14], compression [15–19], and hybrid implicit-explicit representations [20–25]. Despite these advances, the efficiency and scalability of 3DGS still critically depend on how Gaussian primitives are created and evolved during optimization.

In standard 3DGS, densification is mainly driven by accumulated screen-space positional gradients, where Gaussians are cloned or split once the gradient magnitude exceeds predefined thresholds.

*Equal contributions.
†Corresponding authors.

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



Figure 1: The point-cloud visualizations show our **ACD-GS** produces sparser Gaussians while preserving scene structure. The right plot (training time vs. storage, color-coded by PSNR) confirms that we achieve lower storage and faster training without sacrificing rendering quality.

While simple and effective in practice, this mechanism remains largely geometry-agnostic, as first-order gradients primarily reflect local optimization pressure rather than underlying scene structure. Consequently, regions with high-frequency textures, smoothly varying surfaces, and sharp geometric boundaries may all produce similarly large gradients despite requiring fundamentally different density-control behaviors. A flat textured wall, for example, should avoid excessive Gaussian proliferation, whereas thin structures and depth discontinuities require localized geometric refinement. This reveals an intrinsic ambiguity in gradient-driven densification, where first-order photometric signals alone are insufficient for reliable density control.

This ambiguity leads to a persistent trade-off in existing densification strategies [26–30]. Aggressive thresholds tend to trigger excessive Gaussian growth in textured regions, resulting in redundant primitives, increased storage overhead, unstable optimization, and floating artifacts near foreground-background transitions. Conversely, conservative thresholds often fail to reconstruct fine structures and sharp depth discontinuities. Recent approaches attempt to improve densification stability by incorporating pixel-aware gradients [27], absolute gradient magnitudes [28], or gradient-direction statistics [29, 30]. Although these methods alleviate optimization instability to some extent, they still rely primarily on first-order, single-view photometric signals and therefore remain insufficient for disentangling true geometric complexity from texture-induced residuals. As a result, existing methods struggle to simultaneously achieve compact representations, geometric fidelity, and stable optimization. These limitations motivate the need for a geometry-aware density-control prior that directly reflects scene structure rather than relying solely on photometric optimization cues.

In this work, we propose **ACD-GS**, an Asymmetric Curvature-aware Densification framework that reformulates Gaussian densification as a geometry-aware optimization process. Our key insight is that second-order depth responses provide a more reliable structural cue for identifying geometric discontinuities than first-order screen-space gradients. Compared with gradient magnitude alone, depth curvature is more sensitive to local geometric changes such as occlusion boundaries, thin structures, and high-curvature regions, while remaining less responsive to purely planar textured areas. Based on this observation, ACD-GS introduces an asymmetric densification mechanism that explicitly decouples cloning and splitting behaviors according to curvature-aware structural priors. Specifically, cloning is suppressed near depth discontinuities to avoid ambiguous foreground-background duplication and floating artifacts, whereas splitting is promoted to refine boundary-crossing Gaussians into finer geometric primitives.

To further resolve the ambiguity between geometry and appearance, we introduce a multi-view photometric error gating strategy that enforces cross-view consistency instead of relying on volatile single-view residuals. Furthermore, we design a capacity-constrained adaptive optimization scheme that dynamically regulates primitive growth and lifecycle-aware pruning throughout training. Together, these designs establish a unified geometry-aware density-control framework that enables native sparsification, where redundant Gaussian primitives are prevented during training rather than removed through post-hoc compression or quantization pipelines. Extensive experiments demonstrate that ACD-GS reduces nearly **60%** of Gaussian primitives and storage while maintaining competitive rendering fidelity across multiple datasets, consistently achieving a favorable quality and compactness trade-off compared with existing methods.

Our contributions are summarized as follows:

- We propose **ACD-GS**, an asymmetric curvature-aware densification framework that leverages second-order depth priors to explicitly decouple cloning and splitting behaviors for geometry-aware Gaussian evolution.
- We introduce a multi-view photometric error gating strategy and a capacity-constrained adaptive optimization scheme to suppress transient photometric noise, regulate primitive growth, and encourage compact yet high-fidelity representations.
- Extensive experiments demonstrate that ACD-GS significantly reduces Gaussian count and model storage natively during training while maintaining competitive rendering quality across multiple datasets.

2 Related Work

2.1 3D Gaussian Splatting Representation

3D Gaussian Splatting (3DGS) [1] represents a scene as anisotropic Gaussian primitives and renders them through differentiable rasterization. Its explicit formulation has inspired extensions for anti-aliasing [2–4], surface reconstruction [5–9], dynamic scenes [10–14], and appearance modeling [31–34]. Hybrid representations such as Scaffold-GS [20], SplatNet [23], and Octree-GS [21] further organize primitives with learned features or hierarchical structures, while implicit fields [35–38] provide complementary scene parameterizations. These methods mainly change representation or rendering; the number and placement of Gaussians still depend on training-time density control, which is the focus of this work. Because all of them retain a primitive-based rendering pipeline, their efficiency and fidelity are also affected by how many Gaussians remain after optimization.

2.2 Gradient-Driven Density Control

Standard 3DGS [1] densifies Gaussians using accumulated screen-space positional gradients. Pixel-GS [27] accounts for pixel coverage, AbsGS [28] uses absolute gradients to reduce cancellation, and GDAGS [29] incorporates gradient-direction information; other methods add external constraints [26, 30]. FastGS [39] instead estimates Gaussian importance from multi-view reconstruction errors inside projected footprints and uses this score for both densification and pruning. ACD-GS shares the use of multi-view evidence but assigns it a different role: rendered-depth curvature modulates cloning and splitting, while center-projected RGB residuals only verify whether a candidate is consistently under-reconstructed across views. This separates structural guidance from photometric verification and avoids using a footprint error score as the direct densification criterion.

2.3 3DGS Compression

Post-hoc compression reduces Gaussian storage through significance-based pruning and vector quantization [40, 41, 15, 42–44], while context models and entropy coding target rate–distortion efficiency [17, 18, 45–47]. Mini-Splatting [48] studies compact representations under constrained primitive budgets. These approaches mainly operate after or around training and may require codebooks or fine-tuning. ACD-GS instead reduces redundant primitive generation during optimization, without a post-hoc quantization stage. Because it changes where capacity is allocated rather than how parameters are encoded, it can be combined with downstream compression; this combination is not evaluated here. This distinction also delimits the scope of our experiments, which measure training-time sparsification rather than gains from entropy coding or quantization.

3 Method

3.1 Problem Formulation and Overview

3D Gaussian Splatting (3DGS) represents a scene as a set of anisotropic Gaussian primitives $\mathcal{G} = \{G_i\}_{i=1}^{N}$ optimized through differentiable alpha blending. During training, standard 3DGS densifies visible Gaussians according to their accumulated view-space positional gradients:

$$g_i = \sum \left\| \frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_i^{2D}} \right\|, \tag{1}$$

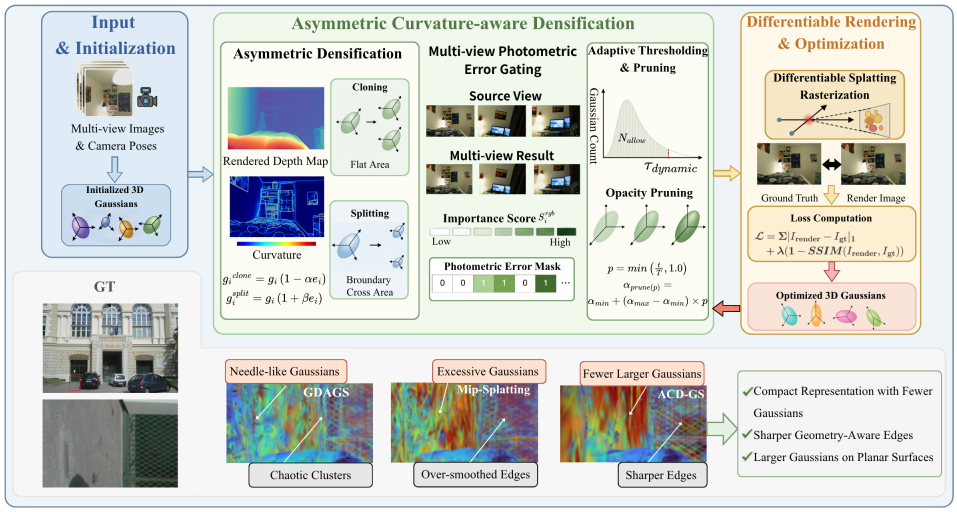


Figure 2: Overview of ACD-GS. Curvature-guided asymmetric densification suppresses cloning and promotes splitting near depth boundaries; multi-view photometric gating verifies candidates; capacity-aware thresholding and opacity pruning regulate growth. Depth is rendered from the current Gaussians rather than input.

where $\boldsymbol{\mu}_i^{2D}$ is the projected 2D center of G_i. A Gaussian is cloned or split once g_i exceeds a fixed threshold τ_{grad}, with the operation determined by scale-based heuristics. However, such first-order gradient-driven densification is geometry-agnostic. Large photometric gradients may be caused by flat textures, smooth slopes, or true depth discontinuities, although these cases require different density-control behaviors. This ambiguity often leads to redundant primitives on planar or textured regions, while sharp boundaries and thin structures may remain under-refined.

We propose **Asymmetric Curvature-aware Densification for 3D Gaussian Splatting** (**ACD-GS**), a plug-and-play density-control module that replaces the heuristic densification step in 3DGS to prevent redundant primitive generation at its source (Fig. 2). At its core, ACD-GS utilizes a second-order depth curvature prior, which is highly responsive to structural discontinuities but robust to planar textures, to asymmetrically suppress cloning near geometric edges while actively promoting the splitting of boundary-crossing Gaussians. To further resolve the geometry-texture trade-off, a multi-view photometric gate restricts growth to candidates exhibiting consistent cross-view RGB residuals, effectively filtering transient noise. Finally, capacity-constrained adaptive thresholding and lifecycle-aware opacity pruning dynamically regulate the global primitive scale under a scene-aware quota. By fundamentally curbing proliferation during optimization, ACD-GS yields highly compact representations without the need to compress an already overgrown Gaussian set post-training.

3.2 Curvature-Guided Asymmetric Densification

To distinguish geometric structures from texture-induced photometric variations, we introduce a curvature-guided structural prior. Given a rendered depth map D, we compute a curvature proxy using the magnitude of the discrete Laplacian:

$$D^{(2)} = \left|\nabla^2 D\right|. \tag{2}$$

This second-order response is less sensitive to high-frequency textures on locally planar surfaces, while producing strong responses at depth discontinuities and high-curvature regions.

To improve robustness against isolated depth spikes, we clip and normalize the curvature response:

$$\tilde{D}^{(2)} = \text{clip}\left(D^{(2)}, 0, \mu_{D^{(2)}} + 3\sigma_{D^{(2)}}\right), \tag{3}$$

$$\hat{D}^{(2)} = \frac{\tilde{D}^{(2)}}{\max(\tilde{D}^{(2)}) + \epsilon}. \tag{4}$$

Weak responses are further suppressed by soft thresholding:

$$E = \frac{\max(\hat{D}^{(2)} - \tau, 0)}{1 - \tau}, \tag{5}$$

yielding a depth-edge confidence map $E \in [0, 1]$. We use $\tau = 0.15$. For each visible Gaussian, we sample this confidence at its projected center:

$$e_i = E(\boldsymbol{\mu}_i^{2D}), \quad e_i \in [0, 1]. \tag{6}$$

Standard 3DGS uses the same gradient statistic for both cloning and splitting, despite their different geometric effects. Cloning increases local representation capacity and is useful for texture-rich smooth regions, but may duplicate ambiguous primitives near foreground-background transitions. Splitting, instead, decomposes large Gaussians and is more suitable for refining primitives that span multiple depth layers.

We therefore introduce operation-specific asymmetric gradient modulation:

$$g_i^{\text{clone}} = g_i(1 - \alpha e_i), \quad g_i^{\text{split}} = g_i(1 + \beta e_i), \tag{7}$$

where $\alpha = 0.8$ and $\beta = 1.5$ control the suppression and enhancement strengths. Near structural boundaries, cloning is suppressed to avoid duplicating ambiguous primitives, while splitting is amplified to encourage fine-grained boundary refinement. In smooth regions, the modulated gradients reduce to the original g_i, preserving the baseline capacity for appearance modeling.

3.3 Multi-view Photometric Error Gating

Although curvature provides a useful structural prior, photometric signals remain necessary for densifying texture-rich flat regions. However, single-view residuals may be unstable due to occlusion, view-dependent effects, or transient optimization noise. We therefore verify photometric necessity across multiple views.

Before each densification step, we sample $K = 8$ auxiliary cameras from the training set. For each view k, we render an image $I_{\text{render}}^{(k)}$ and compare it with the ground-truth image $I_{\text{gt}}^{(k)}$ using the mean absolute RGB residual:

$$E_{\text{rgb}}^{(k)} = \frac{1}{3} \sum_{c=1}^{3} \left| I_{\text{render},c}^{(k)} - I_{\text{gt},c}^{(k)} \right| \odot M_{\text{valid}}^{(k)}, \tag{8}$$

where $M_{\text{valid}}^{(k)}$ is the valid image mask. The residual map is binarized as

$$\mathcal{B}_{\text{rgb}}^{(k)} = \mathbb{I}\left(E_{\text{rgb}}^{(k)} > \tau_{\text{rgb}} \right) \odot M_{\text{valid}}^{(k)}. \tag{9}$$

For each Gaussian G_i, we project its 3D center $\boldsymbol{\mu}_i$ to view k and sample the binary residual map:

$$e_{i,k}^{\text{rgb}} = \mathcal{B}_{\text{rgb}}^{(k)}\left(\mathbf{p}_{i,k}\right), \tag{10}$$

where $\mathbf{p}_{i,k}$ is the projected screen-space coordinate. Let $v_{i,k} \in \{0, 1\}$ indicate whether the projection lies in the valid image region. The multi-view RGB error score is then

$$S_i^{\text{rgb}} = \frac{\sum_{k=1}^{K} v_{i,k} e_{i,k}^{\text{rgb}}}{\max\left(\sum_{k=1}^{K} v_{i,k}, 1 \right)}. \tag{11}$$

We define the photometric gate as

$$\mathcal{M}_i^{\text{rgb}} = \mathbb{I}\left(S_i^{\text{rgb}} \geq \tau_{\text{mv}} \right). \tag{12}$$

Only Gaussians satisfying $\mathcal{M}_i^{\text{rgb}} = 1$ are admitted into the densification candidate pool. This cross-view verification suppresses transient single-view residuals while preserving under-reconstructed appearance details. The valid-projection indicator checks image support and should not be interpreted as an explicit geometric occlusion test.

3.4 Capacity-Constrained Adaptive Optimization

Fixed densification and pruning thresholds are suboptimal because gradient distributions and primitive scales vary across scenes and training stages. We therefore regulate both primitive growth and removal according to the current optimization state.

Capacity-Adaptive Densification. At each densification interval, we estimate an adaptive threshold from the valid gradient distribution. Let N_{total} denote the current number of Gaussians, and let $\mathcal{G}_{\text{valid}} = \{g_i \mid g_i > 0\}$ contains the positive accumulated position gradients used for threshold estimation. We impose a scene-dependent growth quota:

$$N_{\text{allow}} = \lfloor \mu N_{\text{total}} \rfloor , \tag{13}$$

where $\mu = 0.008$ (0.8%) is the default quota ratio. Let $M = |\mathcal{G}_{\text{valid}}|$, and sort the candidate gradients as $g_{(1)} \geq g_{(2)} \geq \cdots \geq g_{(M)}$. The base adaptive threshold is

$$\hat{\tau}_{\text{dyn}} = \begin{cases} g_{(N_{\text{allow}})}, & M > N_{\text{allow}}, \\ g_{(k_{\text{top}})}, & M \leq N_{\text{allow}}, \end{cases} \quad k_{\text{top}} = \max(1, \lceil 0.1M \rceil). \tag{14}$$

When the number of candidates exceeds the quota, this threshold directly bounds the number of densification operations. Otherwise, the top-10% fallback avoids selecting marginal gradients. The final threshold is lower-bounded by a relaxed baseline threshold:

$$\tau_{\text{dyn}} = \max\left(0.8\tau_{\text{base}}, \hat{\tau}_{\text{dyn}}\right). \tag{15}$$

Lifecycle-Aware Opacity Pruning. We further adapt opacity pruning according to training progress. Let

$$p = \min\left(\frac{t}{T}, 1\right), \tag{16}$$

where t and T are the current and total training iterations. The opacity pruning threshold is annealed as

$$\alpha_{\text{prune}}(p) = \alpha_{\min} + (\alpha_{\max} - \alpha_{\min})p. \tag{17}$$

This preserves exploratory low-opacity primitives in early training, while enforcing stricter removal of redundant primitives near convergence.

Overall, ACD-GS couples structural densification, cross-view photometric verification, and capacity-aware optimization. The resulting training process produces compact Gaussian representations without relying on post-hoc compression.

4 Experiments

Experimental Setup. We evaluate ACD-GS on the Mip-NeRF 360 [49], Tanks & Temples [50], and Deep Blending [51] datasets against vanilla 3DGS [1] and representative baselines including Mip-Splatting [2], Mini-Splatting [48], AbsGS [28], GDAGS [29], and Pixel-GS [27]. All models are trained for 30K iterations on a single NVIDIA RTX 5090 GPU. Our curvature-aware density control operates every 100 steps from iteration 600 until training completion to strictly regulate primitive proliferation. We report rendering metrics (PSNR, SSIM, LPIPS) and efficiency statistics (storage size, Gaussian count, training time). Further implementation details are provided in the Appendix.

4.1 Quantitative Comparison

Rendering Fidelity. On the Deep Blending dataset (Tab. 1), ACD-GS achieves the best overall rendering quality among all evaluated methods, obtaining the highest PSNR (25.12) and SSIM (0.834). This demonstrates that our compact representation does not compromise the reconstruction of visually important structures. Compared with vanilla 3DGS, ACD-GS consistently produces sharper reconstructions while avoiding representation size bloat caused by unconstrained densification. For the large-scale indoor and outdoor scenes in Mip-NeRF 360 and Tanks & Temples (Tab. 2), ACD-GS maintains highly competitive visual quality and matches or exceeds strong gradient-based baselines such as AbsGS and GDAGS. These results suggest that the proposed curvature-aware prior effectively

Table 1: Quantitative Comparison on Deep Blending

Method	Aquarium				Bedroom			
	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓
3DGS	0.869	22.66	0.161	177.44	0.907	28.86	0.274	316.93
Mip-Splatting	0.851	22.52	0.176	272.31	0.904	28.68	0.274	473.69
Mini-Splatting	0.851	23.81	0.166	810.17	0.901	28.28	0.295	981.14
AbsGS	0.806	21.82	0.229	183.16	0.901	28.85	0.275	253.97
GDAGS	0.842	22.18	0.186	159.86	0.907	29.05	0.270	283.22
Pixel-GS	0.865	25.77	0.158	268.57	0.904	28.82	0.275	719.19
ACD-GS	0.874	26.42	0.166	41.76	0.907	29.04	0.290	111.74

Method	CreepyAttic				Hugo			
	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓
3DGS	0.892	30.52	0.225	758.95	0.843	23.85	0.139	321.95
Mip-Splatting	0.886	30.16	0.221	1142.25	0.830	22.80	0.149	494.73
Mini-Splatting	0.868	29.41	0.280	1172.18	0.841	23.24	0.143	873.89
AbsGS	0.882	30.16	0.220	704.01	0.789	22.58	0.180	344.76
GDAGS	0.889	30.43	0.214	711.70	0.821	23.39	0.160	318.72
Pixel-GS	0.884	30.24	0.222	1558.26	0.836	23.43	0.139	496.32
ACD-GS	0.891	30.70	0.244	165.98	0.847	24.45	0.161	110.88

Method	Library				Museum-1			
	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓
3DGS	0.781	23.87	0.352	501.54	0.836	23.32	0.135	567.26
Mip-Splatting	0.763	23.32	0.358	829.34	0.836	23.28	0.135	807.78
Mini-Splatting	0.762	23.33	0.394	1129.81	0.845	23.20	0.135	938.93
AbsGS	0.731	23.49	0.416	738.66	0.793	21.73	0.169	568.56
GDAGS	0.768	23.72	0.357	568.79	0.834	22.69	0.138	554.20
Pixel-GS	0.777	23.73	0.358	998.51	0.835	23.15	0.134	841.11
ACD-GS	0.780	24.10	0.359	245.24	0.845	23.49	0.145	223.28

Method	Museum-2				Tree			
	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓
3DGS	0.897	25.34	0.099	333.72	0.797	20.75	0.170	223.22
Mip-Splatting	0.896	25.81	0.098	477.92	0.789	21.06	0.176	289.15
Mini-Splatting	0.905	27.50	0.089	877.42	0.804	21.98	0.171	798.00
AbsGS	0.880	25.48	0.111	310.55	0.746	21.03	0.210	216.07
GDAGS	0.899	25.64	0.095	299.60	0.770	21.26	0.205	192.28
Pixel-GS	0.898	26.58	0.093	561.85	0.796	21.76	0.167	327.29
ACD-GS	0.900	27.61	0.106	121.68	0.805	22.22	0.165	84.78

Method	Yellowhouse				Mean			
	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓
3DGS	0.655	17.11	0.295	176.08	0.831	24.03	0.205	375.23
Mip-Splatting	0.641	16.75	0.316	238.22	0.822	23.82	0.211	558.38
Mini-Splatting	0.639	16.27	0.316	749.61	0.824	24.11	0.221	925.68
AbsGS	0.581	16.29	0.352	236.54	0.790	23.49	0.240	395.14
GDAGS	0.637	16.63	0.308	180.94	0.819	23.89	0.215	363.26
Pixel-GS	0.667	17.61	0.282	267.99	0.829	24.56	0.203	671.01
ACD-GS	0.661	18.09	0.297	95.58	0.834	25.12	0.215	133.44

preserves essential structural and textural details, enabling compactness without sacrificing rendering fidelity.

Model Compactness and Training Efficiency. Tab. 3 reports the efficiency comparison in terms of storage footprint, Gaussian count, and training time. Across all three dataset groups, ACD-GS consistently achieves the smallest model size and the fastest training among the compared methods. This indicates that our native sparsification strategy controls redundancy during optimization rather than relying on post-hoc compression. Compared with GDAGS, ACD-GS reduces average storage from 360.6 MB to 151.9 MB on Deep Blending, corresponding to a 57.8% reduction; from 707.1 MB to 274.3 MB on Mip-NeRF 360, corresponding to a 61.2% reduction; and from 319.5 MB to 150.4 MB on Tanks & Temples, corresponding to a 52.9% reduction. The Gaussian count is reduced by roughly 60% across the datasets, which further lowers memory usage and computational cost during training.

Table 2: Quantitative Comparison on Tanks & Temples and Mip-NeRF 360

Method	Train				Truck			
	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓
3DGS	0.8202	22.04	0.1963	258.35	0.8851	25.47	0.1418	487.82
Mip-Splatting	0.8266	21.87	0.1893	354.45	0.8930	25.74	0.1231	784.70
Mini-Splatting	0.7099	18.77	0.3234	937.32	0.8573	23.66	0.1583	1084.63
AbsGS	0.8166	21.64	0.1942	230.86	0.8874	25.71	0.1314	388.02
GDAGS	0.8129	21.53	0.2068	223.96	0.8858	25.49	0.1380	415.07
Pixel-GS	0.8279	22.13	0.1780	906.22	0.8868	25.48	0.1203	1228.62
ACD-GS	0.8103	22.01	0.2298	123.27	0.8813	25.51	0.1434	177.46

Method	Bicycle				Stump			
	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓
3DGS	0.7666	25.28	0.2081	1154.55	0.7714	26.66	0.2163	1026.20
Mip-Splatting	0.7924	25.55	0.1669	1977.66	0.7902	26.89	0.1883	1440.26
Mini-Splatting	0.7165	23.57	0.2508	1425.72	0.7601	26.03	0.2282	1274.58
AbsGS	0.7817	25.26	0.1720	1411.36	0.7768	26.66	0.1967	1106.18
GDAGS	0.7899	25.42	0.1696	1437.29	0.7743	26.96	0.1874	949.72
Pixel-GS	0.7617	25.26	0.1720	2142.21	0.7768	26.66	0.1967	1618.47
ACD-GS	0.7861	25.47	0.1799	543.99	0.7856	26.83	0.2051	318.65

Method	Treehill				Mean			
	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓
3DGS	0.6335	22.49	0.3251	770.88	0.7754	24.39	0.2175	739.56
Mip-Splatting	0.6387	22.34	0.2744	1223.14	0.7882	24.48	0.1884	1156.04
Mini-Splatting	0.6244	22.06	0.3254	135.32	0.7336	22.82	0.2572	971.51
AbsGS	0.6139	21.96	0.2795	1221.96	0.7753	24.25	0.1948	871.68
GDAGS	0.6246	22.06	0.2786	1171.14	0.7775	24.29	0.1961	839.44
Pixel-GS	0.6139	21.96	0.2795	1868.62	0.7734	24.30	0.1893	1552.83
ACD-GS	0.6457	22.36	0.2936	472.37	0.7818	24.44	0.2104	327.15

Table 3: Efficiency Metrics Comparison

Method	Deep Blending			Mip-NeRF 360			Tanks & Temples		
	Time(s)↓	Size(MB)↓	Points(M)↓	Time(s)↓	Size(MB)↓	Points(M)↓	Time(s)↓	Size(MB)↓	Points(M)↓
3DGS	842	367.30	1.5500	683	638.40	2.7000	552	373.10	1.5800
Mip-Splatting	1007	563.20	2.3400	956	991.10	4.1200	716	569.60	2.3700
Mini-Splatting	1125	959.10	4.1300	893	1108.60	4.6900	857	1011.00	4.2700
AbsGS	795	400.40	1.6900	683	716.40	3.0300	433	309.40	1.3100
GDAGS	628	360.60	1.5800	517	707.10	2.9900	350	319.50	1.3500
Pixel-GS	978	694.80	2.9400	1083	1315.10	5.5600	946	1067.40	4.5100
ACD-GS	545	151.90	0.6420	345	274.30	1.1600	333	150.40	0.6357

Trade-off Advantage. The key advantage of ACD-GS lies in its favorable quality and compactness trade-off. Recent density-control methods, such as Pixel-GS, often improve rendering quality by introducing excessive numbers of primitives, reaching 5.56M Gaussians on Mip-NeRF 360. In contrast, ACD-GS suppresses redundant Gaussian generation at its optimization source through curvature-aware asymmetric clone and split modulation and multi-view gating. As confirmed by the storage–quality plot in Fig. 1, ACD-GS occupies the most favorable region of the trade-off space, achieving competitive fidelity with substantially lower storage. This demonstrates that our method effectively separates geometrically necessary densification from texture-induced or view-specific noise, leading to a compact yet expressive Gaussian representation.

4.2 Qualitative Comparison

Visual Rendering Fidelity. As shown in Fig. 3, ACD-GS achieves novel view synthesis quality that is highly comparable to, and occasionally surpasses, the unconstrained vanilla 3DGS model. Gradient-based baselines like AbsGS and GDAGS frequently suffer from texture degradation to maintain a lower point count, our method consistently preserves high-frequency details. Crucially, vanilla 3DGS is notoriously prone to generating severe floating artifacts in empty spaces (visibly evident in the sky region of the Tree scene) due to its geometric blindness. In contrast, by rigorously suppressing redundant densification at its source, ACD-GS effectively eliminates these floaters and reconstructs remarkably clean backgrounds and sharp object boundaries.



Figure 3: Qualitative comparison of novel view synthesis on the Deep Blending dataset.

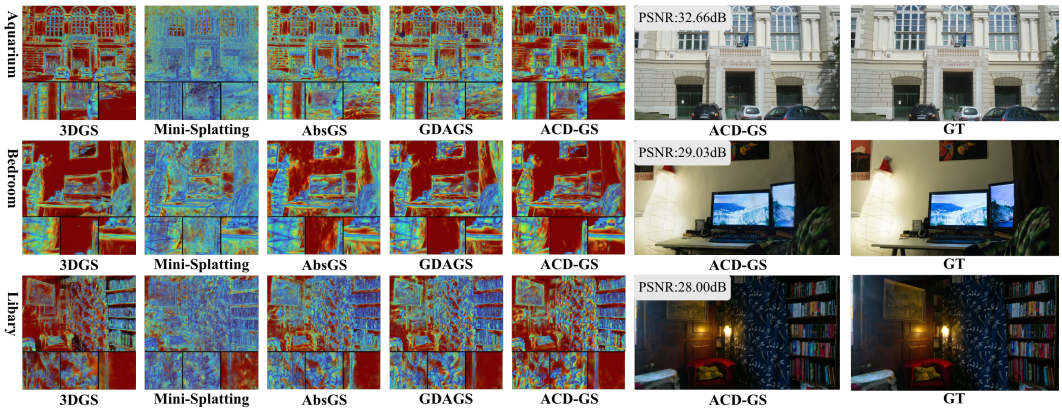


Figure 4: Visualization of Gaussians distribution

Primitive Distribution Analysis. To uncover the structural rationale behind our efficiency and rendering quality, we visualize the spatial density of Gaussian primitives in Fig. 4. Vanilla 3DGS exhibits extreme redundancy, indiscriminately populating flat homogeneous surfaces with overly dense primitives. Other density-control baselines often fail to cleanly separate textures from geometry, leading to either noisy distributions or over-pruned structures. Conversely, the primitive allocation in ACD-GS strongly aligns with the underlying scene geometry. Guided by our second-order curvature prior, ACD-GS allocates dense and highly precise Gaussians strictly along structural edges and complex details. Meanwhile, it elegantly covers planar regions (such as the building facades in the Aquarium scene and the walls in the Bedroom scene) with much larger and highly sparse primitives. This structurally aware evolution precisely explains why ACD-GS achieves extreme model compactness and fewer floating artifacts without compromising visual fidelity. More results are shown in Appendix.

4.3 Parameter Sensitivity Analysis

Tab. 4 analyzes the sensitivity of ACD-GS to α and β. We vary one parameter at a time: α with $\beta = 1.5$ fixed, and β with $\alpha = 0.8$ fixed. Increasing α from 0.5 to 0.8 improves PSNR and SSIM while reducing LPIPS and storage. Setting $\alpha = 1.0$ yields the highest PSNR and SSIM, but also a larger LPIPS and model size. For β, the default $\beta = 1.5$ gives the best PSNR, SSIM, LPIPS, and storage among the tested values. Overall, $(\alpha, \beta) = (0.8, 1.5)$ provides the best quality–compactness balance, while $\alpha = 1.0$ offers higher PSNR and SSIM at a larger model size.

4.4 Ablation Studies

We conduct an incremental ablation study integrating our three core modules: Curvature-Guided Asymmetric Densification (CAD), Multi-view Photometric Error Gating (MPG), and Capacity-

Table 4: Comparison of different settings for α and β, varying one parameter at a time.

α	β	PSNR↑	SSIM↑	LPIPS↓	Size(MB)↓
0.5	1.5	25.5890	0.8424	0.2387	161.37
0.8	1.5	25.5913	0.8450	**0.2363**	**151.85**
1.0	1.5	**25.8259**	**0.8469**	0.2520	162.54
0.8	1.0	25.4263	0.8412	0.2451	157.35
0.8	2.5	25.4441	0.8415	0.2436	164.87

Table 5: Quantitative ablation on DeepBlending. CAD, MPG, and CAO denote the three components.

CAD	MPG	CAO	Size(MB)↓	PSNR↑	SSIM↑	LPIPS↓
×	×	×	360.60	25.22	0.838	0.225
✓	×	×	302.40	25.38	0.840	0.223
✓	✓	×	159.70	25.38	0.841	**0.221**
✓	✓	✓	**151.90**	**25.59**	**0.845**	0.236

Table 6: Sensitivity to the number of auxiliary views K and quota ratio μ.

Setting	PSNR↑	SSIM↑	LPIPS↓	Points↓	Size(MB)↓
$K = 4$	25.3380	0.8432	0.2393	**550,173**	**136.44**
$K = 8$	**25.5914**	**0.8450**	**0.2363**	642,045	151.85
$K = 12$	25.3605	0.8440	0.2390	661,947	156.88
$\mu = 0.004$	25.5145	0.8444	**0.2363**	**641,737**	**151.78**
$\mu = 0.008$	**25.5913**	**0.8450**	**0.2363**	642,044	151.85
$\mu = 0.012$	25.4211	0.8432	0.2372	642,241	151.90

Constrained Adaptive Optimization (CAO). We evaluate the model storage requirements and rendering fidelity on the DeepBlending dataset. As shown in Tab. 5, the baseline 3DGS suffers from severe memory bloat (360.6 MB) due to unconstrained primitive proliferation. Integrating the CAD module effectively addresses this by penalizing redundant cloning in flat regions and promoting boundary-aware splitting, which immediately reduces storage to 302.4 MB while improving PSNR. Subsequently, incorporating MPG further slashes the memory footprint to 159.7 MB. By demanding cross-view residual consensus, MPG prevents the model from blindly overfitting to transient single-view photometric noise, concurrently yielding strict improvements in SSIM and LPIPS. Through dynamic growth quotas and lifecycle-aware pruning, CAO curtails floating artifacts and reduces the final model size to a remarkable 151.9 MB, representing a 58% reduction from the baseline.

4.5 Sensitivity Analysis

We additionally evaluate the number of auxiliary views K and the candidate quota ratio μ on the complete Deep Blending dataset. Increasing K from 4 to 8 improves PSNR, SSIM, and LPIPS, but retains more Gaussians; increasing K to 12 provides no consistent quality benefit and increases the model size, so we use $K = 8$. For the quota ratio, $\mu = 0.004$ yields a marginally smaller model, whereas $\mu = 0.008$ achieves higher PSNR and SSIM with the same LPIPS; $\mu = 0.012$ provides no additional quality improvement. We therefore use $\mu = 0.008$ as the default.

5 Discussion and Limitations

While ACD-GS achieves significant native sparsification, extreme compression inevitably degrades high-frequency texture fidelity due to the inherent capacity limits of pure explicit representations. To address this bottleneck, future work could explore hybrid architectures, replacing explicit spherical harmonics with lightweight Multi-Layer Perceptrons (MLPs) to maintain rendering quality under stricter point budgets. Furthermore, because ACD-GS operates orthogonally to post-hoc compression pipelines, seamlessly integrating it with downstream quantization and entropy coding frameworks presents a promising direction to unlock ultra-compact assets for low-bandwidth streaming.

6 Conclusion

We present ACD-GS, an Asymmetric Curvature-aware Densification framework for compact 3D Gaussian Splatting. Second-order depth priors decouple cloning and splitting, suppressing unnecessary cloning in planar regions while promoting splitting along geometric boundaries. A multi-view photometric gate filters transient noise, and adaptive thresholding with lifecycle-aware pruning regulates capacity. Experiments show that ACD-GS reduces Gaussian count and storage by nearly 60% across benchmarks while preserving competitive rendering fidelity. By directly preventing redundancy during training, ACD-GS offers a practical, memory-efficient foundation for scalable 3D Gaussian representations.

Acknowledgments and Disclosure of Funding

This work is supported by National Natural Science Foundation of China(62431015, 62571322, 62571317),STCSM (24ZR1432000, 22DZ2229005), 111 plan (BP0719010), and State Key Laboratory of UHD Video and Audio Production and Presentation.

References

[1] B. Kerbl, G. Kopanas, T. Leimkuehler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Trans. Graph.*, vol. 42, no. 4, July 2023.

[2] Z. Yu, A. Chen, B. Huang, T. Sattler, and A. Geiger, "Mip-splatting: Alias-free 3d gaussian splatting," in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 19 447–19 456.

[3] H. Zhong, Z. Wu, S. Fu, Z. Zheng, X. Jin, X. Zhang, L. Song, and Q. Hu, "Prismgs: Physically-grounded anti-aliasing for high-fidelity large-scale 3d gaussian splatting," in *2025 International Conference on Visual Communications and Image Processing (VCIP)*, 2025, pp. 1–5.

[4] Z. Liang, Q. Zhang, W. Hu, L. Zhu, Y. Feng, and K. Jia, "Analytic-splatting: Anti-aliased 3d gaussian splatting via analytic integration," in *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part XVII*. Berlin, Heidelberg: Springer-Verlag, 2024, p. 281–297.

[5] Y. Gao, H. Zhong, T. Zhu, Z. Cheng, Q. Hu, and L. Song, "Alignngs: Aligning geometry and semantics for robust indoor reconstruction from sparse views," in *2025 International Conference on Visual Communications and Image Processing (VCIP)*, 2025, pp. 1–5.

[6] A. Guédon and V. Lepetit, "Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 5354–5363.

[7] B. Huang, Z. Yu, A. Chen, A. Geiger, and S. Gao, "2d gaussian splatting for geometrically accurate radiance fields," in *ACM SIGGRAPH 2024 Conference Papers*, ser. SIGGRAPH '24. New York, NY, USA: Association for Computing Machinery, 2024.

[8] Y. Wolf, A. Bracha, and R. Kimmel, "GS2Mesh: Surface reconstruction from Gaussian splatting via novel stereo views," in *European Conference on Computer Vision (ECCV)*, 2024.

[9] D. Chen, H. Li, W. Ye, Y. Wang, W. Xie, S. Zhai, N. Wang, H. Liu, H. Bao, and G. Zhang, "Pgsr: Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–12, 2024.

[10] Q. Hu, Z. Zheng, H. Zhong, S. Fu, L. Song, XiaoyunZhang, G. Zhai, and Y. Wang, "4dgc: Rate-aware 4d gaussian compression for efficient streamable free-viewpoint video," in *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[11] Z. Zheng, Z. Wu, H. Zhong, Y. Tian, N. Cao, L. Xu, J. Yao, X. Zhang, Q. Hu, and W. Zhang, "4DGCPro: Efficient hierarchical 4d gaussian compression for progressive volumetric video streaming," in *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. [Online]. Available: https://openreview.net/forum?id=MbqPKsVe1d

[12] Z. Li, Z. Chen, Z. Li, and Y. Xu, "Spacetime gaussian feature splatting for real-time dynamic view synthesis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 8508–8520.

[13] G. Wu, T. Yi, J. Fang, L. Xie, X. Zhang, W. Wei, W. Liu, Q. Tian, and X. Wang, "4d gaussian splatting for real-time dynamic scene rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 20 310–20 320.

[14] J. Luiten, G. Kopanas, B. Leibe, and D. Ramanan, "Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis," in *3DV*, 2024.

[15] J. C. Lee, D. Rho, X. Sun, J. H. Ko, and E. Park, "Compact 3d gaussian representation for radiance field," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 21 719–21 728.

[16] M. T. Bagdasarian, P. Knoll, Y. Li, F. Barthel, A. Hilsmann, P. Eisert, and W. Morgenstern, "3dgs.zip: A survey on 3d gaussian splatting compression methods," *Computer Graphics Forum*, p. e70078, 2025.

[17] Y. Chen, Q. Wu, W. Lin, M. Harandi, and J. Cai, "Hac: Hash-grid assisted context for 3d gaussian splatting compression," in *European Conference on Computer Vision*, 2024.

[18] Y. Wang, Z. Li, L. Guo, W. Yang, A. Kot, and B. Wen, "ContextGS : Compact 3d gaussian splatting with anchor level context model," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. [Online]. Available: https://openreview.net/forum?id=W2qGSMl2Uu

[19] Y. Chen, Q. Wu, W. Lin, M. Harandi, and J. Cai, "Hac++: Towards 100x compression of 3d gaussian splatting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 47, no. 11, pp. 10 210–10 226, 2025.

[20] T. Lu, M. Yu, L. Xu, Y. Xiangli, L. Wang, D. Lin, and B. Dai, "Scaffold-gs: Structured 3d gaussians for view-adaptive rendering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2024, pp. 20 654–20 664.

[21] K. Ren, L. Jiang, T. Lu, M. Yu, L. Xu, Z. Ni, and B. Dai, "Octree-gs: Towards consistent real-time rendering with lod-structured 3d gaussians," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[22] Y. Xin, Y. Liu, X. Xie, and X. Li, "Learning unified representation of 3d gaussian splatting," in *International Conference on Learning Representations*, 2026.

[23] xilong zhou, B.-H. Nguyen, L. Magne, V. Golyanik, T. Leimkuehler, and C. Theobalt, "Splat the net: Radiance fields with splattable neural primitives," in *The Fourteenth International Conference on Learning Representations*, 2026. [Online]. Available: https://openreview.net/forum?id=v3ejhJxT1W

[24] A. Bouzidi, H. Laga, H. Wannous, and F. Sohel, "Gnf: Gaussian neural fields for multidimensional signal representation and reconstruction," in *Computer Graphics Forum*, vol. 44, no. 7. Wiley Online Library, 2025, p. e70232.

[25] Z. Tang, C. Feng, X. Cheng, W. Yu, J. Zhang, Y. Liu, X.-X. Long, W. Wang, and L. Yuan, "Neuralgs: Bridging neural fields and 3d gaussian splatting for compact 3d representations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 40, no. 11, 2026, pp. 9493–9501.

[26] G. Grubert, F. Barthel, A. Hilsmann, and P. Eisert, "Improving adaptive density control for 3d gaussian splatting," in *Proceedings of the 20th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS - Science and Technology Publications, 2025, p. 610–621. [Online]. Available: http://dx.doi.org/10.5220/0013308500003912

[27] Z. Zhang, W. Hu, Y. Lao, T. He, and H. Zhao, "Pixel-gs: Density control with pixel-aware gradient for 3d gaussian splatting," in *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part XIX*. Berlin, Heidelberg: Springer-Verlag, 2024, p. 326–342. [Online]. Available: https://doi.org/10.1007/978-3-031-72655-2_19

[28] Z. Ye, W. Li, S. Liu, P. Qiao, and Y. Dou, "Absgs: Recovering fine details in 3d gaussian splatting," in *Proceedings of the 32nd ACM International Conference on Multimedia*, ser. MM '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 1053–1061. [Online]. Available: https://doi.org/10.1145/3664647.3681361

[29] Z. Zhou, Y.-J. Xiong, J.-C. Zhang, C.-M. Xia, X. Qiu, and H. Zhan, "Gradient-direction-aware density control for 3d gaussian splatting," in *The Fourteenth International Conference on Learning Representations*, 2026. [Online]. Available: https://openreview.net/forum?id=6qDxK4Gz7F

[30] P. Wang, Y. Wang, D. Wang, S. Mohan, Z. Fan, L. Wu, R. Cai, Y.-Y. Yeh, Z. Wang, Q. Liu, and R. Ranjan, "Steepest descent density control for compact 3d gaussian splatting," in *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, June 2025, pp. 26 663–26 672.

[31] G. Serifi and M. C. Buehler, "Hypergaussians: High-dimensional gaussian splatting for high-fidelity animatable face avatars," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

[32] C. Thirgood, O. Mendez, E. Ling, J. Storey, and S. Hadfield, "Hypergs: Hyperspectral 3d gaussian splatting," in *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, June 2025, pp. 5970–5979.

[33] R. Lu, H. Chen, Z. Zhu, Y. Qin, M. Lu, L. zhang, C. Yan, and anke xue, "Thermalgaussian: Thermal 3d gaussian splatting," in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: https://openreview.net/forum?id=ybFRoGxZjs

[34] Z. Gao, B. Planche, M. Zheng, A. Choudhuri, T. Chen, and Z. Wu, "7dgs: Unified spatial-temporal-angular gaussian splatting," in *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025, pp. 26 316–26 325.

[35] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: representing scenes as neural radiance fields for view synthesis," *Commun. ACM*, vol. 65, no. 1, p. 99–106, December 2021.

[36] T. Müller, A. Evans, C. Schied, and A. Keller, "Instant neural graphics primitives with a multiresolution hash encoding," *ACM Trans. Graph.*, vol. 41, no. 4, pp. 102:1–102:15, Jul. 2022.

[37] Z. Chen, T. Funkhouser, P. Hedman, and A. Tagliasacchi, "Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 16 569–16 578.

[38] P. Hedman, P. P. Srinivasan, B. Mildenhall, J. T. Barron, and P. Debevec, "Baking neural radiance fields for real-time view synthesis," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 5875–5884.

[39] S. Ren, T. Wen, Y. Fang, and B. Lu, "FastGS: Training 3D Gaussian Splatting in 100 seconds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026, pp. 26 094–26 103. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2026/html/Ren_FastGS_Training_3D_Gaussian_Splatting_in_100_Seconds_CVPR_2026_paper.html

[40] Z. Fan, K. Wang, K. Wen, Z. Zhu, D. Xu, Z. Wang *et al.*, "Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps," *Advances in neural information processing systems*, vol. 37, pp. 140 138–140 158, 2024.

[41] S. Girish, K. Gupta, and A. Shrivastava, "Eagles: Efficient accelerated 3d gaussians with lightweight encodings," in *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LXIII*. Berlin, Heidelberg: Springer-Verlag, 2024, p. 54–71. [Online]. Available: https://doi.org/10.1007/978-3-031-73036-8_4

[42] S. Almuhammad Alali, M. Qamar, S.-H. Bae, and E. Tartaglione, "Trimming the Fat: Efficient Compression of 3D Gaussian Splats through Pruning," in *British Machine Vision Conference 2024*, Glasgow (Ecosse), United Kingdom, Nov. 2024. [Online]. Available: https://telecom-paris.hal.science/hal-05208876

[43] S. Liu, X. Tang, Z. Li, Y. He, C. Ye, J. Liu, B. Huang, S. Zhou, and X. Wu, "Occlugaussian: Occlusion-aware gaussian splatting for large scene reconstruction and rendering," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2025, pp. 26 643–26 652.

[44] A. Hanson, A. Tu, V. Singla, M. Jayawardhana, M. Zwicker, and T. Goldstein, "Pup 3d-gs: Principled uncertainty pruning for 3d gaussian splatting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2025, pp. 5949–5958.

[45] H. Wang, H. Zhu, T. He, R. Feng, J. Deng, J. Bian, and Z. Chen, "End-to-end rate-distortion optimized 3d gaussian representation," in *European Conference on Computer Vision*, 2024.

[46] Y. Chen, M. Li, Q. Wu, W. Lin, M. Harandi, and J. Cai, "Pcgs: Progressive compression of 3d gaussian splatting," in *The 40th Annual AAAI Conference on Artificial Intelligence*, 2026.

[47] Y.-T. Zhan, C.-Y. Ho, H. Yang, Y.-H. Chen, J. C. Chiang, Y.-L. Liu, and W.-H. Peng, "CAT-3DGS: A context-adaptive triplane approach to rate-distortion-optimized 3DGS compression," in *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

[48] G. Fang and B. Wang, "Mini-splatting: Representing scenes with a constrained number of gaussians," in *European Conference on Computer Vision*. Springer, 2024, pp. 173–190.

[49] B. Mildenhall, D. Verbin, P. P. Srinivasan, P. Hedman, R. Martin-Brualla, and J. T. Barron, "MultiNeRF: A Code Release for Mip-NeRF 360, Ref-NeRF, and RawNeRF," 2022. [Online]. Available: https://github.com/google-research/multinerf

[50] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: Benchmarking large-scale scene reconstruction," *ACM Transactions on Graphics*, vol. 36, no. 4, 2017.

[51] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, and G. Brostow, "Deep blending for free-viewpoint image-based rendering," in *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, vol. 37, no. 6. ACM, 2018, pp. 257:1–257:15.

A Hyperparameter Sensitivity Analysis

We further evaluate the sensitivity of ACD-GS to three key hyperparameters, including the soft threshold τ, the capacity-control parameters (α, β), and the rate of lifecycle-aware opacity pruning. As shown in Tables A1–A3, ACD-GS exhibits stable rendering quality across different hyperparameter settings, while the representation size and the number of Gaussians vary more noticeably. For the soft threshold τ, all tested settings achieve comparable average PSNR, SSIM, and LPIPS, indicating that the proposed soft modulation is not overly sensitive to the exact threshold value. The default setting $\tau = 0.15$ provides a balanced trade-off, maintaining competitive rendering quality while yielding fewer Gaussians on average than the alternatives.

For the capacity-control parameters (α, β), the default configuration $(0.8, 1.5)$ consistently produces the most compact representations across almost all scenes. Compared with both smaller and larger parameter settings, it substantially reduces the average storage and Gaussian count while preserving the best average PSNR and SSIM. This suggests that moderate capacity regulation is important: overly conservative control may retain redundant primitives, whereas overly aggressive densification-pruning schedules can increase representation size without clear quality gains.

Finally, varying the lifecycle-aware pruning rate leads to only minor changes in reconstruction quality. The average metrics remain very close across different rates, demonstrating that the pruning schedule is robust within a reasonable range. Although a smaller rate can slightly reduce the average model size, the default rate achieves the best overall PSNR, SSIM, and LPIPS, indicating a favorable balance between compactness and fidelity. Overall, these results show that ACD-GS is robust to hyperparameter variations and that the chosen default configuration provides a stable quality-efficiency trade-off across diverse scenes.

Table A1: Quality and efficiency under different depth soft thresholds.

Scene	Setting	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓	Points(M)↓
Aquarium-20	$\tau = 0.05$	0.8722	26.16	0.1673	52.04	0.2777
	$\tau = 0.15$	**0.8738**	**26.42**	**0.1655**	**41.76**	**0.1766**
	$\tau = 0.25$	0.8726	26.03	0.1656	51.73	0.2764
Bedroom	$\tau = 0.05$	0.9070	29.00	**0.2897**	121.89	0.5731
	$\tau = 0.15$	0.9073	29.04	0.2899	**111.74**	**0.4724**
	$\tau = 0.25$	**0.9076**	**29.04**	0.2903	121.59	0.5718
Boats	$\tau = 0.05$	0.9101	26.41	0.2209	167.23	0.7648
	$\tau = 0.15$	0.9087	26.15	0.2213	**158.20**	**0.6689**
	$\tau = 0.25$	**0.9102**	**26.47**	**0.2197**	163.70	0.7499
Bridge	$\tau = 0.05$	0.7873	24.09	**0.3102**	163.37	0.7484
	$\tau = 0.15$	**0.7904**	**24.18**	0.3104	**149.19**	**0.6308**
	$\tau = 0.25$	0.7897	23.91	0.3103	161.57	0.7409
CreepyAttic	$\tau = 0.05$	0.8903	30.62	0.2457	177.24	0.8071
	$\tau = 0.15$	**0.8908**	**30.70**	**0.2442**	**165.98**	**0.7018**
	$\tau = 0.25$	0.8904	30.65	0.2453	175.90	0.8014
Hugo-1	$\tau = 0.05$	0.8464	**24.53**	0.1613	124.61	0.5846
	$\tau = 0.15$	**0.8468**	24.45	**0.1608**	**110.88**	**0.4688**
	$\tau = 0.25$	0.8423	23.92	0.1637	122.62	0.5762
Library	$\tau = 0.05$	0.7814	23.98	0.3597	254.96	1.1357
	$\tau = 0.15$	0.7798	**24.10**	0.3592	**245.24**	**1.0369**
	$\tau = 0.25$	**0.7820**	24.10	**0.3592**	256.48	1.1421
Lumber	$\tau = 0.05$	0.7842	22.28	0.2653	508.99	2.2098
	$\tau = 0.15$	**0.7867**	**22.46**	**0.2634**	495.92	2.0968
	$\tau = 0.25$	0.7843	22.34	0.2661	**466.31**	**2.0293**
Museum-1	$\tau = 0.05$	0.8445	**23.64**	0.1469	**198.21**	**0.8958**
	$\tau = 0.15$	**0.8450**	23.49	**0.1452**	223.28	0.9440
	$\tau = 0.25$	0.8442	23.45	0.1456	234.51	1.0493
Museum-2	$\tau = 0.05$	**0.9013**	**28.07**	0.1065	**108.91**	0.5182
	$\tau = 0.15$	0.9002	27.61	**0.1056**	121.68	**0.5145**
	$\tau = 0.25$	0.8987	27.55	0.1066	132.65	0.6186

Continued on next page

Table A1 (continued)

Scene	Setting	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓	Points(M)↓
NightSnow	$\tau = 0.05$	0.9055	30.16	0.2242	**61.68**	0.3185
	$\tau = 0.15$	0.9054	30.22	0.2200	64.62	**0.2732**
	$\tau = 0.25$	**0.9062**	**30.33**	**0.2199**	73.96	0.3704
Playroom	$\tau = 0.05$	0.8949	29.67	0.2921	**112.51**	**0.5334**
	$\tau = 0.15$	**0.8960**	**30.07**	**0.2883**	129.12	0.5459
	$\tau = 0.25$	0.8952	29.92	0.2888	137.39	0.6386
Ponche	$\tau = 0.05$	0.8211	23.11	0.2558	**103.84**	0.4968
	$\tau = 0.15$	0.8191	23.07	0.2537	114.83	**0.4855**
	$\tau = 0.25$	**0.8234**	**23.24**	**0.2494**	123.42	0.5796
SaintAnne	$\tau = 0.05$	0.8195	**25.60**	0.3372	**135.70**	**0.6315**
	$\tau = 0.15$	**0.8203**	25.51	0.3309	160.94	0.6805
	$\tau = 0.25$	0.8189	25.34	**0.3309**	171.86	0.7844
Shed	$\tau = 0.05$	0.9338	28.32	0.1956	**96.72**	0.4666
	$\tau = 0.15$	0.9295	27.28	0.1963	107.75	**0.4556**
	$\tau = 0.25$	**0.9347**	**28.39**	**0.1912**	117.14	0.5530
Tree-18	$\tau = 0.05$	0.8027	22.18	0.1747	**81.61**	0.4028
	$\tau = 0.15$	**0.8052**	22.22	**0.1650**	84.78	**0.3584**
	$\tau = 0.25$	0.8044	**22.33**	0.1658	95.18	0.4602
Yellowhouse-12	$\tau = 0.05$	0.6536	17.58	0.3066	**90.42**	0.4400
	$\tau = 0.15$	**0.6608**	**18.09**	0.2975	95.58	**0.4041**
	$\tau = 0.25$	0.6485	17.06	**0.2972**	109.54	0.5209
Mean	$\tau = 0.05$	0.8445	**25.61**	0.2388	**150.58**	0.6944
	$\tau = 0.15$	**0.8451**	25.59	0.2363	151.85	**0.6420**
	$\tau = 0.25$	0.8113	25.53	**0.2362**	151.43	0.7331

Table A2: Quality and efficiency under different settings of α and β.

Scene	Setting	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓	Points(M)↓
Aquarium-20	$(0.5, 1.0)$	0.8517	23.38	0.1751	132.12	0.5586
	$(0.8, 1.5)$	**0.8738**	**26.42**	**0.1655**	**41.76**	**0.1766**
	$(1.0, 2.5)$	0.8486	24.40	0.1760	140.75	0.5951
Bedroom	$(0.5, 1.0)$	0.9068	28.87	0.2729	185.58	0.7847
	$(0.8, 1.5)$	**0.9073**	**29.04**	0.2899	**111.74**	**0.4724**
	$(1.0, 2.5)$	0.9072	28.92	**0.2719**	197.59	0.8355
Boats	$(0.5, 1.0)$	0.9122	26.57	0.2074	224.80	0.9505
	$(0.8, 1.5)$	0.9087	26.15	0.2213	**158.20**	**0.6689**
	$(1.0, 2.5)$	**0.9136**	**26.72**	**0.2020**	238.70	1.0092
Bridge	$(0.5, 1.0)$	0.7830	23.86	0.3175	152.05	0.6429
	$(0.8, 1.5)$	**0.7904**	**24.18**	**0.3104**	**149.19**	**0.6308**
	$(1.0, 2.5)$	0.7823	23.41	0.3147	168.09	0.7107
CreepyAttic	$(0.5, 1.0)$	0.8910	30.64	0.2176	463.91	1.9615
	$(0.8, 1.5)$	0.8908	**30.70**	0.2442	**165.98**	**0.7018**
	$(1.0, 2.5)$	**0.8910**	30.55	**0.2166**	483.30	2.0435
Hugo-1	$(0.5, 1.0)$	0.8319	23.87	**0.1498**	256.77	1.0857
	$(0.8, 1.5)$	**0.8468**	**24.45**	0.1608	**110.88**	**0.4688**
	$(1.0, 2.5)$	0.8276	23.61	0.1504	312.31	1.3205
Library	$(0.5, 1.0)$	0.7722	23.84	0.3580	371.66	1.5714
	$(0.8, 1.5)$	**0.7798**	**24.10**	0.3592	**245.24**	**1.0369**
	$(1.0, 2.5)$	0.7705	23.86	**0.3566**	405.50	1.7145
Lumber	$(0.5, 1.0)$	0.7899	**22.63**	0.2479	560.72	2.3708
	$(0.8, 1.5)$	0.7867	22.46	0.2634	**495.92**	**2.0968**
	$(1.0, 2.5)$	**0.7924**	22.56	**0.2364**	681.21	2.8802

Continued on next page

Table A2 (continued)

Scene	Setting	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓	Points(M)↓
Museum-1	$(0.5, 1.0)$	0.8348	22.81	0.1373	395.33	1.6715
	$(0.8, 1.5)$	**0.8450**	**23.49**	0.1452	**223.28**	**0.9440**
	$(1.0, 2.5)$	0.8354	22.82	**0.1359**	414.99	1.7546
Museum-2	$(0.5, 1.0)$	0.9016	26.62	0.0925	212.18	0.8971
	$(0.8, 1.5)$	0.9002	**27.61**	0.1056	**121.68**	**0.5145**
	$(1.0, 2.5)$	**0.9022**	27.15	**0.0909**	228.20	0.9648
NightSnow	$(0.5, 1.0)$	0.9147	30.49	0.1912	94.44	0.3993
	$(0.8, 1.5)$	0.9054	30.22	0.2200	**64.62**	**0.2732**
	$(1.0, 2.5)$	**0.9151**	**30.51**	**0.1888**	101.40	0.4287
Playroom	$(0.5, 1.0)$	**0.8960**	**30.18**	0.2675	287.29	1.2147
	$(0.8, 1.5)$	0.8960	30.07	0.2883	**129.12**	**0.5459**
	$(1.0, 2.5)$	0.8954	30.07	**0.2671**	300.31	1.2697
Ponche	$(0.5, 1.0)$	**0.8194**	22.54	**0.2289**	181.75	0.7684
	$(0.8, 1.5)$	0.8191	**23.07**	0.2537	**114.83**	**0.4855**
	$(1.0, 2.5)$	0.8148	21.89	0.2291	196.96	0.8328
SaintAnne	$(0.5, 1.0)$	0.8225	25.61	0.3109	199.34	0.8428
	$(0.8, 1.5)$	0.8203	25.51	0.3309	**160.94**	**0.6805**
	$(1.0, 2.5)$	**0.8236**	**25.68**	**0.2955**	238.51	1.0084
Shed	$(0.5, 1.0)$	0.9484	32.21	0.1536	168.72	0.7134
	$(0.8, 1.5)$	0.9295	27.28	0.1963	**107.75**	**0.4556**
	$(1.0, 2.5)$	**0.9489**	**32.37**	**0.1510**	174.77	0.7389
Tree-18	$(0.5, 1.0)$	0.7722	20.85	0.2055	164.06	0.6937
	$(0.8, 1.5)$	**0.8052**	22.22	**0.1650**	**84.78**	**0.3584**
	$(1.0, 2.5)$	0.7873	**22.28**	0.1778	200.11	0.8461
Yellowhouse-12	$(0.5, 1.0)$	0.6240	15.33	0.3198	154.89	0.6549
	$(0.8, 1.5)$	**0.6608**	**18.09**	**0.2975**	**95.58**	**0.4041**
	$(1.0, 2.5)$	0.6397	17.16	0.3077	145.98	0.6172
Mean	$(0.5, 1.0)$	0.8395	25.31	0.2267	247.39	1.0460
	$(0.8, 1.5)$	**0.8451**	**25.59**	0.2363	**151.85**	**0.6420**
	$(1.0, 2.5)$	0.8409	25.53	**0.2217**	272.28	1.1512

Table A3: Quality and efficiency under different opacity-pruning rates.

Scene	Setting	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓	Points(M)↓
Aquarium-20	Rate $= 0.5$	**0.8739**	26.20	0.1652	**40.21**	**0.1700**
	Ours (Rate $= 1.0$)	0.8738	**26.42**	0.1655	41.76	0.1766
	Rate $= 1.5$	0.8729	26.28	**0.1647**	42.01	0.1776
Bedroom	Rate $= 0.5$	0.9078	**29.10**	0.2898	**111.66**	**0.4721**
	Ours (Rate $= 1.0$)	0.9073	29.04	0.2899	111.74	0.4724
	Rate $= 1.5$	**0.9084**	29.07	**0.2891**	111.76	0.4725
Boats	Rate $= 0.5$	**0.9105**	**26.37**	**0.2207**	158.79	0.6714
	Ours (Rate $= 1.0$)	0.9087	26.15	0.2213	158.20	0.6689
	Rate $= 1.5$	0.9093	26.25	0.2222	**153.99**	**0.6511**
Bridge	Rate $= 0.5$	0.7887	24.16	0.3110	153.40	0.6486
	Ours (Rate $= 1.0$)	**0.7904**	**24.18**	**0.3104**	**149.19**	**0.6308**
	Rate $= 1.5$	0.7888	23.89	0.3107	150.39	0.6358
CreepyAttic	Rate $= 0.5$	0.8902	30.60	0.2449	167.08	0.7064
	Ours (Rate $= 1.0$)	**0.8908**	30.70	**0.2442**	**165.98**	**0.7018**
	Rate $= 1.5$	0.8902	**30.71**	0.2455	167.41	0.7078
Hugo-1	Rate $= 0.5$	0.8454	24.51	0.1621	111.38	0.4709
	Ours (Rate $= 1.0$)	**0.8468**	24.45	**0.1608**	**110.88**	**0.4688**
	Rate $= 1.5$	0.8451	**24.65**	0.1620	114.41	0.4837

Continued on next page

Table A3 (continued)

Scene	Setting	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓	Points(M)↓
Library	Rate = 0.5	0.7775	23.98	0.3600	**244.56**	**1.0340**
	Ours (Rate = 1.0)	0.7798	**24.10**	**0.3592**	245.24	1.0369
	Rate = 1.5	**0.7818**	23.96	0.3601	247.66	1.0471
Lumber	Rate = 0.5	0.7775	21.81	0.2704	**469.24**	**1.9840**
	Ours (Rate = 1.0)	**0.7867**	**22.46**	**0.2634**	495.92	2.0968
	Rate = 1.5	0.7808	21.99	0.2670	480.98	2.0336
Museum-1	Rate = 0.5	0.8435	23.41	0.1469	223.95	0.9469
	Ours (Rate = 1.0)	**0.8450**	23.49	**0.1452**	**223.28**	**0.9440**
	Rate = 1.5	0.8443	**23.55**	0.1457	223.91	0.9467
Museum-2	Rate = 0.5	0.8996	27.56	0.1061	**120.31**	**0.5087**
	Ours (Rate = 1.0)	**0.9002**	**27.61**	**0.1056**	121.68	0.5145
	Rate = 1.5	0.8978	27.58	0.1082	120.43	0.5092
NightSnow	Rate = 0.5	0.9034	29.98	0.2216	**63.16**	**0.2671**
	Ours (Rate = 1.0)	**0.9054**	30.22	0.2200	64.62	0.2732
	Rate = 1.5	0.9053	**30.45**	**0.2199**	65.45	0.2767
Playroom	Rate = 0.5	**0.8962**	**30.12**	0.2888	**127.31**	**0.5383**
	Ours (Rate = 1.0)	0.8960	30.07	**0.2883**	129.12	0.5459
	Rate = 1.5	0.8957	29.93	0.2891	129.34	0.5469
Ponche	Rate = 0.5	**0.8236**	**23.22**	**0.2491**	**114.69**	**0.4849**
	Ours (Rate = 1.0)	0.8191	23.07	0.2537	114.83	0.4855
	Rate = 1.5	0.8223	23.09	0.2505	116.27	0.4916
SaintAnne	Rate = 0.5	0.8209	**25.74**	**0.3277**	168.31	0.7116
	Ours (Rate = 1.0)	0.8203	25.51	0.3309	**160.94**	**0.6805**
	Rate = 1.5	**0.8216**	25.66	0.3284	161.27	0.6819
Shed	Rate = 0.5	**0.9320**	**27.67**	**0.1943**	106.29	0.4494
	Ours (Rate = 1.0)	0.9295	27.28	0.1963	107.75	0.4556
	Rate = 1.5	0.9273	27.41	0.1999	**104.45**	**0.4416**
Tree-18	Rate = 0.5	**0.8056**	**22.28**	0.1656	87.19	0.3686
	Ours (Rate = 1.0)	0.8052	22.22	**0.1650**	**84.78**	**0.3584**
	Rate = 1.5	0.8028	22.24	0.1678	92.55	0.3913
Yellowhouse-12	Rate = 0.5	0.6540	17.82	0.3058	96.52	0.4081
	Ours (Rate = 1.0)	**0.6608**	**18.09**	**0.2975**	**95.58**	**0.4041**
	Rate = 1.5	0.6537	17.65	0.3009	97.89	0.4139
Mean	Rate = 0.5	0.8441	25.56	0.2371	**150.83**	**0.6377**
	Ours (Rate = 1.0)	**0.8451**	**25.59**	**0.2363**	151.85	0.6420
	Rate = 1.5	0.8440	25.55	0.2372	151.77	0.6417

B Detailed Results

We provide detailed per-scene results on three benchmark datasets in the appendix. Specifically, Tables A4, A5, and A6 report per-scene rendering quality together with model size, Gaussian count, and training time for Deep Blending, Mip-NeRF 360, and Tanks & Temples, respectively.

Table A4: Per-scene quality and efficiency on Deep Blending.

Scene	Method	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓	Points(M)↓	Time(s)↓
Aquarium-20	AbsGS	0.8060	21.82	0.2288	183.16	0.7700	828
	3DGS	0.8692	22.66	0.1607	177.44	0.7500	942
	Mip-Splatting	0.8507	22.52	0.1764	272.31	1.1300	1034
	Pixel-GS	0.8653	25.77	**0.1582**	268.57	1.1400	900
	GDAGS	0.8418	22.18	0.1865	159.86	0.7900	743
	Mini-Splatting	0.8509	23.81	0.1659	810.17	3.4300	1329
	ACD-GS	**0.8738**	**26.42**	0.1655	**41.76**	**0.1800**	**649**

Continued on next page

Table A4 (continued)

Scene	Method	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓	Points(M)↓	Time(s)↓
Bedroom	AbsGS	0.9012	28.85	0.2751	253.97	1.0700	854
	3DGS	0.9072	28.86	0.2743	316.93	1.3400	691
	Mip-Splatting	0.9045	28.68	0.2744	473.69	1.9700	791
	Pixel-GS	0.9041	28.82	0.2748	719.19	3.0400	834
	GDAGS	0.9070	**29.05**	**0.2701**	283.22	1.2200	476
	Mini-Splatting	0.9007	28.28	0.2955	981.14	4.1500	976
	ACD-GS	**0.9073**	29.04	0.2899	**111.74**	**0.4700**	**405**
Boats	AbsGS	0.9036	26.61	0.2162	399.18	1.6900	737
	3DGS	0.9122	26.68	0.2099	389.69	1.6500	766
	Mip-Splatting	**0.9159**	26.75	**0.1934**	629.55	2.6200	930
	Pixel-GS	0.9145	**26.98**	0.1967	846.92	3.5800	975
	GDAGS	0.9137	26.67	0.2018	344.09	1.4800	568
	Mini-Splatting	0.9054	25.73	0.2133	1063.13	4.5000	1111
	ACD-GS	0.9087	26.15	0.2213	**158.20**	**0.6700**	**509**
Bridge	AbsGS	0.7694	22.65	0.3123	245.33	1.0400	701
	3DGS	**0.8103**	25.13	0.2884	259.22	1.1000	817
	Mip-Splatting	0.8062	24.41	**0.2848**	332.83	1.3800	921
	Pixel-GS	0.8056	**25.77**	0.3049	305.77	1.2900	780
	GDAGS	0.7834	23.59	0.3078	232.13	0.9900	593
	Mini-Splatting	0.7301	21.32	0.4274	952.32	4.0300	1269
	ACD-GS	0.7904	24.18	0.3104	**149.19**	**0.6300**	**586**
CreepyAttic	AbsGS	0.8821	30.16	0.2199	704.01	2.9800	829
	3DGS	**0.8924**	30.52	0.2245	758.95	3.2100	968
	Mip-Splatting	0.8860	30.16	0.2208	1142.25	4.7500	1235
	Pixel-GS	0.8841	30.24	0.2216	1558.26	6.5900	1329
	GDAGS	0.8890	30.43	**0.2138**	711.70	2.9400	596
	Mini-Splatting	0.8682	29.41	0.2800	1172.18	4.9600	1005
	ACD-GS	0.8908	**30.70**	0.2442	**165.98**	**0.7000**	**422**
Hugo-1	AbsGS	0.7891	22.58	0.1804	344.76	1.4600	811
	3DGS	0.8429	23.85	0.1394	321.95	1.3600	830
	Mip-Splatting	0.8304	22.80	0.1486	494.73	2.0600	1030
	Pixel-GS	0.8355	23.43	**0.1393**	496.32	2.1000	923
	GDAGS	0.8208	23.39	0.1597	318.72	1.4800	699
	Mini-Splatting	0.8412	23.24	0.1433	873.89	3.6900	1075
	ACD-GS	**0.8468**	**24.45**	0.1608	**110.88**	**0.4700**	**574**
Library	AbsGS	0.7311	23.49	0.4162	738.66	3.1200	991
	3DGS	**0.7806**	23.87	**0.3520**	501.54	2.1200	900
	Mip-Splatting	0.7629	23.32	0.3576	829.34	3.4500	1144
	Pixel-GS	0.7767	23.73	0.3577	998.51	4.2200	1099
	GDAGS	0.7682	23.72	0.3570	568.79	2.5100	682
	Mini-Splatting	0.7616	23.33	0.3941	1129.81	4.7800	1216
	ACD-GS	0.7798	**24.10**	0.3592	**245.24**	**1.0400**	**570**
Lumber	AbsGS	0.7691	22.04	0.2559	1163.96	4.9200	1188
	3DGS	0.7705	22.03	0.2787	861.48	3.6400	1042
	Mip-Splatting	**0.7947**	22.61	**0.2381**	1411.98	5.8800	1442
	Pixel-GS	0.7777	22.24	0.2638	1549.63	6.5500	1414
	GDAGS	0.7907	**22.67**	0.2445	921.82	3.9700	755
	Mini-Splatting	0.7576	21.77	0.2694	1275.84	5.3900	1208
	ACD-GS	0.7867	22.46	0.2634	**495.92**	**2.1000**	623
Museum-1	AbsGS	0.7931	21.73	0.1689	568.56	2.4000	1005
	3DGS	0.8364	23.32	0.1346	567.26	2.4000	1272
	Mip-Splatting	0.8363	23.28	0.1354	807.78	3.3600	1451
	Pixel-GS	0.8346	23.15	**0.1340**	841.11	3.5600	1321
	GDAGS	0.8339	22.69	0.1384	554.20	2.6700	829
	Mini-Splatting	**0.8451**	23.20	0.1351	938.93	3.9700	1207
	ACD-GS	0.8450	**23.49**	0.1452	**223.28**	**0.9400**	691
Museum-2	AbsGS	0.8798	25.48	0.1109	310.55	1.3100	782
	3DGS	0.8970	25.34	0.0985	333.72	1.4100	904
	Mip-Splatting	0.8964	25.81	0.0976	477.92	1.9900	1045
	Pixel-GS	0.8982	26.58	0.0932	561.85	2.3800	1006
	GDAGS	0.8988	25.64	0.0951	299.60	1.2900	676
	Mini-Splatting	**0.9053**	27.50	**0.0891**	877.42	3.9700	1140
	ACD-GS	0.9002	**27.61**	0.1056	**121.68**	**0.5100**	**557**

Continued on next page

Table A4 (continued)

Scene	Method	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓	Points(M)↓	Time(s)↓
NightSnow	AbsGS	0.9033	30.09	0.2016	145.95	0.6200	601
	3DGS	0.9095	30.53	0.1962	166.04	0.7000	625
	Mip-Splatting	**0.9160**	30.37	**0.1765**	251.49	1.0500	747
	Pixel-GS	0.9119	30.13	0.1925	285.68	1.2100	658
	GDAGS	0.9153	**30.63**	0.1857	131.93	0.5400	517
	Mini-Splatting	0.9080	29.56	0.1901	877.21	3.7100	1113
	ACD-GS	0.9054	30.22	0.2200	**64.62**	**0.2700**	**465**
Playroom	AbsGS	0.8893	30.00	0.2644	405.12	1.7100	679
	3DGS	0.8919	29.99	0.2763	499.96	2.1100	856
	Mip-Splatting	0.8906	29.68	0.2688	797.40	3.3200	1044
	Pixel-GS	0.8892	29.62	0.2763	959.15	4.0600	995
	GDAGS	0.8949	**30.22**	**0.2634**	444.93	1.9200	544
	Mini-Splatting	0.8858	29.59	0.2979	1080.89	4.5700	962
	ACD-GS	**0.8960**	30.07	0.2883	**129.12**	**0.5500**	**432**
Ponche	AbsGS	0.8000	22.31	0.2376	254.42	1.0800	690
	3DGS	0.8204	22.25	0.2464	205.92	0.8700	672
	Mip-Splatting	0.8201	21.80	0.2250	337.44	1.4000	842
	Pixel-GS	0.8123	21.57	0.2360	404.74	1.7100	834
	GDAGS	0.8148	22.09	0.2333	225.64	0.9800	589
	Mini-Splatting	**0.8341**	22.86	**0.2067**	848.29	3.5900	1094
	ACD-GS	0.8191	**23.07**	0.2537	**114.83**	**0.4900**	**543**
SaintAnne	AbsGS	0.8054	25.09	0.3259	286.99	1.2100	704
	3DGS	0.8208	25.60	0.3398	171.70	0.7300	814
	Mip-Splatting	0.8216	25.27	0.3238	261.31	1.0900	819
	Pixel-GS	**0.8251**	**25.67**	0.3082	757.02	3.2000	1055
	GDAGS	0.8232	25.58	**0.3078**	252.59	1.2100	593
	Mini-Splatting	0.8177	25.44	0.3163	893.32	3.7800	1160
	ACD-GS	0.8203	25.51	0.3309	**160.94**	**0.6800**	**576**
Shed	AbsGS	0.9420	32.32	0.1593	349.88	1.4800	713
	3DGS	0.9474	32.22	0.1657	312.16	1.3200	735
	Mip-Splatting	0.9489	32.32	**0.1452**	526.94	2.1900	912
	Pixel-GS	**0.9507**	**32.58**	0.1487	664.38	2.8100	906
	GDAGS	0.9501	32.34	0.1474	307.70	1.1800	563
	Mini-Splatting	0.9384	30.76	0.1547	983.32	4.1600	1126
	ACD-GS	0.9295	27.28	0.1963	**107.75**	**0.4600**	**490**
Tree-18	AbsGS	0.7458	21.03	0.2105	216.07	0.9100	751
	3DGS	0.7974	20.75	0.1695	223.22	0.9400	838
	Mip-Splatting	0.7887	21.06	0.1765	289.15	1.2000	946
	Pixel-GS	0.7958	21.76	0.1668	327.29	1.3800	884
	GDAGS	0.7705	21.26	0.2049	192.28	0.8800	679
	Mini-Splatting	0.8036	21.98	0.1714	798.00	3.3700	1167
	ACD-GS	**0.8052**	**22.22**	**0.1650**	**84.78**	**0.3600**	**639**
Yellowhouse-12	AbsGS	0.5812	16.29	0.3517	236.54	1.0000	655
	3DGS	0.6554	17.11	0.2949	176.08	0.7400	642
	Mip-Splatting	0.6414	16.75	0.3157	238.22	0.9900	780
	Pixel-GS	**0.6670**	17.61	**0.2816**	267.99	1.1300	709
	GDAGS	0.6371	16.63	0.3081	180.94	0.8500	575
	Mini-Splatting	0.6394	16.27	0.3155	749.61	4.1600	970
	ACD-GS	0.6608	**18.09**	0.2975	**95.58**	**0.4000**	**537**
Mean	AbsGS	0.8171	24.86	0.2433	400.42	1.6900	795
	3DGS	0.8448	25.34	0.2265	367.25	1.5500	842
	Mip-Splatting	0.8418	25.15	0.2211	563.20	2.3400	1006
	Pixel-GS	0.8440	**25.63**	**0.2208**	694.85	2.9400	977
	GDAGS	0.8384	25.22	0.2250	360.60	1.5800	628
	Mini-Splatting	0.8349	24.94	0.2392	959.15	3.3700	1125
	ACD-GS	**0.8451**	25.59	0.2363	**151.85**	**0.6400**	**545**

Table A5: Per-scene quality and efficiency on Mip-NeRF 360.

Scene	Method	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓	Points(M)↓	Time(s)↓
Bicycle	AbsGS	0.7817	25.26	0.1720	1411.36	5.9700	1084
	3DGS	0.7666	25.28	0.2081	1154.55	4.8800	1084
	Mip-Splatting	**0.7924**	**25.55**	**0.1669**	1977.66	8.2300	1674
	Pixel-GS	0.7817	25.26	0.1720	2142.21	9.0600	1639
	GDAGS	0.7899	25.42	0.1696	1437.29	6.0800	807
	Mini-Splatting	0.7165	23.57	0.2508	1425.72	6.0300	1031
	ACD-GS	0.7861	25.36	0.1799	**543.99**	**2.3000**	**597**
Bonsai	AbsGS	0.9575	32.67	0.0847	198.66	0.8400	360
	3DGS	0.9434	31.54	0.0964	253.60	1.0700	360
	Mip-Splatting	**0.9588**	**32.95**	**0.0819**	385.98	1.6100	485
	Pixel-GS	0.9575	32.67	0.0847	481.79	2.0400	499
	GDAGS	0.9552	32.22	0.0901	184.95	0.7800	261
	Mini-Splatting	0.9353	29.05	0.1055	882.88	3.7300	715
	ACD-GS	0.9455	31.07	0.1143	**76.09**	**0.3200**	**249**
Counter	AbsGS	0.9216	29.35	0.1073	190.66	0.8100	411
	3DGS	0.9212	28.43	0.1121	235.90	1.0000	411
	Mip-Splatting	**0.9249**	**29.62**	**0.1042**	343.31	1.4300	494
	Pixel-GS	0.9216	29.35	0.1073	574.00	2.4300	613
	GDAGS	0.9174	28.95	0.1136	188.99	0.8000	283
	Mini-Splatting	0.8804	26.38	0.1465	880.19	3.7200	770
	ACD-GS	0.9127	28.83	0.1251	**69.44**	**0.2900**	**254**
Flowers	AbsGS	0.6210	21.26	**0.2706**	907.98	3.8400	732
	3DGS	0.6044	21.50	0.3366	686.30	2.9000	732
	Mip-Splatting	**0.6404**	**21.68**	0.2732	1104.81	4.6000	1044
	Pixel-GS	0.6210	21.26	**0.2706**	1774.43	7.5000	1433
	GDAGS	0.6224	21.34	0.2746	878.11	3.7100	622
	Mini-Splatting	0.5724	20.85	0.3516	1152.61	4.8700	967
	ACD-GS	0.6199	21.32	0.2811	**379.47**	**1.6000**	**530**
Garden	AbsGS	0.8716	27.55	0.0991	896.76	3.7900	1003
	3DGS	0.8680	27.45	0.1063	981.85	4.1500	1003
	Mip-Splatting	**0.8772**	**27.66**	**0.0945**	1499.50	6.2400	1472
	Pixel-GS	0.8716	27.55	0.0991	2075.27	8.7700	1654
	GDAGS	0.8715	27.44	0.1006	1042.35	4.4100	767
	Mini-Splatting	0.8300	25.73	0.1476	1375.65	5.8200	1136
	ACD-GS	0.8689	27.49	0.1056	**461.87**	**1.9500**	**613**
Kitchen	AbsGS	0.9495	**32.58**	0.0652	245.16	1.0400	483
	3DGS	0.9488	32.25	0.0667	363.99	1.5400	483
	Mip-Splatting	**0.9498**	32.26	**0.0638**	492.76	2.0500	589
	Pixel-GS	0.9495	**32.58**	0.0652	720.13	3.0400	710
	GDAGS	0.9483	32.12	0.0676	249.48	1.0500	321
	Mini-Splatting	0.9208	28.93	0.0969	864.55	3.6600	791
	ACD-GS	0.9359	30.59	0.0843	**84.22**	**0.3600**	**258**
Room	AbsGS	0.9448	32.23	0.1068	268.59	1.1400	400
	3DGS	0.9343	32.40	0.1118	272.45	1.1500	400
	Mip-Splatting	**0.9464**	**32.49**	**0.1040**	452.25	1.8800	509
	Pixel-GS	0.9448	32.23	0.1068	581.21	2.4600	542
	GDAGS	0.9431	31.97	0.1097	262.01	1.1100	275
	Mini-Splatting	0.9238	29.64	0.1347	966.48	4.0900	730
	ACD-GS	0.9385	31.87	0.1230	**62.42**	**0.2600**	**227**
Stump	AbsGS	0.7768	26.66	0.1967	1106.18	4.6800	869
	3DGS	0.7714	26.66	0.2163	1026.20	4.3400	869
	Mip-Splatting	0.7902	26.89	0.1883	1440.26	5.9900	1218
	Pixel-GS	0.7768	26.66	0.1967	1618.47	6.8400	1262
	GDAGS	**0.7943**	**26.96**	**0.1874**	949.72	4.0200	613
	Mini-Splatting	0.7601	26.03	0.2282	1274.58	5.3900	949
	ACD-GS	0.7830	26.83	0.2051	**318.65**	**1.3500**	**455**
Treehill	AbsGS	0.6139	21.96	0.2795	1221.96	5.1700	807
	3DGS	0.6335	**22.49**	0.3251	770.88	3.2600	807
	Mip-Splatting	**0.6387**	22.34	**0.2744**	1223.14	5.0900	1118
	Pixel-GS	0.6139	21.96	0.2795	1868.62	7.9000	1396
	GDAGS	0.6246	22.06	0.2786	1171.14	4.9500	700
	Mini-Splatting	0.6244	22.05	0.3254	1154.33	4.8800	944
	ACD-GS	0.6253	21.79	0.2936	**472.37**	**2.0000**	**552**

Continued on next page

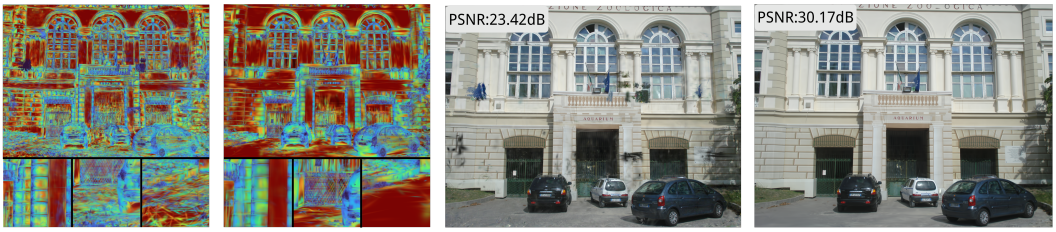
Table A5 (continued)

Scene	Method	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓	Points(M)↓	Time(s)↓
Mean	AbsGS	0.8265	27.72	0.1535	716.37	3.0300	683
	3DGS	0.8213	27.56	0.1755	638.41	2.7000	683
	Mip-Splatting	**0.8354**	**27.94**	**0.1501**	991.07	4.1200	955
	Pixel-GS	0.8265	27.72	0.1535	1315.13	5.5600	1083
	GDAGS	0.8296	27.61	0.1546	764.32	2.9900	516
	Mini-Splatting	0.7960	25.80	0.1986	1108.55	4.6900	892
	ACD-GS	0.8240	27.24	0.1680	**274.28**	**1.1600**	**415**

Table A6: Per-scene quality and efficiency on Tanks & Temples.

Scene	Method	SSIM↑	PSNR↑	LPIPS↓	Size(MB)↓	Points(M)↓	Time(s)↓
Train	AbsGS	0.8166	21.64	0.1942	230.86	0.9800	401
	3DGS	0.8202	22.04	0.1963	258.35	1.0900	510
	Mip-Splatting	0.8266	21.87	0.1893	354.45	1.4700	611
	Pixel-GS	**0.8279**	**22.13**	**0.1780**	906.22	3.8300	838
	GDAGS	0.8129	21.53	0.2068	223.96	0.9500	**326**
	Mini-Splatting	0.7099	18.77	0.3234	937.32	3.9600	855
	ACD-GS	0.8103	21.54	0.2298	**123.27**	**0.5200**	335
Truck	AbsGS	0.8874	25.71	0.1314	388.02	1.6400	464
	3DGS	0.8851	25.47	0.1418	487.82	2.0600	593
	Mip-Splatting	**0.8930**	**25.74**	0.1231	784.70	3.2700	820
	Pixel-GS	0.8868	25.48	**0.1203**	1228.62	5.1900	1053
	GDAGS	0.8858	25.49	0.1380	415.07	1.7500	374
	Mini-Splatting	0.8573	23.66	0.1583	1084.63	4.5900	859
	ACD-GS	0.8813	25.21	0.1434	**177.46**	**0.7500**	**331**
Mean	AbsGS	0.8520	23.68	0.1628	309.44	1.3100	432
	3DGS	0.8527	23.76	0.1691	373.09	1.5800	551
	Mip-Splatting	**0.8598**	23.80	0.1562	569.58	2.3700	715
	Pixel-GS	0.8574	**23.81**	**0.1492**	1067.42	4.5100	945
	GDAGS	0.8494	23.51	0.1724	319.51	1.3500	350
	Mini-Splatting	0.7836	21.22	0.2409	1010.98	4.2700	857
	ACD-GS	0.8458	23.37	0.1866	**150.37**	**0.6400**	**333**

C More Visual comparison



GDAGS ACD-GS GDAGS ACD-GS

Figure 5: Qualitative comparison of Gaussian distributions. Our curvature-aware strategy allocates sparse and elongated Gaussians to smooth surfaces, while preserving sharp boundaries with compact fine-scale primitives.

D Geometric Reconstruction

To verify whether the proposed geometry-aware densification preserves reconstructed geometry, we conduct an additional surface reconstruction evaluation. Specifically, we extract meshes from the reconstructed Gaussian representations and evaluate them using Precision, Recall, and F-score. The results are reported in Table A7.

Our method primarily aims to obtain a more compact Gaussian representation while maintaining comparable novel-view synthesis quality, without introducing additional geometric supervision or

Table A7: Geometric reconstruction evaluation using Precision, Recall, and F-score.

Method	Precision(%)↑	Recall(%)↑	F-score(%)↑	Points↓
ACD-GS	66.34	91.21	76.81	**642,044**
AbsGS	**69.58**	**93.33**	**79.72**	1,693,016
3DGS	60.52	79.08	68.57	1,552,777
GDAGS	66.13	88.80	75.81	1,582,823
Mini-Splatting	56.04	85.58	67.72	4,495,856
Mip-Splatting	66.02	88.60	78.03	2,343,461
Pixel-GS	68.16	86.41	76.21	2,937,890

explicit constraints on reconstruction accuracy. ACD-GS uses only approximately 38% and 41% of the Gaussian primitives required by AbsGS and GDAGS, respectively. Moreover, because ACD-GS tends to represent planar regions with fewer and larger Gaussians, TSDF-based surface extraction may be more susceptible to holes and local reconstruction errors. Nevertheless, ACD-GS still achieves competitive geometric reconstruction quality, indicating that curvature-aware densification preserves essential scene structures despite the substantially reduced Gaussian budget.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

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

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss the limitations of this work in a single section.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: The assumptions for each theoretical result are clearly stated.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We provide sufficient information to reproduce the main experimental results and the code will be released.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We provide the implementation code of our method and model checkpoints.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We specify all the training and test details.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: The paper does not include hypothesis testing or statistical significance analysis, and error bars are not a focus of the experiments.

   Guidelines:

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

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The computational resources are discussed in the Implementation section, providing sufficient information to reproduce the experiments.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research conducted in the paper confirms the NeurIPS code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: Our work mainly focuses on the improvement of algorithms and models, rather than areas directly related to social or ethical issues.

    Guidelines:

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

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All resources used in our paper, including code, data, and models, strictly adhere to appropriate licenses and terms of use, and have been appropriately authored by the original owner

    Guidelines:

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

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: The code is provided and includes accompanying documentation.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: This work does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

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

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

    Answer: [N/A]

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

    Guidelines:

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

16. **Declaration of LLM usage**

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

Answer:[N/A]

Justification: We use LLM for writing, editing, and formatting purposes. Coding Agents are only used to manage training tasks.

Guidelines:

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