GRADIENT-DRIVEN NATURAL SELECTION FOR COMPACT 3D GAUSSIAN SPLATTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Compact 3D Gaussian Splatting must reduce the number of primitives while preserving reconstruction quality. In a dense representation, overlapping Gaussians jointly support local details, so a member's current contribution may not reflect its value after other members leave and the scene is reoptimized. We introduce Gradient-Driven Natural Selection (GNS), which organizes competition through shrinkage and reconstruction-driven adaptation of native opacity, allowing necessary local support to concentrate on fewer members before pruning. GNS applies a shared linear penalty to pre-activation opacity parameters to reduce selection bias introduced by the shrinkage term itself. It also increases the opacity learning rate during selection so that members respond to reconstruction feedback as the representation contracts. The surviving subset emerges during reconstruction optimization without extra learnable selection variables or inference modules. With common dense inputs, training iterations, and target Gaussian budgets, these two simple changes achieve better overall reconstruction quality than the compared learned pruning baselines across multiple datasets.

1 INTRODUCTION

3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) uses millions of primitives for complex scenes, incurring substantial storage and rendering costs. Compact 3DGS seeks a budgeted subset that preserves reconstruction after further optimization. Many pruning methods rank Gaussians by current opacity, gradients, or visibility, treating retention value as an individual attribute of the dense representation. Yet overlapping Gaussians share local observations through alpha compositing. When one fades or leaves, transmittance, other members' compositing weights, and their reconstruction gradients change. Current individual contribution is therefore a snapshot; useful retention preferences can emerge as the representation contracts and adapts.

The variable carrying such competition should control rendering participation and receive reconstruction gradients. Learned pruning often introduces a separate mask, whereas native opacity already controls per-pixel alpha and compositing weight and responds to reconstruction loss. We propose Gradient-Driven Natural Selection (GNS) to organize selection in this native parameter space. Shared pressure contracts the representation, reconstruction feedback differentiates overlapping members, and alpha compositing reshapes their local roles before discrete deletion.

GNS follows two principles. *Competition neutrality* reduces preferences introduced by shrinkage itself: an L_1 penalty on activated opacity passes through a state-dependent sigmoid derivative, while our pre-activation linear penalty adds the same first-order gradient to every member before the optimizer. *Competition sufficiency* gives task-driven differences time to develop before irreversible deletion. Near-duplicate members may jointly support a detail yet receive weak, similar feedback; if deletion outruns adaptation, that detail can be lost. We raise the opacity learning rate during regularized pruning so that rendering participation responds faster and local support concentrates on fewer members (Figure 1).

With matched dense inputs, training iterations, and budgets, GNS attains better overall quality than the compared heuristic and learned pruning methods. Ablations show gains from both design changes. In a two-scene matched-count diagnostic, higher opacity learning rates accompany more opposite compositing-weight changes among paired members and corresponding opacity changes.

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

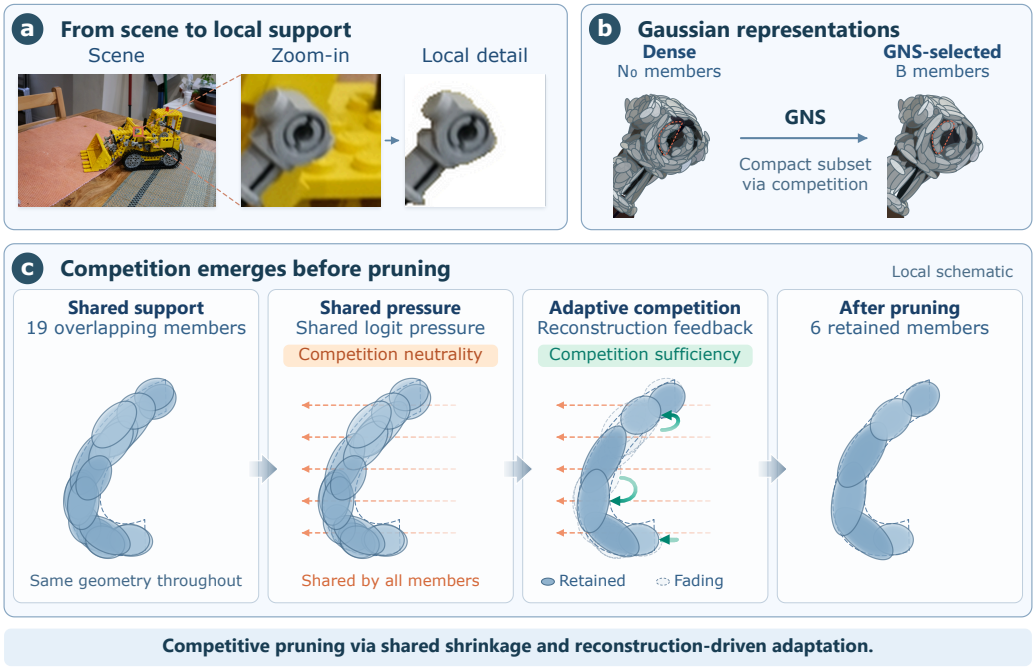

Figure 1: **GNS: competitive pruning through shared shrinkage and reconstruction-driven adaptation.** (a) A local detail in a real scene. (b) Selecting a compact subset of budget B from N_0 dense members. (c) In the local schematic, 19 overlapping members receive a shared pre-activation shrinkage gradient; reconstruction feedback differentiates their roles, and six survivors maintain local support after pruning. Competition neutrality corresponds to shared shrinkage input and competition sufficiency to adaptation before deletion. The fixed outline marks the shared support region, and green arrows indicate concentration of local support.

GNS also retains its advantage when the selected IDs are applied to the same original dense model and reoptimized under common conditions, supporting the quality of its surviving subset.

Our contributions are:

- **Design perspective.** We view local responsibility coupling in 3DGS through competition neutrality and sufficiency, using them to organize selection in native opacity parameters.
- **Simple implementation.** Shared pre-activation linear shrinkage and a higher opacity learning rate provide task-driven pruning without an additional learnable selection variable.
- **Mechanism and subset evidence.** Compositing-weight tracking and controlled ID replay support pre-deletion contribution differentiation and improved subset quality, alongside matched-budget quality gains across datasets.

2 Related Work

3D Gaussian Splatting representations and optimization. 3DGS (Kerbl et al., 2023) combines explicit Gaussians, adaptive densification, and differentiable rasterization. EDGS replaces densification with dense initialization (Kotovenko et al., 2026); AbsGS and ImprovedGS refine densification (Ye et al., 2024; Deng et al., 2026). Other work extends primitives and structured or dynamic representations (Huang et al., 2024; Held et al., 2025; Wu et al., 2024; Lu et al., 2024), and scales scenes with partitioning and hierarchy (Lin et al., 2024; Liu et al., 2025b; Kerbl et al., 2024). Other directions address sampling and visibility (Yu et al., 2024; Radl et al., 2024), efficiency (Mallick et al., 2024; Hanson et al., 2025a), and relighting (Liang et al., 2024; Gao et al., 2025). GNS focuses on pruning a detailed scene after densification.

Gaussian pruning and compression. HAC (Chen et al., 2024) compresses neural anchors, while SOGS, Compressed3DGS, EAGLES, and MesonGS reduce per-member storage through attribute organization or quantization (Morgenstern et al., 2025; Niedermayr et al., 2024; Girish et al., 2025; Xie et al., 2025). For member selection, LightGaussian estimates global contribution (Fan et al., 2024), PUP 3D-GS evaluates sensitivity to geometric perturbation (Hanson et al., 2025b), and Speedy-Splat scores rendering sensitivity (Hanson et al., 2025a). Trimming the Fat combines opacity and gradients (Ali et al., 2024); Mini-Splatting couples spatial reorganization with importance sampling (Fang & Wang, 2024); Prune Wisely adjusts pruning pace (Wang et al., 2026a). GHAP and NanoGS reduce or merge Gaussian mixtures (Wang et al., 2025; Xiong et al., 2026); Mobile-GS combines pruning, distillation, and quantization (Du et al., 2026). RAP predicts importance from local features (Yang et al., 2026a), and FlexGS couples budget-aware selection with parameter transformation (Liu et al., 2025a). GNS examines how shrinkage input and adaptation speed shape member differentiation before deletion.

Learned pruning and opacity sparsification. Compact-3DGS (Lee et al., 2024) learns a binary mask for opacity and scale, LP-3DGS (Zhang et al., 2024) uses Gumbel-Sigmoid masks, and MaskGaussian (Liu et al., 2025c) lets unsampled members update their existence probabilities. They learn retention through extra selection variables; GNS uses native opacity for both rendering participation and reconstruction feedback.

CompGS and ControlGS penalize activated opacity with L_1 (Navaneet et al., 2025; Zhang et al., 2025a), and GS^2 combines linear and quadratic opacity penalties with spatial adjustment (Yang et al., 2026b). 3DGS-MCMC and MH-3DGS regularize opacity and scale during densification or relocation (Kheradmand et al., 2024; Kim et al., 2025). ConeGS (Baranowski et al., 2026) penalizes pre-activation opacity during ongoing densification; its public implementation uses the same parameter-mean form as ours. GNS applies this term to a fixed post-densification candidate set, analyzes its shared gradient pressure, raises opacity learning rate for pre-deletion adaptation, and controls shrinkage by the budget boundary.

GaussianSpa (Zhang et al., 2025b) alternates reconstruction, projection of an auxiliary sparse opacity variable, and multiplier updates; SplatOpt (Wang et al., 2026b) combines related alternating optimization with geometry-aligned splitting. GNS retains native parameter updates in the reconstruction optimizer and coordinates shared shrinkage with opacity adaptation before deletion.

3 METHOD

GNS selects from a fixed post-densification candidate set of N_0 Gaussians under target budget B. Within a finite window, it retains the original rendering and reconstruction paths while shared shrinkage and faster opacity adaptation accompany gradual pruning.

3.1 COMPETITION DYNAMICS AND DESIGN PRINCIPLES

Each anisotropic 3D Gaussian has a center, covariance, color features, and opacity $\alpha_i = \sigma(z_i)$, where z_i is the pre-activation parameter. Projected Gaussians are alpha-composited in depth order; the color at pixel p is

$$\hat{\mathbf{C}}_p = \sum_i w_{i,p}\mathbf{c}_i + T_{\mathrm{bg},p}\mathbf{C}_{\mathrm{bg}}, \tag{1}$$

where $\mathbf{c}_i$ is view-dependent color, $\mathbf{C}_{\mathrm{bg}}$ is background color, and $T_{\mathrm{bg},p}$ is remaining transmittance. The compositing weight is

$$w_{i,p} = T_{i,p}\alpha_i\phi_{i,p}, \qquad T_{i,p} = \prod_{j\prec i}(1-\alpha_j\phi_{j,p}), \tag{2}$$

where $\phi_{i,p}$ is the projected 2D Gaussian response and $j \prec i$ denotes a member preceding i in compositing order. Opacity changes both a member's own weight and the participation of overlapping members behind it. Members explaining the same local observations therefore change one another's compositing conditions and reconstruction feedback.

Native opacity controls rendering participation and receives reconstruction gradients, allowing continuous selection as shared shrinkage contracts the representation. Two principles guide the shrinkage input and the adaptation of local responsibility.

Competition neutrality concerns the source of selection differences among members. The budget requires contraction, while retention value develops as members maintain or take over support during adaptation. Dense parameter states reflect an existing division of labor and optimization history; state-dependent shrinkage can impose extra exit preferences beyond reconstruction feedback. For near-duplicates, these preferences can decide exit order prematurely. Competition neutrality reduces member differences created by the penalty itself.

Competition sufficiency concerns how local representational needs become individual retention differences. Near-duplicates may jointly support a detail while each receives weak feedback from an already well-fitted local composite. As some fade, others can adjust participation under changed compositing and reconstruction feedback. Competition sufficiency requires this adaptation to keep pace with contraction so that necessary support concentrates before irreversible deletion.

3.2 Shared Linear Shrinkage Before Activation

Let S_t be the surviving index set at step t and $N_t = |S_t|$. With an L_1 penalty on activated opacity, the regularizer and its gradient with respect to the pre-activation parameter are

$$\mathcal{R}^t_{\mathrm{act}} = \frac{1}{N_t}\sum_{i\in S_t}\alpha^t_i, \qquad \frac{\partial(\lambda_t\mathcal{R}^t_{\mathrm{act}})}{\partial z^t_i} = \frac{\lambda_t}{N_t}\alpha^t_i(1-\alpha^t_i). \tag{3}$$

The sigmoid derivative makes the shrinkage input in the optimized variable depend on each member's current opacity.

GNS places the penalty directly in the pre-activation domain and optimizes at every step of the selection window

$$\mathcal{L}_t = \mathcal{L}_{\mathrm{rec}} + \lambda_t\frac{1}{N_t}\sum_{i\in S_t} z^t_i, \tag{4}$$

where $\mathcal{L}_{\mathrm{rec}}$ is the candidate model's reconstruction loss, $\lambda_t > 0$ controls shrinkage strength, and Appendix A gives the general configuration. Let $g^t_i = \partial\mathcal{L}_{\mathrm{rec}}/\partial z^t_i$ be the reconstruction gradient and $p_t = \lambda_t/N_t$ the shared per-member pressure. Then

$$\tilde{g}^t_i = g^t_i + p_t, \qquad \tilde{g}^t_i - \tilde{g}^t_j = g^t_i - g^t_j. \tag{5}$$

The linear term adds an equal first-order gradient to every pre-activation parameter, canceling in pairwise differences before the optimizer and removing the penalty's explicit activation-state preference.

3.3 Faster Adaptation Before Pruning

To let responsibility adjustment keep pace with structural contraction, GNS raises only the opacity parameter group's learning rate during the selection window:

$$\eta^{\mathrm{GNS},t}_z = s\eta^t_z, \qquad s > 1, \tag{6}$$

where η^t_z is the original learning rate and other parameter groups retain their settings. Let u^t_i be the effective update direction formed by the optimizer from the current total gradient and its history, and define relative state $d^t_{ij} = z^t_i - z^t_j$. The updates satisfy

$$z^{t+1}_i = z^t_i - s\eta^t_z u^t_i, \qquad d^{t+1}_{ij} - d^t_{ij} = -s\eta^t_z(u^t_i - u^t_j). \tag{7}$$

For given effective update directions, s enlarges the step's relative logit update. With position, scale, and color learning rates unchanged, opacity can respond faster to reconstruction feedback before discrete pruning, promoting concentration of local support. Section 4.4 and Section 4.5 examine its quality effect and the accompanying contribution-differentiation trend.

3.4 TARGET-BUDGET SHRINKAGE FEEDBACK

Under a fixed coefficient and selection window, differences in scene redundancy and candidate state change the natural decay rate. We let users specify budget B directly and adjust shrinkage through feedback, avoiding per-scene fixed-coefficient searches.

Let q_t be the B-th largest native opacity at step t, and define a linear reference trajectory for this budget boundary that decreases to zero from the start of the window. When q_t lies above the reference, we increase λ_t to strengthen shared shrinkage; when it lies below, we decrease λ_t to leave more room for reconstruction-driven adaptation. The feedback thereby controls the decline of the budget boundary.

Native opacity governs both rendering participation and low-opacity pruning, while changing λ_t preserves a shared first-order additive gradient before the optimizer. For independent probability masks or auxiliary sparse variables, the same native-opacity quantile does not directly measure selection progress; the activated-opacity L_1 gradient also varies by member.

During the selection window, we gradually remove low-opacity members as their opacities evolve. Once the budget is reached, we stop shared shrinkage, restore the original opacity learning rate, and continue standard reconstruction optimization on the survivors.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets and metrics. Following common practice in 3DGS (Kerbl et al., 2023), MaskGaussian (Liu et al., 2025c), and GaussianSpa (Zhang et al., 2025b), we evaluate 13 scenes from three datasets: nine Mip-NeRF 360 scenes (Barron et al., 2022), Train and Truck from Tanks and Temples (Knapitsch et al., 2017), and Playroom and Dr Johnson from Deep Blending (Hedman et al., 2018). Following the 3DGS view split, we sort each scene's images by filename, reserve every eighth image starting at index zero for testing, and use the rest for training. We report PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018) on all test views, averaging first over views within each scene and then equally over scenes within each dataset. As a scale reference for 3DGS comparisons, differences of about $0.05\,\mathrm{dB}$ in PSNR and 0.001 in SSIM or LPIPS are often noticeable in reported results.

Implementation and comparison methods. We use common source models, training endpoints, non-opacity parameter-group update schedules, and final counts to compare reconstruction quality under matched budgets. We compare LightGaussian (Fan et al., 2024), PUP (Hanson et al., 2025b), Mini-Splatting (Fang & Wang, 2024), GaussianSpa (Zhang et al., 2025b), and MaskGaussian (Liu et al., 2025c), alongside Naive L_1, which applies a fixed penalty to native activated opacity. Their pruning strategies are transferred to one training platform and share the per-scene dense checkpoint at the end of 15K densification iterations. Except for the budget curve, all methods target removal of approximately 80% of the source Gaussians and retain exactly $B = \mathrm{round}(0.2N_0)$ members, where N_0 is the source count. They continue from the common source to 30K; after 15K, no members are added, merged, or reinitialized, and each method performs its selection followed by reconstruction optimization. Each baseline retains its scoring rule, probabilistic mask, or auxiliary sparsity constraint; Appendix B gives its selection steps, budget operations, main parameters, and implementation checks. Dense continues from the same checkpoint to 30K without pruning and serves as an unpruned reference.

Controlled ID replay. The final quality of learned pruning depends on both the surviving subset and the parameter states accumulated during selection. To compare subset quality directly, we retain a stable ID for every Gaussian in the common 15K checkpoint, record the final surviving IDs of each method, return to that checkpoint, prune once by the recorded ID set, and optimize under common conditions to 30K. Every selected subset is thus evaluated from the same source state through one-shot pruning and subsequent optimization.

Table 1: Pruning methods on a dense input trained with ImprovedGS densification. Bold marks the best mean within each pruning protocol; pale blue marks GNS.

		Mip-NeRF 360			Tanks and Temples			Deep Blending		
	Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
	Dense	28.149	0.8368	0.1868	24.410	0.8717	0.1455	30.250	0.9147	0.2288
Regular pruning	LightGaussian	25.991	0.7447	0.3074	23.316	0.8164	0.2341	28.964	0.8861	0.2973
	PUP	26.779	0.8001	0.2499	22.874	0.8213	0.2151	29.828	0.9102	0.2551
	Mini-Splatting	27.028	0.8014	0.2514	23.417	0.8380	0.2018	29.791	0.9044	0.2656
	Naive L_1	27.543	0.8140	0.2442	23.966	0.8499	0.1929	30.286	0.9121	0.2560
	GaussianSpa	27.603	0.8051	0.2555	24.228	0.8545	0.1885	30.177	0.9130	0.2509
	MaskGaussian	27.888	0.8243	0.2289	24.203	0.8585	0.1796	**30.343**	0.9145	0.2491
	GNS	**27.946**	**0.8278**	**0.2224**	**24.264**	**0.8625**	**0.1720**	30.248	**0.9148**	**0.2436**
ID replay	Naive L_1	27.719	0.8182	0.2376	24.054	0.8539	0.1868	30.246	0.9126	0.2546
	GaussianSpa	27.657	0.8081	0.2514	24.239	0.8560	0.1859	30.254	0.9125	0.2506
	MaskGaussian	27.894	0.8249	0.2276	24.198	0.8587	0.1789	**30.310**	0.9139	0.2492
	GNS	**27.970**	**0.8282**	**0.2214**	**24.266**	**0.8627**	**0.1713**	30.253	**0.9146**	**0.2435**

4.2 Quantitative Comparison

Table 1 shows that, with a common dense input and matched final counts, GNS achieves the best mean of all three quality metrics on Mip-NeRF 360 and Tanks and Temples. On Deep Blending it has the best SSIM and LPIPS, while its PSNR is below MaskGaussian. Relative to explicit-score pruning and learned methods with extra selection variables, shared shrinkage and opacity adaptation yield better overall compact reconstruction quality with two simple changes.

In controlled ID replay, GNS retains the best three metrics on Mip-NeRF 360 and Tanks and Temples and the best SSIM and LPIPS on Deep Blending. The replayed Naive L_1, GaussianSpa, MaskGaussian, and GNS subsets also outperform all three heuristic methods' regular-pruning means across datasets and metrics. GNS's lead among the replayed subsets shows that its selection produces a strong retained set from the common checkpoint.

To examine quality across retention ratios, we compare the adapted methods over a range of budgets. Figure 2 shows higher SSIM and lower LPIPS for GNS at the measured 10%, 20%, and 40% retention points; as the budget grows, the gaps narrow toward the shared Full reference. The methods spread furthest apart at the smallest retention ratios, where each retained Gaussian must account for more of the reconstructed scene.

We also test GNS with dense inputs from 3DGS (Kerbl et al., 2023), MiniGS-D (Fang & Wang, 2024), and 2DGS (Huang et al., 2024). Although 2DGS changes the rendering primitive, it retains opacity as an explicit gate on rendering strength, allowing opacity-gated pruning strategies to be transferred.

Table 2 shows that GNS has the best pruned mean on all three metrics with 3DGS and 2DGS inputs. With 3DGS, its margin over MaskGaussian is small but consistent: $+0.014\,\mathrm{dB}$ PSNR, $+0.0014$ SSIM, and -0.0032 LPIPS. MiniGS-D gives a different trade-off: Naive L_1 has a $0.008\,\mathrm{dB}$ PSNR edge, while GNS raises SSIM from 0.8320 to 0.8343 and lowers LPIPS from 0.1979 to 0.1917. On the 2DGS surfel input, GNS improves PSNR and SSIM over both other reported pruning methods and lowers LPIPS to 0.2356, compared with 0.2565 for Naive L_1 and 0.2822 for GaussianSpa. The gains thus appear with different dense sources and rendering primitives, while the strongest competing method changes with the input. Per-scene results for all four inputs appear in Appendix D.

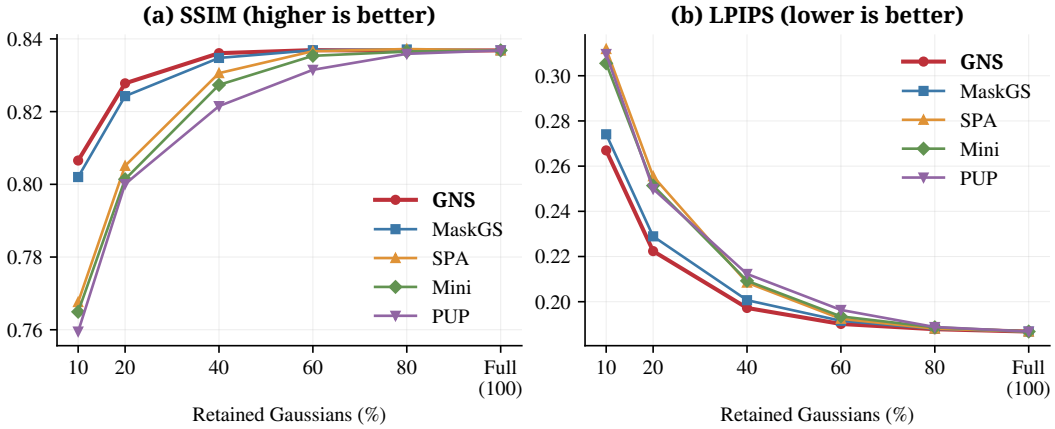

Figure 2: **Reconstruction quality at different retention budgets.** The horizontal axis gives the retained fraction of the common ImprovedGS dense source; 100% is the unpruned Full reference. Each point is an equal-weight mean over the nine Mip-NeRF 360 scenes, with lines connecting the measured $10/20/40/60/80/100\%$ points. Higher SSIM and lower LPIPS are better. MaskGS, SPA, and Mini denote MaskGaussian, GaussianSpa, and Mini-Splatting.

Table 2: Pruning results on Mip-NeRF 360 with different dense inputs. Bold marks the best reported pruned mean within each input; pale blue marks GNS. A dash means the strategy was not adapted or run.

	3DGS			MiniGS-D			2DGS		
Method	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Dense	27.882	0.8211	0.2113	27.669	0.8317	0.1761	27.277	0.8191	0.2099
LightGaussian	27.193	0.7993	0.2528	26.584	0.7844	0.2550	—	—	—
PUP	27.447	0.8089	0.2362	27.359	0.8166	0.2132	—	—	—
Mini-Splatting	27.373	0.8107	0.2375	27.277	0.8232	0.2060	—	—	—
Naive L_1	27.592	0.8141	0.2381	**27.774**	0.8320	0.1979	26.812	0.8019	0.2565
GaussianSpa	27.652	0.8129	0.2396	27.185	0.8086	0.2332	26.501	0.7826	0.2822
MaskGaussian	27.770	0.8193	0.2263	27.726	0.8329	0.1956	—	—	—
GNS	**27.784**	**0.8207**	**0.2231**	27.766	**0.8343**	**0.1917**	**27.100**	**0.8141**	**0.2356**

4.3 Qualitative Comparison

Figure 3 compares local reconstructions in Flowers, Garden, and Dr Johnson. In Flowers, GNS, MaskGaussian, Naive L_1, and Mini-Splatting preserve more vegetation texture and thin flower stems, whereas other methods show smoothing or stretching. In Garden, the learned pruning methods better preserve the contour and inner spacing of the dark grid than the heuristic methods. In Dr Johnson, GNS and GaussianSpa recover the horizontal and vertical bookshelf boundaries more clearly; MaskGaussian blurs the vertical dividers and LightGaussian loses the shown details. These local comparisons complement Table 1: GNS preserves fine structure and boundaries while reducing Gaussian count.

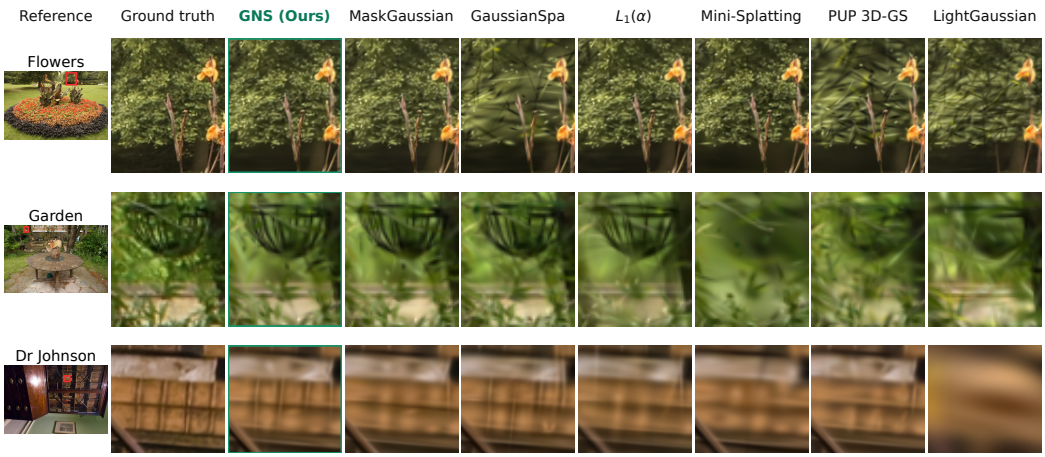

Figure 3: **Qualitative pruning comparison with an ImprovedGS dense input.** The final budgets in Flowers, Garden, and Dr Johnson are 300K, 600K, and 300K Gaussians, respectively. The left column identifies the local region; the remaining columns use the same camera, crop, and display size. $L_1(\alpha)$ denotes Naive L_1 in the main table, and the green border marks GNS.

4.4 Ablation Study

Table 3: Ablation of pre-activation shrinkage (Pre), opacity adaptation (LR), and automatic feedback (Auto). Values are means over nine Mip-NeRF 360 scenes; the Fixed L_1 row reuses Naive L_1 from the main table. Bold marks the best value in each protocol; pale blue marks full GNS.

		Regular pruning			ID replay		
Mechanism	Reg. control	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
L_1	Fixed	27.543	0.8140	0.2442	27.719	0.8182	0.2376
L_1	Auto	27.574	0.8110	0.2535	27.721	0.8155	0.2455
Pre only	Fixed	27.896	0.8247	0.2290	27.911	0.8253	0.2277
Pre only	Auto	27.901	0.8244	0.2292	27.929	0.8254	0.2271
LR only	Fixed	27.657	0.8208	0.2324	27.748	0.8232	0.2280
LR only	Auto	27.704	0.8201	0.2359	27.803	0.8226	0.2311
Full GNS	Fixed	27.937	0.8270	0.2243	27.963	0.8274	0.2234
Full GNS	Auto	**27.946**	**0.8278**	**0.2224**	**27.970**	**0.8282**	**0.2214**

Table 3 examines pre-activation shared shrinkage (Pre), faster opacity adaptation (LR), and their combination, each with fixed regularization strength (Fixed) or automatic feedback (Auto). The added Fixed variants use per-scene strengths refined through repeated trials; Auto adjusts to the budget boundary without per-scene manual selection of a fixed coefficient. Under both Fixed and Auto, Pre alone and LR alone improve the means of all three metrics over their corresponding L_1 baselines in regular pruning and ID replay, with a larger individual gain from Pre. Full GNS further improves over Pre only and LR only in each control group, showing that the two changes each help and work better together.

Auto has different quality effects under different competition mechanisms. For activated-opacity L_1 and LR only, Auto yields slightly higher PSNR but weaker SSIM and LPIPS than Fixed; the two controls are close for Pre only. With full GNS, Auto improves all three metrics slightly under both evaluation protocols and gives the best means in the table. This agrees with the applicability

discussion in Section 3.4: with native-opacity gating and neutral shrinkage, feedback avoids per-scene fixed-strength tuning while retaining high pruning quality.

4.5 Competition Sufficiency Diagnostic

We examine whether a higher opacity learning rate promotes differentiation of rendering responsibilities between nearby, similar Gaussians before pruning. Starting from the same ImprovedGS 15K checkpoint in bicycle and counter, we compare the original opacity learning rate 0.025 (LR1) with its fivefold value 0.125 (LR5). The two branches follow the same Gaussian-count trajectory and retain equal counts at each observation, reducing the influence of different deletion progress. From the common source, we select Gaussian pairs that are close in position and overlap in geometry and color, and track each member's full-image compositing contribution $W_i^{v,t} = \sum_{p \in v} w_{i,p}^t$ over four fixed training views. As a pairing control, we cyclically reassign one member of each pair to another pair while keeping the member set unchanged, disrupting the original neighborhood relationships.

Two event rates describe pre-deletion differentiation. *Opposite weight change* means one member's W rises while the other's falls. *Synchronous opposite change* also requires each opacity to move with its own W. Both rates use pairs, views, and intervals where the members survive and are visible in both branches as their common denominator.

Table 4: Pre-deletion responsibility differentiation at matched Gaussian counts. Event rates (%) and LR5–LR1 increments (pp) are rounded separately to one decimal. The main and auxiliary intervals are 10 and about 100 steps; both events use the 0.1% magnitude threshold.

Scene	Interval	Event	Similar pairs LR1 $\rightarrow$ LR5 (%)	Gain: real/control (pp)
bicycle	10 steps	Opposite weight	$17.3 \rightarrow 22.4$	$+5.1/+2.8$
bicycle	10 steps	Synchronous opposite	$6.0 \rightarrow 9.7$	$+3.7/+1.0$
counter	10 steps	Opposite weight	$17.5 \rightarrow 21.4$	$+3.9/+1.3$
counter	10 steps	Synchronous opposite	$6.4 \rightarrow 9.5$	$+3.0/+0.5$
bicycle	$\sim$100 steps	Opposite weight	$22.8 \rightarrow 32.3$	$+9.6/+1.3$
bicycle	$\sim$100 steps	Synchronous opposite	$8.5 \rightarrow 18.6$	$+10.1/+3.7$
counter	$\sim$100 steps	Opposite weight	$24.9 \rightarrow 32.8$	$+7.8/+0.3$
counter	$\sim$100 steps	Synchronous opposite	$10.0 \rightarrow 19.2$	$+9.2/+4.1$

Table 4 shows that both events become more frequent at the higher learning rate in both scenes: compositing contributions move in opposite directions more often, accompanied more often by corresponding opacity changes. When similar neighbors are replaced by mismatched pairs, the event-rate increments are substantially smaller at the main magnitude threshold; an auxiliary observation over about 100 steps preserves this pattern. These observations indirectly support faster adjustment of rendering responsibilities among nearby Gaussians at matched pruning progress. Appendix C gives the pair construction and event calculations.

5 Conclusion

GNS treats budget-constrained pruning after densification as competitive selection among overlapping Gaussians. It combines shared pre-activation shrinkage and faster opacity updates to promote responsibility differentiation through native parameters. Under matched dense inputs, training iterations, and target budgets, it improves compact reconstruction quality across datasets and dense representation sources. Controlled ID replay supports the quality of the selected subsets; a matched-count diagnostic finds more frequent opposite compositing and opacity changes at higher learning rates. Coordinating shrinkage and adaptation can therefore achieve high-quality pruning through simple changes.

AI USE STATEMENT

Generative AI tools assisted with code writing, figure and table preparation, manuscript revision and English translation, and provided feedback on the method, ablation design, and interpretation of results. The authors reviewed all AI-assisted work, checked the algorithm, experimental data, and manuscript claims, and take responsibility for the final content.

ETHICS STATEMENT

This work studies 3D reconstruction on established benchmark datasets. We are not aware of ethical issues specific to the proposed method or its evaluation.

REPRODUCIBILITY STATEMENT

Section 4.1 specifies the datasets and comparison protocol. Appendix A provides the complete GNS procedure and general parameters; Appendix B describes the transferred baselines and implementation checks. Appendices C and D give the mechanism measurements and per-scene results.

REFERENCES

Muhammad Salman Ali, Maryam Qamar, Sung-Ho Bae, and Enzo Tartaglione. Trimming the fat: Efficient compression of 3D Gaussian Splats through pruning. In *British Machine Vision Conference (BMVC)*, 2024. URL https://bmvc2024.org/proceedings/358/.

Bartłomiej Baranowski, Stefano Esposito, Patricia Gschoßmann, Anpei Chen, and Andreas Geiger. ConeGS: Error-guided densification using pixel cones for improved reconstruction with fewer primitives. In *International Conference on 3D Vision (3DV)*, 2026. URL https://arxiv.org/abs/2511.06810v2.

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. URL https://jonbarron.info/mipnerf360/.

Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. HAC: Hash-grid assisted context for 3D Gaussian Splatting compression. In *European Conference on Computer Vision (ECCV)*, 2024.

Xiaobin Deng, Changyu Diao, Min Li, Ruohan Yu, and Duanqing Xu. Improving Densification in 3D Gaussian Splatting for High-Fidelity Rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 223–232, 2026.

Xiaobiao Du, Yida Wang, Kun Zhan, and Xin Yu. Mobile-GS: Real-time Gaussian Splatting for mobile devices. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2603.11531.

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. LightGaussian: Unbounded 3D Gaussian compression with 15x reduction and 200+ FPS. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Guangchi Fang and Bing Wang. Mini-Splatting: Representing Scenes with a Constrained Number of Gaussians. In *European Conference on Computer Vision*, 2024.

Jian Gao, Chun Gu, Youtian Lin, Zhihao Li, Hao Zhu, Xun Cao, Li Zhang, and Yao Yao. Relightable 3D Gaussians: Realistic Point Cloud Relighting with BRDF Decomposition and Ray Tracing. In *Computer Vision – ECCV 2024*, pp. 73–89, 2025. doi: 10.1007/978-3-031-72995-9_5.

Sharath Girish, Kamal Gupta, and Abhinav Shrivastava. EAGLES: Efficient Accelerated 3D Gaussians with Lightweight EncodingS. In *Computer Vision – ECCV 2024*, pp. 54–71, 2025. doi: 10.1007/978-3-031-73036-8_4.

Alex Hanson, Allen Tu, Geng Lin, Vasu Singla, Matthias Zwicker, and Tom Goldstein. Speedy-Splat: Fast 3D Gaussian Splatting with Sparse Pixels and Sparse Primitives. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21537–21546, June 2025a.

Alex Hanson, Allen Tu, Vasu Singla, Mayuka Jayawardhana, Matthias Zwicker, and Tom 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)*, pp. 5949–5958, June 2025b.

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep Blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics*, 37(6), 2018. URL https://www-sop.inria.fr/reves/Basilic/2018/HPPFDB18/.

Jan Held, Renaud Vandeghen, Adrien Deliege, Abdullah Hamdi, Silvio Giancola, Anthony Cioppa, Andrea Vedaldi, Bernard Ghanem, Andrea Tagliasacchi, and Marc Van Droogenbroeck. Triangle Splatting for Real-Time Radiance Field Rendering. *arXiv preprint arXiv:2505.19175*, 2025. URL https://arxiv.org/abs/2505.19175v1.

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. In *SIGGRAPH 2024 Conference Papers*, 2024. doi: 10.1145/3641519.3657428.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023.

Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A Hierarchical 3D Gaussian Representation for Real-Time Rendering of Very Large Datasets. *ACM Transactions on Graphics*, 43(4), 2024. doi: 10.1145/3658160.

Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3D Gaussian Splatting as Markov Chain Monte Carlo. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Hyunjin Kim, Haebeom Jung, and Jaesik Park. Metropolis-hastings sampling for 3D Gaussian reconstruction. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://openreview.net/forum?id=BAscADMoQc.

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and Temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017. URL https://www.tanksandtemples.org/.

Dmytro Kotovenko, Olga Grebenkova, and Björn Ommer. EDGS: Eliminating Densification for Efficient Convergence of 3DGS. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 41065–41076, 2026.

Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3D Gaussian Representation for Radiance Field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21719–21728, June 2024.

Zhihao Liang, Qi Zhang, Ying Feng, Ying Shan, and Kui Jia. GS-IR: 3D Gaussian Splatting for Inverse Rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21644–21653, 2024.

Jiaqi Lin, Zhihao Li, Xiao Tang, Jianzhuang Liu, Shiyong Liu, Jiayue Liu, Yangdi Lu, Xiaofei Wu, Songcen Xu, Youliang Yan, and Wenming Yang. VastGaussian: Vast 3D Gaussians for Large Scene Reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5166–5175, 2024.

Hengyu Liu, Yuehao Wang, Chenxin Li, Ruisi Cai, Kevin Wang, Wuyang Li, Pavlo Molchanov, Peihao Wang, and Zhangyang Wang. FlexGS: Train once, deploy everywhere with many-in-one flexible 3D Gaussian Splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16336–16345, 2025a.

Yang Liu, Chuanchen Luo, Lue Fan, Naiyan Wang, Junran Peng, and Zhaoxiang Zhang. CityGaussian: Real-time High-quality Large-Scale Scene Rendering with Gaussians. In *Computer Vision – ECCV 2024*, pp. 265–282, 2025b.

Yifei Liu, Zhihang Zhong, Yifan Zhan, Sheng Xu, and Xiao Sun. MaskGaussian: Adaptive 3D Gaussian Representation from Probabilistic Masks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 681–690, June 2025c.

Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-GS: Structured 3D Gaussians for View-Adaptive Rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20654–20664, June 2024.

Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Markus Steinberger, Francisco Vicente Carrasco, and Fernando de la Torre. Taming 3DGS: High-quality radiance fields with limited resources. In *SIGGRAPH Asia 2024 Conference Papers*, 2024. doi: 10.1145/3680528.3687694.

Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. Compact 3D Scene Representation via Self-Organizing Gaussian Grids. In *Computer Vision – ECCV 2024*, pp. 18–34, 2025. doi: 10.1007/978-3-031-73013-9_2.

K L Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. CompGS: Smaller and Faster Gaussian Splatting with Vector Quantization. In *Computer Vision – ECCV 2024*, pp. 330–349, 2025. doi: 10.1007/978-3-031-73411-3_19.

Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3D Gaussian Splatting for Accelerated Novel View Synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10349–10358, 2024.

Lukas Radl, Michael Steiner, Mathias Parger, Alexander Weinrauch, Bernhard Kerbl, and Markus Steinberger. StopThePop: Sorted Gaussian Splatting for View-Consistent Real-time Rendering. *ACM Transactions on Graphics*, 43(4), 2024.

Haoran Wang, Guoxi Huang, Fan Zhang, David Bull, and Nantheera Anantrasirichai. Prune Wisely, Reconstruct Sharply: Compact 3D Gaussian Splatting via Adaptive Pruning and Difference-of-Gaussian Primitives. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11716–11725, June 2026a.

Hongfei Wang, Sisi Zeng, and Qing Yu. SplatOpt: An Optimization-Guided Framework for Joint Sparsification and High-Quality Reconstruction in 3D Gaussian Splatting. In *PRICAI 2025: Trends in Artificial Intelligence*, pp. 625–638, 2026b. doi: 10.1007/978-981-95-7081-2_43.

Tao Wang, Mengyu Li, Geduo Zeng, Cheng Meng, and Qiong Zhang. Gaussian herding across pens: An optimal transport perspective on global gaussian reduction for 3DGS. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-5277.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4), 2004. URL https://ece.uwaterloo.ca/~z70wang/publications/ssim.html.

Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4D Gaussian Splatting for Real-Time Dynamic Scene Rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20310–20320, 2024.

Shuzhao Xie, Weixiang Zhang, Chen Tang, Yunpeng Bai, Rongwei Lu, Shijia Ge, and Zhi Wang. MesonGS: Post-training Compression of 3D Gaussians via Efficient Attribute Transformation. In *Computer Vision – ECCV 2024*, pp. 434–452, 2025. doi: 10.1007/978-3-031-73414-4_25.

Butian Xiong, Rong Liu, Tiantian Zhou, Meida Chen, Zhiwen Fan, and Andrew Feng. NanoGS: Training-Free Gaussian Splat Simplification. *arXiv preprint arXiv:2603.16103*, 2026. URL https://arxiv.org/abs/2603.16103v3.

Kaifa Yang, Qi Yang, Yiling Xu, and Zhu Li. RAP: Fast Feedforward Rendering-Free Attribute-Guided Primitive Importance Score Prediction for Efficient 3D Gaussian Splatting Processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 33323–33332, 2026a.

Xianben Yang, Tao Wang, Yuxuan Li, Yi Jin, and Haibin Ling. GS^2: Graph-based Spatial Distribution Optimization for Compact 3D Gaussian Splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 33259–33268, 2026b.

Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. AbsGS: Recovering fine details in 3D Gaussian Splatting. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 1053–1061, 2024. doi: 10.1145/3664647.3681361.

Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-Splatting: Alias-free 3D Gaussian Splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19447–19456, June 2024.

Fengdi Zhang, Yibao Sun, Hongkun Cao, and Ruqi Huang. ControlGS: Consistent structural compression control for deployment-aware Gaussian Splatting. *arXiv preprint arXiv:2505.10473*, 2025a. URL https://arxiv.org/abs/2505.10473v3.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. URL https://richzhang.github.io/PerceptualSimilarity/.

Yangming Zhang, Wenqi Jia, Wei Niu, and Miao Yin. GaussianSpa: An "Optimizing-Sparsifying" Simplification Framework for Compact and High-Quality 3D Gaussian Splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26673–26682, June 2025b.

Zhaoliang Zhang, Tianchen Song, Yongjae Lee, Li Yang, Cheng Peng, Rama Chellappa, and Deliang Fan. LP-3DGS: Learning to Prune 3D Gaussian Splatting. In *Advances in Neural Information Processing Systems*, volume 37, pp. 122434–122457, 2024.

A General Configuration and Execution of GNS

This section gives the general settings and order of operations for running full GNS from a dense checkpoint. The inputs are the Gaussian parameters and optimizer states at the end of densification. For source count N_0 and requested retention ratio r, we fix budget $B = \mathrm{round}(rN_0)$; the main experiments use $r = 0.2$. Training continues from the common 15K checkpoint to 30K, with no new, merged, or reinitialized Gaussians during selection.

A.1 General Hyperparameters

Table 5 lists the general selection settings. Each dense input retains its own renderer, reconstruction objective, and non-opacity parameter-group settings; GNS adds pre-activation linear shrinkage and adjusts the native opacity learning rate during selection. For the 3D Gaussian inputs, the reconstruction loss is $0.8\,\mathrm{mean}|I - \hat{I}| + 0.2(1 - \mathrm{SSIM}_{\mathrm{fused}})$; the 2D primitive input retains its original reconstruction terms.

Table 5: General GNS selection parameters. Budget B is determined by source size and the requested retention ratio.

Parameter	General setting
Selection window	$a = 15001$, $b = 20001$; regularization applies for $a \leq t < b$
Initial regularization coefficient	$\lambda_a = 0.002$
Opacity learning rate during selection	0.125: $5\times$ a base rate of 0.025, or $2.5\times$ a base rate of 0.05
Low-opacity deletion threshold	$\tau = 0.005$
Feedback and deletion interval	Every 10 iterations
Reference trajectory for budget boundary	Linear decline from its initial value to zero, reaching zero 100 iterations before the window ends
Feedback tolerance and coefficient update	10% relative tolerance; multiply by 0.95 below or 1.05 above the tolerance band

A.2 Updates Within the Selection Window

Let S_t be the current survivor set, $N_t = |S_t|$, and $\alpha_i = \sigma(z_i)$ the native opacity. At every iteration $a \leq t < b$ before the budget is reached, the pre-activation linear term is added to the reconstruction objective. The feedback interval controls coefficient adjustment; regularization itself is applied every iteration. Upon entering the window, we read the B-th largest opacity among all current members as q_a and set $\lambda_a = 0.002$. At subsequent iteration numbers divisible by 10, we read the current B-th largest opacity q_t and compare it with the reference trajectory:

$$q_t^\star = q_a \max\left(1 - \frac{t-a}{b-a-100}, 0\right), \qquad \lambda_t \leftarrow \begin{cases} 0.95\lambda_t, & q_t < 0.9q_t^\star, \\ 1.05\lambda_t, & q_t > 1.1q_t^\star, \\ \lambda_t, & \text{otherwise.} \end{cases} \tag{8}$$

This feedback tracks the opacity at the budget boundary, while the survivor count follows its own decay trajectory. The coefficient remains unchanged between feedback checks.

At each iteration, we first update the coefficient when due and then construct

$$\mathcal{L}_t = \mathcal{L}_{\mathrm{rec}} + \lambda_t \frac{1}{N_t} \sum_{i \in S_t} z_i^t. \tag{9}$$

We backpropagate and update parameters according to the original optimization schedule. Here z_i is the signed pre-activation parameter, and the mean is taken over all surviving members. The first update in the selection window still uses the source model's base opacity learning rate; at the end of that iteration, it is raised to the selection value in Table 5. Other parameter groups retain their original schedules.

A.3 Budget Protection and Selection Termination

After the parameter update at every iteration number divisible by 10, we remove members with opacity below τ. If this would leave fewer than B members, we instead retain exactly the top B by current opacity. Tie-breaking uses $\alpha_i + \epsilon_i$ with independent $\epsilon_i \in [0, 10^{-6})$, which can also swap members with nearly equal opacity. The corresponding optimizer states are pruned together with the members, and survivor states are preserved.

Upon reaching B, we stop regularization and pruning immediately and restore the source model's base opacity learning rate. If the budget has not been reached earlier, iteration b updates parameters without the regularizer and then retains the top B by the same ranking. The selection mechanism is removed, and survivors continue under the original reconstruction objective and schedule to the common 30K endpoint.

B Transferred Pruning Baselines and Implementation Checks

This section records the baseline selection rules and their execution on the common training platform. For each scene, methods resume from the same 15K dense checkpoint and its Adam states, target $B = \mathrm{round}(0.2N_0)$ surviving members in the main comparison, and are evaluated at 30K. Method-specific scores, masks, and sparse variables are updated during selection as in Table 6; common reconstruction optimization continues after the budget is reached.

B.1 Selection Rules and Budget Schedules

Table 6: Transferred baseline selection mechanisms and main-experiment settings. N_0 is the 15K source count and B the final retention budget; iterations continue the numbering of the source training run.

Method	Selection computation	Steps and main parameters
Naive L_1	Apply $\lambda\,\mathrm{mean}(\alpha)$ to activated opacity and gradually remove low-opacity members.	Check threshold 0.005 every 10 iterations from 15001; retain top B by opacity by 23001. Coefficient groups are described below.
MaskGaussian	Use a straight-through binary Gumbel-Softmax mask in rendering; unsampled members update their existence probabilities through mask gradients.	Prune every 1000 iterations from 15001 using 10 hard samples; retain top B by survival probability by 23001. Regularizer: $0.05[\mathrm{mean}(m)]^2$; mask learning rate: 0.01.
GaussianSpa	Optimize native opacity under reconstruction and ADMM constraints while sparsely projecting an auxiliary variable; select by native opacity at the end.	Apply the penalty and alternating updates every 50 iterations from 15001–20000, retaining B at 20001; $\rho = 0.0005$.
Mini-Splatting	Sample without replacement from the original indoor/outdoor importance distribution, then recompute contribution and rank.	Retain $2B$ before the update at 15001 and B before the update at 20000.
PUP	Accumulate a 6×6 Fisher matrix for position and log-scale and score a log-sum of its singular values; downsample score images by four in width and height.	As in Mini-Splatting, retain $2B$ and B before updates at 15001 and 20000.
LightGaussian	Compute global importance from visibility count, opacity, and volume with exponent 0.1.	Retain B once, before rendering and update at 15001.

Naive L_1 uses $\lambda = 0.5$ for its lower-strength group and 1.0 for its higher-strength group with ImprovedGS, 3DGS, and 2DGS inputs; MiniGS-D uses 0.1 and 0.2, respectively. The higher-strength group comprises bicycle, flowers, garden, stump, treehill, and truck; all other scenes use the lower strength. In the formal runs, MaskGaussian and Naive L_1 finish pruning between 17K and 19K in most scenes. MaskGaussian reaches the budget through periodic removal and never triggers the 23K budget fallback. With MiniGS-D, a small fraction of members in Naive L_1 are removed by the final top-B selection: the activated-opacity penalty contributes a logit gradient proportional to $\alpha(1-\alpha)$, which weakens at low opacity. Increasing the fixed coefficient for this input produced similar or slightly worse final quality, so the reported results use the configuration in Table 6. MiniGS-D and 2DGS retain their base opacity learning rate of 0.05, twice the 0.025 used with ImprovedGS and 3DGS; the faster opacity response gives members more opportunity to adjust rendering participation before deletion. Among the four dense inputs in Table 2, Naive L_1 is closest to GNS in SSIM and LPIPS with MiniGS-D, and its PSNR is $0.008\,\mathrm{dB}$ higher. For the budget curve at retention ratios above 50%, Mini-Splatting and PUP select B at their first selection step.

Selection progress and ablation protocol. Across the tested GNS selection-window endpoints from 17K to 23K, final quality is similar; the formal experiments use the rounded setting near 20K. Fixed coefficients and automatic feedback each produce their own regularization strengths and removal trajectories. In the LR1/LR5 diagnostic with matched Gaussian counts, the feedback adjusts the coefficient separately in each branch to align count progress; using the same fixed coefficient in both branches would change their natural decay progress when the learning rate rises. In Table 3, Fixed and Auto full GNS have similar mean quality, while Pre and LR yield gains in the same direction under both controls. The matched-count diagnostic in Appendix C further observes stronger opposite changes in weight and opacity for nearby, similar pairs at the higher learning rate, in line with the competition-sufficiency design.

B.2 Reference-Implementation Checks

Using fixed Gaussian parameters, cameras, and random states where needed, we compared the selection computations and gradients against reference implementations or independent calculations. Table 7 summarizes the key checks.

Table 7: Fixed-input checks of the transferred baselines.

Method	Check
MaskGaussian	Mask samples and logit gradients match the reference function; rendered images agree, and the maximum model-parameter gradient difference is 2.86×10^{-6}, including the masked-out member gradient path.
Mini-Splatting	Sampling without replacement gives identical indices at matched inputs and random states; the maximum cumulative-weight difference is 7.15×10^{-7}.
PUP	The per-color-channel Fisher computation agrees with per-pixel automatic differentiation to relative Frobenius error below 2.0×10^{-7}.
LightGaussian	Volume-weighted scores agree with the reference function; atomic visibility counts agree with an independent count implementation.
GaussianSpa	Auxiliary updates in a released-code-compatible branch agree with the reference; projections and continuous updates in the formal ADMM branch agree with an independent calculation to 10^{-12} tolerance.
2DGS renderer	Rendered images agree with the reference kernel on fixed 2D surfels; the maximum parameter-gradient difference is 1.91×10^{-6}.

The formal PUP Fisher calculation uses independent RGB gradient buffers; LightGaussian uses atomic visibility counting. Our formal GaussianSpa comparison follows its paper's algorithm: initialize $z^{(0)} = \alpha^{(0)}$ and $u^{(0)} = 0$, then update the original variable, sparse auxiliary variable, and dual variable in order. The released-code-compatible branch differs in auxiliary initialization and

update timing. The formal branch projects $\alpha + u$, retaining the B components of greatest absolute value, and selects the final B Gaussians by native opacity.

C Construction and Measurement of the Competition-Sufficiency Diagnostic

Matching structural contraction. Both diagnostic branches start from the same dense model and change only the native opacity learning rate; their regularization strengths are adjusted separately by the automatic feedback controller. To compare adaptation at matched pruning progress, we use the same target count sequence N_t in both branches. When natural deletion at a pruning step falls short, we remove additional members by native-opacity ranking so that both branches retain equal counts at the corresponding observations. We then record states and rendering contributions at fixed training views and matched observation times.

Nearby similar pairs and mismatched controls. Candidate pairs are formed from the common source model using position, geometry, and color. For each Gaussian, we first retrieve its $K = 32$ nearest neighbors by center position, then measure geometric overlap and DC color similarity. For Gaussian i with center $\boldsymbol{\mu}_i$, rotation R_i, and log-scale $\mathbf{s}_i$, let $\Sigma_i = R_i \operatorname{diag}(e^{2\mathbf{s}_i}) R_i^\top$, $\bar{\Sigma}_{ij} = (\Sigma_i + \Sigma_j)/2$, and $\boldsymbol{\delta}_{ij} = \boldsymbol{\mu}_i - \boldsymbol{\mu}_j$. The Bhattacharyya coefficient of the 3D Gaussian distributions measures geometric overlap:

$$\begin{aligned} D_{ij} &= \frac{1}{8}\boldsymbol{\delta}_{ij}^\top \bar{\Sigma}_{ij}^{-1}\boldsymbol{\delta}_{ij} + \frac{1}{2}\log\frac{|\bar{\Sigma}_{ij}|}{\sqrt{|\Sigma_i||\Sigma_j|}}, \\ BC_{ij} &= \exp(-D_{ij}). \end{aligned} \tag{10}$$

We construct DC color $\mathbf{c}_i$ from zero-order spherical harmonics and define $d_{ij}^{\mathrm{DC}} = \|\mathbf{c}_i - \mathbf{c}_j\|_2$. Each member selects the neighbor maximizing $BC_{ij}\exp[-(d_{ij}^{\mathrm{DC}})^2/(2\sigma_c^2)]$ with $\sigma_c = 0.1$. We retain mutual best matches satisfying $BC_{ij} \geq 0.75$ and $d_{ij}^{\mathrm{DC}} \leq 0.05$, so each member occurs in at most one pair.

The mismatched control retains the same members and cyclically shifts the ordered list of right-side partners by $\lfloor P/2 \rfloor$, where P is the number of pairs. Thus (l_k, r_k) becomes $(l_k, r_{(k+\lfloor P/2 \rfloor) \bmod P})$, changing pair relationships while preserving the member set.

Compositing contribution and opposite-change events. For view v and snapshot t, native opacity α_i^t and projected response $\phi_{i,p}^{v,t}$ give effective pixel opacity $a_{i,p}^{v,t} = \min(0.99, \alpha_i^t \phi_{i,p}^{v,t})$. The compositing weight is $w_{i,p}^{v,t} = T_{i,p}^{v,t} a_{i,p}^{v,t}$, where T is transmittance immediately before this member. Summing over the full image gives $W_i^{v,t}$.

For each pairing, we form a common set of valid pair–view–interval observations: both members survive at both endpoints in LR1 and LR5, each has $W > 10^{-6}$, and they are visible at the shared start. Let $\Delta W_k = W_k^{v,t'} - W_k^{v,t}$ and $\Delta\alpha_k = \alpha_k^{t'} - \alpha_k^t$. The change threshold uses the LR1 pair contribution at the start of that view, $W_{ij,0}^v = W_{i,0}^v + W_{j,0}^v$. For each branch, define

$$\begin{aligned} E_W &= \mathbf{1}[\Delta W_i \Delta W_j < 0] \prod_{k\in\{i,j\}} \mathbf{1}[|\Delta W_k| > 0.001 W_{ij,0}^v], \\ E_{\alpha W} &= E_W \prod_{k\in\{i,j\}} \mathbf{1}[\Delta\alpha_k \Delta W_k > 0,\ |\Delta\alpha_k| > 10^{-6}]. \end{aligned} \tag{11}$$

Both rates use all common valid pair–view–interval observations as their denominator. For similar and mismatched pairs, we calculate each event-rate increment from LR1 to LR5 and compare those increments.

The main analysis compares short intervals between adjacent snapshots. The auxiliary analysis samples the same snapshots more sparsely, rebuilds the valid observation sets, and measures the two opposite-change events over longer intervals.

D PER-SCENE RESULTS FOR FOUR DENSE INPUTS

Tables 8–19 report regular-pruning results per scene for four dense inputs, with separate PSNR, SSIM, and LPIPS tables for each input. ImprovedGS covers 13 scenes from three datasets; the other inputs cover the nine Mip-NeRF 360 scenes. Bold marks the best reported pruning method in each scene; Dense is the unpruned reference. Table 20 lists the per-scene source counts and main-experiment retention budgets.

D.1 IMPROVEDGS

Table 8: ImprovedGS input: per-scene PSNR↑ (dB)

	Mip-NeRF 360								
Method	bicycle	flowers	garden	stump	treehill	bonsai	counter	kitchen	room
Dense	25.872	21.808	28.026	27.306	22.850	32.786	29.640	32.448	32.602
LightGaussian	23.559	19.737	25.975	25.324	22.114	29.162	27.779	29.636	30.632
PUP	24.977	20.848	26.723	26.819	22.298	31.394	28.435	28.114	31.404
Mini-Splatting	25.228	20.754	25.272	27.076	22.225	31.489	28.729	30.921	31.554
Naive L_1	25.539	21.473	27.300	27.158	23.210	31.613	28.891	30.780	31.922
GaussianSpa	25.236	21.402	27.220	27.204	23.010	32.019	29.055	31.249	32.036
MaskGaussian	25.672	21.780	27.691	27.412	**23.222**	32.178	29.268	31.578	**32.193**
GNS	**25.749**	**21.817**	**27.726**	**27.433**	23.218	**32.339**	**29.317**	**31.729**	32.188

	Tanks and Temples		Deep Blending	
Method	train	truck	playroom	Dr Johnson
Dense	22.409	26.411	30.668	29.832
LightGaussian	21.591	25.041	29.263	28.664
PUP	20.718	25.030	30.263	29.393
Mini-Splatting	21.468	25.365	30.211	29.371
Naive L_1	22.062	25.870	30.650	29.922
GaussianSpa	22.237	26.218	30.486	29.868
MaskGaussian	22.223	26.183	**30.707**	**29.978**
GNS	**22.268**	**26.259**	30.607	29.889

Table 9: ImprovedGS input: per-scene SSIM↑

Method	Mip-NeRF 360								
	bicycle	flowers	garden	stump	treehill	bonsai	counter	kitchen	room
Dense	0.80441	0.64070	0.88082	0.81254	0.65704	0.94930	0.91785	0.93606	0.93281
LightGaussian	0.63668	0.48312	0.79929	0.71404	0.54906	0.88219	0.86241	0.88114	0.89468
PUP	0.75244	0.55811	0.83251	0.78733	0.61938	0.93613	0.89629	0.90311	0.91540
Mini-Splatting	0.76604	0.55904	0.81658	0.79656	0.62846	0.93042	0.88983	0.91136	0.91429
Naive L_1	0.77867	0.59024	0.85171	0.80030	0.64125	0.93472	0.90013	0.91078	0.91789
GaussianSpa	0.74932	0.57311	0.84254	0.80040	0.61337	0.93753	0.90080	0.90975	0.91890
MaskGaussian	0.78765	0.61192	0.86453	0.81383	0.64879	0.94145	0.90696	0.92112	0.92209
GNS	**0.79338**	**0.61626**	**0.86739**	**0.81591**	**0.65390**	**0.94393**	**0.90995**	**0.92366**	**0.92560**

Method	Tanks and Temples		Deep Blending	
	train	truck	playroom	Dr Johnson
Dense	0.84473	0.89857	0.91651	0.91284
LightGaussian	0.77100	0.86184	0.89336	0.87891
PUP	0.77674	0.86581	0.91263	0.90776
Mini-Splatting	0.79894	0.87706	0.90965	0.89910
Naive L_1	0.81617	0.88369	0.91258	0.91165
GaussianSpa	0.82087	0.88806	0.91464	0.91139
MaskGaussian	0.82502	0.89200	0.91604	0.91293
GNS	**0.83053**	**0.89448**	**0.91623**	**0.91336**

Table 10: ImprovedGS input: per-scene LPIPS↓

Method	Mip-NeRF 360								
	bicycle	flowers	garden	stump	treehill	bonsai	counter	kitchen	room
Dense	0.16312	0.28342	0.09262	0.17481	0.28337	0.18898	0.18323	0.11607	0.19552
LightGaussian	0.34826	0.43067	0.19843	0.29550	0.45640	0.28251	0.27072	0.20809	0.27576
PUP	0.24723	0.39053	0.16105	0.21688	0.38536	0.22051	0.22465	0.16844	0.23452
Mini-Splatting	0.23516	0.40563	0.17778	0.20997	0.36926	0.22678	0.23675	0.16288	0.23836
Naive L_1	0.23423	0.37615	0.15408	0.22187	0.37570	0.22059	0.22074	0.16031	0.23380
GaussianSpa	0.27224	0.38907	0.16495	0.21927	0.40719	0.22150	0.22136	0.17166	0.23229
MaskGaussian	0.22197	0.35613	0.13163	0.19940	0.36101	0.20942	0.20939	0.14525	0.22615
GNS	**0.20981**	**0.35361**	**0.12607**	**0.19440**	**0.35256**	**0.20509**	**0.20407**	**0.14090**	**0.21505**

Method	Tanks and Temples		Deep Blending	
	train	truck	playroom	Dr Johnson
Dense	0.17081	0.12017	0.22904	0.22863
LightGaussian	0.28054	0.18775	0.29971	0.29488
PUP	0.26581	0.16433	0.25619	0.25393
Mini-Splatting	0.24221	0.16143	0.26274	0.26847
Naive L_1	0.23235	0.15340	0.26164	0.25027
GaussianSpa	0.22588	0.15113	0.25329	0.24855
MaskGaussian	0.22101	0.13826	0.25225	0.24585
GNS	**0.21118**	**0.13289**	**0.24497**	**0.24221**

D.2 3DGS

Table 11: 3DGS input: per-scene PSNR↑ (dB)

Method	bicycle	flowers	garden	stump	treehill	bonsai	counter	kitchen	room
Dense	25.472	21.814	27.694	26.811	22.971	32.630	29.303	32.050	32.195
LightGaussian	25.028	21.565	26.770	26.589	22.867	30.738	28.521	31.159	31.496
PUP	25.151	21.553	27.054	26.651	22.914	31.856	28.850	31.195	31.798
Mini-Splatting	25.351	21.327	26.614	26.787	22.769	31.723	28.765	31.269	31.753
Naive L_1	25.467	21.677	27.391	26.919	22.980	31.674	28.917	31.464	31.842
GaussianSpa	25.330	21.686	27.325	26.871	23.020	32.042	28.990	31.625	31.981
MaskGaussian	**25.513**	21.797	27.594	**26.923**	23.079	32.161	29.100	31.752	**32.009**
GNS	25.493	**21.802**	**27.607**	26.909	**23.091**	**32.257**	**29.108**	**31.784**	32.005

Table 12: 3DGS input: per-scene SSIM↑

Method	bicycle	flowers	garden	stump	treehill	bonsai	counter	kitchen	room
Dense	0.77327	0.61209	0.87134	0.77849	0.64231	0.94463	0.91171	0.93183	0.92391
LightGaussian	0.74354	0.58987	0.83752	0.76954	0.62560	0.91623	0.88528	0.91563	0.91053
PUP	0.75493	0.58787	0.84696	0.77144	0.63818	0.93656	0.90180	0.92326	0.91883
Mini-Splatting	0.76835	0.58967	0.84582	0.77880	0.63928	0.93435	0.89984	0.92181	0.91848
Naive L_1	0.76938	0.59839	0.85663	0.78326	0.64482	0.93313	0.90104	0.92224	0.91770
GaussianSpa	0.76330	0.59641	0.85584	0.78154	0.63218	0.93730	0.90414	0.92441	0.92066
MaskGaussian	0.77427	0.60792	0.86624	**0.78580**	0.64622	0.93930	0.90640	0.92698	0.92062
GNS	**0.77593**	**0.60910**	**0.86766**	0.78560	**0.64787**	**0.94151**	**0.90796**	**0.92830**	**0.92219**

Table 13: 3DGS input: per-scene LPIPS↓

Method	bicycle	flowers	garden	stump	treehill	bonsai	counter	kitchen	room
Dense	0.20006	0.33214	0.10329	0.20931	0.31673	0.20372	0.19780	0.12314	0.21545
LightGaussian	0.25718	0.37301	0.15247	0.24009	0.36601	0.24512	0.24338	0.15298	0.24471
PUP	0.23263	0.37007	0.14025	0.22599	0.35009	0.22252	0.21801	0.13887	0.22751
Mini-Splatting	0.22334	0.37059	0.14450	0.22701	0.34804	0.22637	0.22400	0.14377	0.22966
Naive L_1	0.23170	0.36751	0.13994	0.22777	0.35147	0.22752	0.22234	0.14169	0.23284
GaussianSpa	0.24158	0.37172	0.13631	0.23448	0.37132	0.22126	0.21516	0.13936	0.22562
MaskGaussian	0.21938	0.35263	0.11996	0.21914	0.34179	0.21553	0.21112	0.13249	0.22496
GNS	**0.21277**	**0.35064**	**0.11755**	**0.21619**	**0.33684**	**0.21318**	**0.20828**	**0.13088**	**0.22163**

D.3 MINIGS-D

Table 14: MiniGS-D input: per-scene PSNR↑ (dB)

Method	bicycle	flowers	garden	stump	treehill	bonsai	counter	kitchen	room
Dense	25.632	21.539	27.843	27.173	22.356	32.069	29.010	31.693	31.707
LightGaussian	24.140	20.724	26.601	26.065	22.378	30.131	27.871	30.319	31.025
PUP	25.225	21.305	27.093	26.953	22.526	31.932	28.828	30.990	31.376
Mini-Splatting	25.417	21.479	25.947	27.206	22.474	31.679	28.756	31.094	31.442
Naive L_1	25.677	**21.895**	27.507	27.343	**23.053**	31.988	**29.047**	31.639	**31.821**
GaussianSpa	25.089	21.201	27.203	26.734	22.567	31.269	28.539	30.697	31.365
MaskGaussian	25.665	21.802	27.700	27.420	22.776	31.956	28.987	31.543	31.682
GNS	**25.685**	21.720	**27.732**	**27.430**	22.709	**32.051**	29.031	**31.820**	31.720

Table 15: MiniGS-D input: per-scene SSIM↑

Method	bicycle	flowers	garden	stump	treehill	bonsai	counter	kitchen	room
Dense	0.79683	0.64275	0.87836	0.80291	0.63788	0.94792	0.91467	0.93528	0.92850
LightGaussian	0.69881	0.58317	0.83023	0.75333	0.61038	0.90997	0.86641	0.90088	0.90608
PUP	0.76386	0.60715	0.84763	0.79089	0.63437	0.94555	0.91046	0.92664	0.92262
Mini-Splatting	0.78719	0.63103	0.84315	0.80635	0.64506	0.94245	0.90536	0.92545	0.92306
Naive L_1	0.79370	0.64064	0.86616	0.81214	**0.65878**	0.94633	0.91331	0.92947	0.92737
GaussianSpa	0.75873	0.61036	0.84782	0.77928	0.61754	0.93555	0.89635	0.91460	0.91677
MaskGaussian	0.79527	0.64480	0.87183	0.81514	0.65337	0.94589	0.91281	0.93042	0.92633
GNS	**0.79877**	**0.64580**	**0.87361**	**0.81544**	0.65351	**0.94714**	**0.91431**	**0.93207**	**0.92790**

Table 16: MiniGS-D input: per-scene LPIPS↓

Method	bicycle	flowers	garden	stump	treehill	bonsai	counter	kitchen	room
Dense	0.15693	0.24977	0.08962	0.16886	0.25892	0.18102	0.17326	0.11410	0.19276
LightGaussian	0.28247	0.33673	0.16089	0.24449	0.35505	0.24380	0.24902	0.17478	0.24778
PUP	0.22357	0.31884	0.13903	0.19912	0.30925	0.19542	0.19222	0.13113	0.21048
Mini-Splatting	0.18905	0.30496	0.13964	0.18354	0.29398	0.19861	0.19984	0.13235	0.21180
Naive L_1	0.18884	0.29293	0.11842	0.18677	**0.28244**	0.19355	0.18802	0.12770	0.20271
GaussianSpa	0.23548	0.31813	0.14876	0.23320	0.35098	0.21472	0.21739	0.15147	0.22886
MaskGaussian	0.18889	0.28401	0.10731	0.18151	0.28838	0.19312	0.18757	0.12438	0.20500
GNS	**0.17941**	**0.28119**	**0.10487**	**0.17839**	0.28289	**0.19102**	**0.18471**	**0.12265**	**0.20059**

D.4 2DGS

Table 17: 2DGS input: per-scene PSNR↑ (dB)

Method	bicycle	flowers	garden	stump	treehill	bonsai	counter	kitchen	room
Dense	25.221	21.473	27.135	26.671	22.160	31.918	28.619	31.223	31.069
Naive L_1	25.072	21.400	26.558	26.601	**22.664**	30.618	27.901	29.939	30.551
GaussianSpa	24.375	21.120	25.989	25.969	22.443	30.524	27.888	29.732	30.470
GNS	**25.181**	**21.549**	**26.932**	**26.727**	22.542	**31.345**	**28.260**	**30.551**	**30.817**

Table 18: 2DGS input: per-scene SSIM↑

Method	bicycle	flowers	garden	stump	treehill	bonsai	counter	kitchen	room
Dense	0.77432	0.62490	0.85937	0.78570	0.63401	0.94056	0.90544	0.92738	0.92039
Naive L_1	0.75817	0.60102	0.82804	0.77885	0.63910	0.92277	0.88167	0.90190	0.90524
GaussianSpa	0.70956	0.58240	0.80008	0.74746	0.61360	0.91456	0.87914	0.89088	0.90536
GNS	**0.77126**	**0.62015**	**0.84708**	**0.78937**	**0.64347**	**0.93325**	**0.89453**	**0.91248**	**0.91535**

Table 19: 2DGS input: per-scene LPIPS↓

Method	bicycle	flowers	garden	stump	treehill	bonsai	counter	kitchen	room
Dense	0.19231	0.30141	0.11804	0.20712	0.29902	0.20953	0.20966	0.13148	0.22012
Naive L_1	0.24222	0.35806	0.18081	0.23949	0.34344	0.24682	0.25515	0.18197	0.26070
GaussianSpa	0.29605	0.37985	0.21714	0.28166	0.38078	0.26303	0.25959	0.20334	0.25866
GNS	**0.21928**	**0.33792**	**0.15193**	**0.22077**	**0.32670**	**0.22996**	**0.23446**	**0.16271**	**0.23691**

D.5 Dense Input Size and Target Budgets

Table 20: Dense source count N_0 and the 20% retention budget B for four input sources.

Dense input	Scene	Dense N_0	Retention budget B
ImprovedGS	bicycle	3,000,000	600,000
	flowers	1,500,000	300,000
	garden	3,000,000	600,000
	stump	3,000,000	600,000
	treehill	1,500,000	300,000
	bonsai	1,000,000	200,000
	counter	919,532	183,906
	kitchen	942,710	188,542
	room	1,000,000	200,000
	train	1,000,000	200,000
	truck	1,500,000	300,000
	playroom	1,000,000	200,000
	Dr Johnson	1,500,000	300,000
3DGS	bicycle	4,879,101	975,820
	flowers	2,835,293	567,059
	garden	4,231,621	846,324
	stump	3,935,485	787,097
	treehill	3,162,830	632,566
	bonsai	1,068,156	213,631
	counter	1,088,084	217,617
	kitchen	1,564,858	312,972
	room	1,368,825	273,765
MiniGS-D	bicycle	4,521,515	904,303
	flowers	4,525,636	905,127
	garden	4,554,884	910,977
	stump	4,545,830	909,166
	treehill	4,517,359	903,472
	bonsai	2,257,308	451,462
	counter	2,232,519	446,504
	kitchen	2,093,990	418,798
	room	2,251,677	450,335
2DGS	bicycle	4,135,133	827,027
	flowers	2,805,030	561,006
	garden	2,814,954	562,991
	stump	3,450,456	690,091
	treehill	4,017,412	803,482
	bonsai	806,382	161,276
	counter	681,958	136,392
	kitchen	929,797	185,959
	room	1,029,009	205,802