SPLATQ: INTERACTION-AWARE PRUNING OF 3D GAUSSIAN SPLATTING VIA QUADRATIC SELECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing 3D Gaussian Splatting (3DGS) pruning methods primarily rely on global Top-K selection based on independent importance scores, lacking regional budget control and ignoring joint deletion effects. Consequently, they cannot ensure balanced reconstruction quality across a scene. We propose SplatQ, a combinatorial pruning algorithm that coordinates regional budget allocation and joint Gaussian selection under a fixed budget. SplatQ partitions Gaussians by spatial proximity and appearance similarity into size-limited blocks, decomposing global selection into subproblems of a specific size. It then smooths importance-based block retention quotas to prevent excessive budget concentration. Finally, it constructs a quadratic approximation of joint deletion effects using the responses of representative pixel colors to changes in Gaussian opacity. Individual responses, pairwise interactions, and quota penalties are incorporated into a quadratic unconstrained binary optimization (QUBO) objective, solved using simulated annealing (SA) or a coherent Ising machine (CIM) to jointly select Gaussians. Experiments on 5 benchmarks demonstrate that SplatQ outperforms state-of-the-art pruning methods in aggregate reconstruction quality, spatial quality balance, and rendering speed. Ablations confirm that budget smoothing improves local detail reconstruction, while interaction modeling improves Gaussian selection quality under a fixed budget.

1 INTRODUCTION

Gaussian Splatting (Kerbl et al., 2023) enables high-quality real-time rendering through an explicit Gaussian representation, providing a practical foundation for the reconstruction and interaction of 3D digital assets. Complex scenes, however, often contain large numbers of Gaussians, whose storage, memory, and rendering costs limit asset transmission and deployment (Fan et al., 2024; Girish et al., 2024). Gaussian pruning reduces these costs by removing primitives. Methods such as LightGaussian (Fan et al., 2024) and PUP 3D-GS (Hanson et al., 2025b) combine importance estimation with post-pruning fine-tuning to balance model size and rendering quality.

Under a fixed budget and aggressive pruning, we observe that results with similar overall image-quality metrics can exhibit different patterns of local degradation: structures and textures remain clear in some regions, while other regions become visibly blurred or distorted. Existing methods also make different regional trade-offs. For example, PUP 3D-GS discusses the trade-off between foreground detail and background quality (Hanson et al., 2025b), whereas GaussianPOP emphasizes the preservation of detail-rich regions (Lee et al., 2026). These choices have their own motivations, but they raise a question: when the number of retained Gaussians is strictly limited, how can pruning preserve both overall rendering quality and reconstruction quality across different regions?

We have two observations from previous global importance ranking works (Fan et al., 2024; Hanson et al., 2025b; Chen et al., 2026): **First, global ranking does not directly control how the retention budget is distributed across a scene.** Such methods score each Gaussian separately and retain the highest-scoring Gaussians globally (Fan et al., 2024; Hanson et al., 2025b; Chen et al., 2026). The number retained in each region therefore depends primarily on the score distribution. If Gaussians in certain regions consistently receive higher scores, the limited budget may concentrate there, leaving other regions with insufficient representational capacity. Thus,relying on a global ranking cannot guarantee that the rendering quality of all regions is well preserved under aggressive pruning Figure 1(a) illustrates this spatial quality imbalance: pruning baselines preserve some structures while losing detail in other regions of the same scene. **Second, Gaussian pruning under a fixed budget is a combinatorial optimization problem.** It requires selecting, subject to a retention constraint, a set of Gaussians that renders well together. Independent scoring followed by Top-K ranking reduces this set-selection problem to a comparison of individual scores. Yet multiple Gaussians may jointly support the same pixel or make overlapping contributions during rendering (Kerbl et al., 2023). The

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.

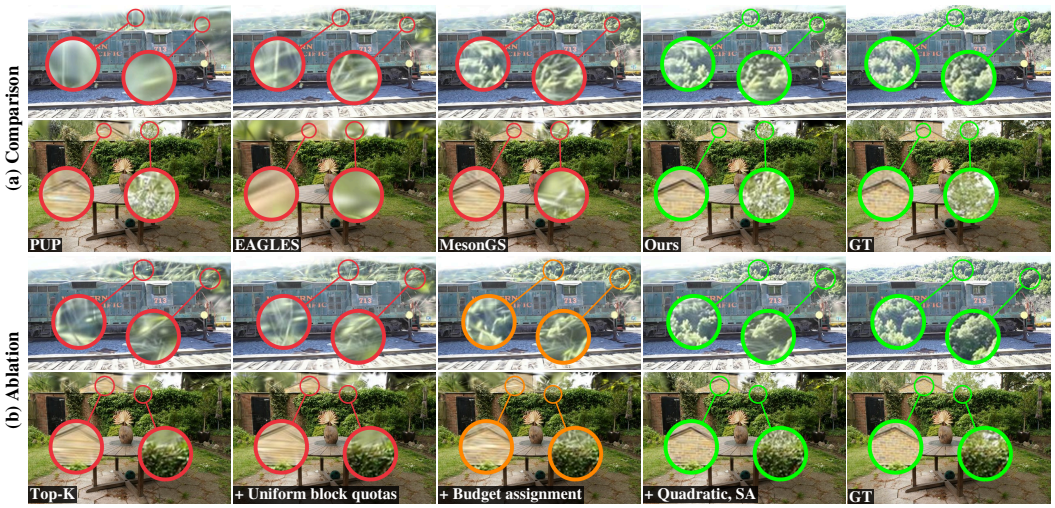

Figure 1: **Addressing spatial quality imbalance in 3DGS pruning.** (a) Comparisons with baselines illustrate uneven detail preservation across image regions, while SplatQ preserves the highlighted structures more consistently. (b) Component ablations show how block quotas, budget assignment, and quadratic selection progressively improve spatial quality balance over global Top-K pruning.

change caused by deleting one Gaussian cannot fully predict the effect of deleting it together with others. Two candidate sets may have similar individual deletion costs but produce different image changes when their members are deleted jointly. Consequently, accurate individual scores alone cannot determine which set of Gaussians should be retained. Ignoring interactions may prevent the limited budget from being used for a combination that achieves better reconstruction quality.

To address these limitations, we propose **SplatQ**, a post-training Gaussian pruning framework that performs similarity-guided partitioning, budget assignment, and interaction-aware quadratic selection in sequence. SplatQ aims to reduce disparities in local reconstruction quality under aggressive pruning and use the retained Gaussian budget more effectively, improving overall and local reconstruction quality with the same number of Gaussians. **First,** SplatQ partitions Gaussians into local blocks according to spatial proximity and appearance similarity. These blocks provide units for budget assignment while keeping the selection problems at a specific size. **Second,** to control local retention counts, SplatQ derives an initial quota for each block from importance estimates and adjusts the quota distribution to reduce the risk of concentrating the budget in a few blocks while leaving others with insufficient capacity. **Finally,** to account for the joint effects of deleting multiple Gaussians, SplatQ uses rendering responses to changes in opacity multipliers to estimate individual deletion effects and pairwise interactions. Figure 1(b) illustrates how block quotas, budget assignment, and quadratic selection progressively improve detail preservation across the highlighted regions. It then formulates and solves a QUBO (Glover et al., 2018) under each block's quota to select a retained set with a smaller estimated joint deletion effect. Because solving QUBOs with classical methods requires more computation than direct Top-K ranking, we also examine a coherent Ising machine (CIM) as a hardware solver backend and assess its potential to improve solving efficiency. After pruning, SplatQ fine-tunes the retained Gaussians to recover rendering quality.

We conduct extensive experiments on 27 scenes from 5 benchmarks covering indoor, outdoor, and objects, comparing SplatQ with eight open-source pruning methods. Under a common 90% pruning setting, SplatQ achieves the best overall image quality and rendering speed among the compared methods, while providing more uniform reconstruction quality across image regions. We further evaluate different pruning ratios, fine-tuning budgets, and combinations of pruning rounds and per-round fine-tuning iterations, facilitating comparisons with the reported settings of methods without released code. SplatQ also remains effective in a picture-free setting, where neither pruning nor fine-tuning uses the original training photographs and fine-tuning relies on images rendered by the fixed pretrained model. Even under this constraint, SplatQ surpasses the strongest baseline, MesonGS (Xie et al., 2024), in dataset-macro PSNR. Ablation studies further validate the effectiveness of partition-based budget assignment and interaction-aware combinatorial optimization, while experiments with CIM demonstrate that an alternative solver backend can also produce competitive selection results.

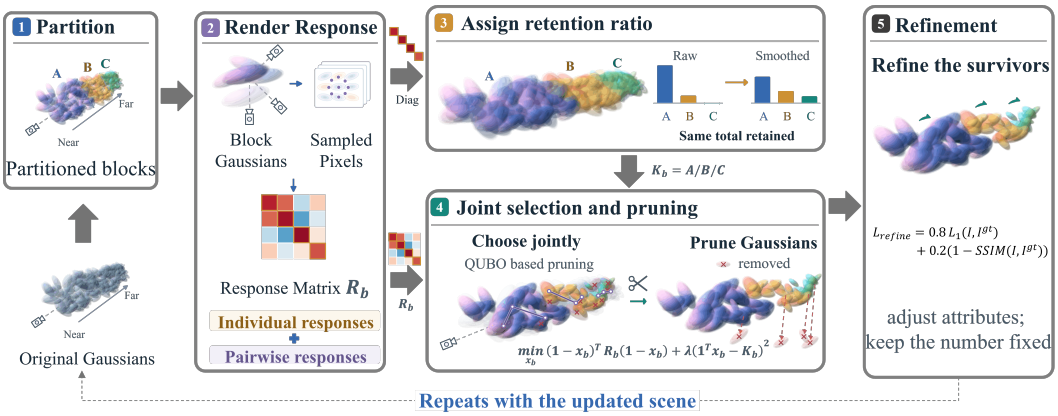

Figure 2: Overview of SplatQ. We partition Gaussians, compute a response matrix R_b for each block using sampled pixels from multiple views. Diagonal scores determine block retention quotas, while the full matrix is used to formulate a blockwise QUBO that determines which Gaussians to delete. The retained Gaussians are then fine-tuned without changing their count. For iterative pruning, the process repeats on the updated model.

2 Related Work

Gaussian compression and pruning. Existing 3DGS pruning methods often rank or sample Gaussians by rendering contributions, using compositing weights, coverage, volume, opacity, or gradients (Girish et al., 2024; Fan et al., 2024; Xie et al., 2024; Ali et al., 2024; Mousa & Maalouf, 2026). These criteria may overlook compensation by other Gaussians and undervalue sparsely observed regions. Other methods use parameter sensitivity or its approximations (Hanson et al., 2025b; Chen et al., 2026; Niedermayr et al., 2024), or estimate single-Gaussian deletion error relative to renderings of the current or original model (Lee et al., 2026; Ramlot et al., 2026). However, sensitivity is not equivalent to actual deletion error, and individual deletion errors do not directly characterize joint removal. Learned scores, masks, or sparse representations enable reconstruction-driven adaptation but require additional learning (Yang et al., 2026; Lee et al., 2024; Zhang et al., 2024; 2025; Liu et al., 2025b). Further methods consider relationships among Gaussians through neighboring appearance differences or local overlap (Lee et al., 2025; Jiang et al., 2026), or merge Gaussians into new representatives (Wang et al., 2025). In contrast, SplatQ partitions pretrained Gaussians by spatial and appearance similarity, initializes and smooths blockwise retention quotas under an exact global budget, and jointly selects Gaussians within each block using a quadratic objective built from individual and pairwise linearized rendering responses.

Balancing Rendering Quality Across Image Regions. NeRF++ (Zhang et al., 2020), Mip-NeRF 360 (Barron et al., 2022), and HoGS (Liu et al., 2025a) study how to represent near and distant content during scene reconstruction. For a pretrained 3DGS model, pruning additionally requires deciding how many primitives to retain in different parts of the scene under a limited Gaussian budget. PUP 3D-GS reports a trade-off between foreground detail and background quality under aggressive pruning (Hanson et al., 2025b). These regional differences motivate SplatQ to partition Gaussians with similar spatial positions and appearances into size-limited blocks, smooth the block quotas derived from importance estimates while preserving the exact global retention count, and perform joint selection within each block. In addition to overall reconstruction quality, we evaluate pruning results by measuring quality differences across image patches.

3 Method

Given a pretrained 3D Gaussian Splatting model with N Gaussians, SplatQ aims to retain exactly K Gaussians while accounting for their joint effects on the rendered image and reducing disparities in retention across scene regions. We represent the model as $\mathcal{G} = \{G_i = (\boldsymbol{\mu}_i, \mathbf{s}_i, \mathbf{r}_i, \mathbf{c}_i, \mathbf{h}_i, o_i)\}_{i=1}^{N}$ where each Gaussian is parameterized by its center $\boldsymbol{\mu}_i$, scale $\mathbf{s}_i$, rotation quaternion $\mathbf{r}_i$, zeroth-order (DC) and higher-order spherical-harmonic coefficients $\mathbf{c}_i$ and $\mathbf{h}_i$, and opacity o_i.

Previous methods suffer from two limitations: 1) They may produce imbalanced Gaussian retention across near and distant regions. 2) Moreover, independent Gaussian scores cannot capture the joint rendering effects of simultaneous deletions. SplatQ addresses these limitations through blockwise retention ratio allocation and interaction-aware quadratic selection, respectively.

SplatQ decomposes this global selection problem into five stages. Partitioning in Section 3.1 first converts Gaussians $\mathcal{G}$ into several size-limited blocks $\{\mathcal{C}_b\}$. Opacity response modeling in Section 3.2 then produces a response matrix R_b for each block. Budget allocation in Section 3.3 uses part of the matrix R_b to assign a retention quota K_b to every block, with $\sum_b K_b = K$. Given R_b and K_b, quadratic selection in Section 3.4 determines which K_b block members to retain. Finally, we fine-tune the retained attributes without changing the count of Gaussians in Section 3.5.

3.1 Similarity Guided Partition

Independent importance scores do not capture how the image changes when several Gaussians are deleted together. To better describe these changes, we approximate their joint rendering effects using a quadratic objective over binary retention variables. Each variable equals 1 if the corresponding Gaussian is retained and 0 if it is deleted, as detailed in Section 3.4. Formulating this optimization problem over the entire 3D-GS model requires N binary variables and an $N \times N$ matrix of pairwise coefficients. For models containing millions of Gaussians, the variable count and memory requirements make this approach impractical. We therefore divide the Gaussians into blocks of at most M members and solve a smaller pruning problem within each block.

Simply dividing Gaussians into consecutive blocks in their stored order ignores their spatial and appearance relationships. Nearby Gaussians with similar appearance may fall into different blocks, preventing their joint effects from being considered in the same pruning problem. We therefore use spatial proximity and appearance similarity to guide partitioning, so that potentially redundant Gaussians can be evaluated together. Finding related Gaussians through all pairwise comparisons itself requires quadratic work. We therefore use a 3D grid and Morton ordering to efficiently generate candidate pairs, whose spatial and appearance similarity is measured by an affinity.

Candidate generation. Gaussian centers are assigned to grid cells, and the occupied cells are ordered by their Morton order to preserve spatial proximity. Then we divide the resulting Gaussian sequence into groups of approximately equal size, each within a size limit of M. These groups are temporary search windows rather than the partition results $\mathcal{C}_b$.

Affinity graph. To encourage spatially and visually related Gaussians to be assigned to the same block, we define the following pairwise affinity. For Gaussian i, let $\boldsymbol{\mu}_i$ denote its center, $\mathbf{s}_i$ its activated scale vector, and $\mathbf{c}_i$ its three-channel DC spherical-harmonic descriptor. For each candidate pair, we define the scale-aware affinity as

$$a_{ij} = \exp\left[-\sum_{d=1}^{3} \frac{(\mu_{i,d} - \mu_{j,d})^2}{s_{i,d}^2 + s_{j,d}^2 + \epsilon_g} - \|\mathbf{c}_i - \mathbf{c}_j\|_2^2 \right]. \tag{1}$$

The first term measures center separation relative to the Gaussian scales and the second term reduces the affinity between spatially close Gaussians with different appearance. Each Gaussian retains at most the k candidates with the largest affinities, and the retained pairs form a sparse graph with a_{ij} as edge weights. The candidate groups depend on the grid origin, so one pass can miss pairs separated by a group boundary. We repeat candidate generation and affinity calculating with a shifted grid and merge both passes into one sparse graph. Repeated edges record their weights only once. The merged sparse graph then guides the construction of the final partition.

Partition. We next convert the sparse graph into the final size-limited blocks. Using Morton order in the first pass, we select one seed from each interval containing at most M Gaussians and grow all blocks in parallel. Each unassigned Gaussian proposes the block reached by its strongest edge to an assigned member, and proposals are accepted only while that block has capacity. Gaussians not reached by the graph fill the blocks in Morton order. This procedure yields disjoint blocks $\{\mathcal{C}_b\}_{b=1}^{B}$ with $n_b = |\mathcal{C}_b| \leq M$. These blocks serve as the units for response modeling, quota allocation, and quadratic selection in the following sections.

3.2 Opacity Response Modeling

The partition defines the local Gaussian sets, but it does not determine how many Gaussians to retain in each block or which ones to retain. For each block, we select pixels in rendered training views and calculate how their RGB values change when a block member's opacity changes. These measurements provide both individual importance scores for each block's budget allocation in Section 3.3 and pairwise interactions for quadratic selection of Gaussians in 3.4.

For each block b and camera view v, we use the projected Gaussian centers as the initially selected pixel coordinates. We augment this set with neighboring pixels obtained by offsetting each projected

center. After removing duplicate locations and discarding those outside the image, we collect the selected pixels p of this view v as pairs (v, p) into $\mathcal{P}_b$. These representative pixels are used to evaluate the RGB changes as Gaussian opacity changes.

Let L_{vp} denote the number of Gaussians covering pixel p in view v. We index these Gaussians in front-to-back order as $g_1, \ldots, g_{L_{vp}}$. Let α_ℓ denote the original alpha of g_ℓ, given by its activated opacity times its projected Gaussian value. We introduce a continuous multiplier $\eta_i \in [0, 1]$ for each Gaussian's activated opacity. The modified alpha and incoming transmittance are

$$\alpha_\ell(\boldsymbol{\eta}) = \eta_{g_\ell}\alpha_\ell, \qquad T_\ell(\boldsymbol{\eta}) = \prod_{r=1}^{\ell-1}\big(1 - \alpha_r(\boldsymbol{\eta})\big). \tag{2}$$

Using front-to-back alpha compositing, the rendered RGB channel $q \in \{1, 2, 3\}$ at this pixel is

$$I_{vpq}(\boldsymbol{\eta}) = \sum_{\ell=1}^{L_{vp}} T_\ell(\boldsymbol{\eta})\alpha_\ell(\boldsymbol{\eta})c_\ell + T_{L_{vp}+1}(\boldsymbol{\eta})c_q^{\mathrm{bg}}. \tag{3}$$

Here c_ℓ is the Gaussian's view-dependent color component, c_q^{bg} is the background color component, and $T_{L_{vp}+1}$ is the transmittance remaining after all contributors.

We differentiate each sampled RGB channel with respect to the opacity multiplier of every block member. For $i \in \mathcal{C}_b$, the corresponding entry of $J_b \in \mathbb{R}^{3|\mathcal{P}_b| \times n_b}$ is

$$(J_b)_{(v,p,q),i} = \left.\frac{\partial I_{vpq}(\boldsymbol{\eta})}{\partial \eta_i}\right|_{\boldsymbol{\eta}=\mathbf{1}}. \tag{4}$$

Rows correspond to sampled view-pixel-channel triples (v, p, q), and columns correspond to the Gaussians in block b. The derivatives define the opacity-response Gram matrix $R_b = J_b^{\mathsf{T}} J_b$ and diagonal score $D_i = (R_b)_{ii}, \quad i \in \mathcal{C}_b$. The diagonal score D_i sums the squared color derivatives for opacity of Gaussian i over all selected pixels, views, and RGB channels. A larger D_i therefore indicates that Gaussian i has a greater influence on the rendered image. Each off-diagonal entry is the signed inner product between two response columns. Positive entries indicate reinforcing linearized deletion effects, whereas negative entries indicate partial cancellation. It serves as the interactions used in Section 3.4.

3.3 Budget Assignment

This stage determines how many Gaussians each block retains. We use the diagonal scores D_i defined in Section 3.2 to assign each block a retention quota K_b, with $\sum_b K_b = K$. We first rank all Gaussians by D_i and count the global Top-K members contained in each block: $K_b = \big|\mathcal{C}_b \cap \mathrm{TopK}\big(\{D_i\}_{i=1}^N\big)\big|$. Top-$K$ returns exactly K indices, with equal scores resolved by global index. The resulting initial quotas therefore satisfy $\sum_b K_b = K$. Global Top-K selection may assign zero retained Gaussians to a block if none of its members ranks among the top K. Thus, some blocks may lose all their Gaussians. These blocks may include distant scene regions. To reduce this imbalance, We smooth the initial quotas to reduce disparities in retention across blocks. For initial quotas $0 < K < N$, the smoothed quotas are

$$r_b = \frac{K_b + \epsilon_a}{(K/N)n_b + \epsilon_a}, \qquad \widehat{K}_b = K\frac{n_b \log(1 + \beta r_b)}{\sum_{b'=1}^{B} n_{b'} \log(1 + \beta r_{b'})}. \tag{5}$$

Here r_b compares the initial quota with the proportional quota $(K/N)n_b$, while n_b denotes block capacity. Increasing $\beta > 0$ reduces disparities among block retention ratios. The pseudocount $\epsilon_a > 0$ handles blocks with zero initial quota. We round each continuous quota down and allocate the remaining units to eligible blocks in descending order of their fractional remainders. The final quotas satisfy $0 \le K_b \le n_b$ and $\sum_b K_b = K$ exactly.

3.4 Quadratic Selection

Budget allocation fixes the retained count K_b for each block. We now use R_b to determine which K_b members should be retained while accounting for their joint effects on the rendered image. Let $\mathbf{x}_b \in \{0, 1\}^{n_b}$ denote the block retention vector, where $x_{b,i} = 1$ indicates that the corresponding Gaussian is retained. We define the QUBO problem as

$$\min_{\mathbf{x}_b \in \{0,1\}^{n_b}} E_b(\mathbf{x}_b) = (\mathbf{1} - \mathbf{x}_b)^{\mathsf{T}} R_b (\mathbf{1} - \mathbf{x}_b) + \lambda \left(\mathbf{1}^{\mathsf{T}}\mathbf{x}_b - K_b\right)^2. \tag{6}$$

The first term approximates the rendering change caused by jointly deleting the Gaussians with $x_{b,i} = 0$. The second term penalizes deviations from the allocated quota K_b. Each block's QUBO

can be solved independently using greedy selection, simulated annealing, or an Ising solver. After repairing infeasible solver outputs when necessary, we combine the block masks into a model containing exactly K Gaussians. Appendix A.6 and Appendix A.7 provide the algebraic derivations and solver details.

3.5 Fine-Tuning and Iterative Pruning

We combine the result binary masks $\mathbf{x}_b$ from all blockwise QUBOs and fine-tune the retained Gaussians with densification disabled. The refinement loss is $\mathcal{L}_{\text{refine}} = 0.8\,\mathcal{L}_1(I, I^{\text{gt}}) + 0.2\big(1 - \text{SSIM}(I, I^{\text{gt}})\big)$, where I and I^{gt} are the rendered and ground-truth images. For iterative pruning, we repeat partitioning, opacity response computation, budget allocation, and quadratic selection on the refined model until the target retention ratio is reached.

4 Experiments

4.1 Experimental Settings

Datasets and metrics. We evaluate SplatQ on 27 scenes from 5 benchmarks: Mip-NeRF 360 (9), Deep Blending (2), Tanks and Temples (2), NeRF Synthetic (8), and Ref-NeRF/Shiny Blender (6). These benchmarks cover indoor scenes, outdoor scenes, and objects. We compare against unpruned 3DGS and eight pruning baselines, reporting average PSNR, SSIM, LPIPS-VGG, and rendering FPS. To assess spatial quality balance, we divide each test image into a 16×16 grid of non-overlapping patches and compute PSNR against the corresponding ground-truth region for each patch. For each image, we calculate the variance and range of its 256 patch PSNR values. We then average these statistics over test views within each scene and equally over scenes within each dataset. Considered alongside full-image PSNR, lower patch variance and range indicate smaller quality differences across image regions. Macro scores average the five dataset means equally. FPS is measured on Ubuntu 22.04 servers equipped with RTX 5090 GPUs and Intel Xeon Gold 6459C processors.

Experimental protocol. All methods start from the same 3DGS model pretrained for 30k iterations for each scene and follow a shared two-round pruning and fine-tuning schedule. The first round removes 80% of the original Gaussians, and the second removes 50% of the survivors, leaving 10% of the original Gaussians. Each round includes 10k fine-tuning iterations, for 20k in total. Baselines that originally did not use fine-tuning receive the same schedule. All methods use the refinement loss defined in Section 3.5, with densification disabled. We preserve each baseline's Gaussian importance estimation procedure. For methods that do not natively enforce an exact retention count, we apply Top-K selection using their importance scores to meet the target retention ratio. To facilitate comparison with reported results from methods without released code, we additionally evaluate pruning ratios from 10% to 90%, fine-tuning budgets from 0 to 30k iterations, and combinations of one to four pruning rounds with 5k to 30k fine-tuning iterations per round at 90% final removal. We also evaluate a *picture-free* setting mentioned in Section 1.

Ablation settings. Global Top-K retains the K Gaussians with the highest individual importance scores D_i across the entire model. Uniform block quotas assigns the same retention ratio to every block, whereas budget assignment uses our smoothed quotas. Both variants retain the highest-scoring Gaussians within each block according to D_i. The Quadratic variants use the full objective with smoothed quotas and are solved using greedy selection, simulated annealing (SA), or a coherent Ising machine (CIM). Quadratic, uniform quotas instead combines uniform block quotas with greedy selection. For each variant, we repeat the complete pruning and fine-tuning pipeline five times with random seeds 0 through 4, starting from the same pretrained models.

4.2 Comparison with Strong Baseline

Quality and rendering speed. As shown in Table 1, SplatQ achieves a macro-average PSNR of 28.426 dB, exceeding MesonGS and PUP by 0.224 dB and 0.300 dB, respectively. Among the compared pruning methods, it achieves the highest PSNR on four of the five datasets (MesonGS leads on Tanks and Temples), as well as the highest SSIM and lowest LPIPS-VGG on all five. The lower entry in each quality cell reports the picture-free result. Under this setting, SplatQ achieves macro-average PSNR, SSIM, and LPIPS-VGG values of 28.250 dB, 0.88216, and 0.17936, respectively, with its macro-average PSNR exceeding MesonGS by 0.238 dB. SplatQ also achieves the highest rendering FPS among the pruning methods on all five datasets. Figure 3 compares the Gaussian volume distributions of the nine pruning methods. At matched cumulative fractions, SplatQ generally yields smaller normalized Gaussian volumes. Details of volume normalization and cross-scene

aggregation are provided in Appendix B.1. Prior work shows that, for a fixed number of Gaussians, rendering cost is related to the number of screen-space pixels processed per Gaussian (Liu & Han, 2026; Hanson et al., 2025a; Feng et al., 2025). Thus, the smaller Gaussian volumes in SplatQ may correspond to more localized screen-space footprints and hence a lower rendering workload, providing a possible explanation for the observed FPS gains.

Spatial quality balance. To evaluate how effectively SplatQ alleviates regional imbalance in reconstruction quality, Figure 6 compares SplatQ (SA) with eight pruning baselines across five benchmarks. Alongside full-image PSNR, we use the variance and range of patch PSNR introduced in Section 4.1 to measure spatial variation in reconstruction quality. SplatQ (SA) achieves the lowest patch-PSNR variance and range on all five benchmarks, while attaining the highest full-image PSNR on four benchmarks and remaining close to the best result on Tanks and Temples. These results indicate that SplatQ reduces quality differences across regions while maintaining high overall reconstruction quality, thereby alleviating regional quality imbalance.

Sensitivity to pruning and fine-tuning. The pruning-ratio study evaluates removal ratios from 10% to 90% in increments of 10 percentage points, together with seven fine-tuning budgets ranging from 0 to 30,000 steps. The pruning-round study evaluates one to four rounds with 5,000–30,000 fine-tuning steps per round, at a final removal ratio of 90%. The main paper presents the 2D slice of pruning-ratio results at 10,000 fine-tuning steps (Figure 4) and the full 3D sweep over pruning rounds and fine-tuning steps (Figure 5). SplatQ outperforms the compared open-source pruning methods in reconstruction quality under the vast majority of evaluated configurations, demonstrating that its advantages extend across different pruning ratios, numbers of pruning rounds, and fine-tuning budgets. Under a few settings on Tanks and Temples, some quality metrics of SplatQ are slightly below those of the best-performing baseline. This suggests room for further improvement on datasets containing distant scene regions.

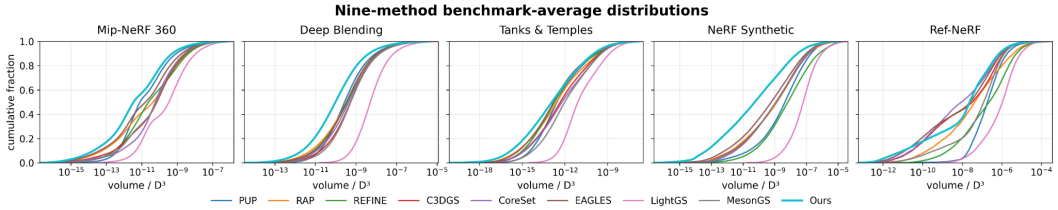

Figure 3: Normalized Gaussian-volume distributions across five benchmarks. Curves farther left indicate smaller volumes at the same cumulative fraction.

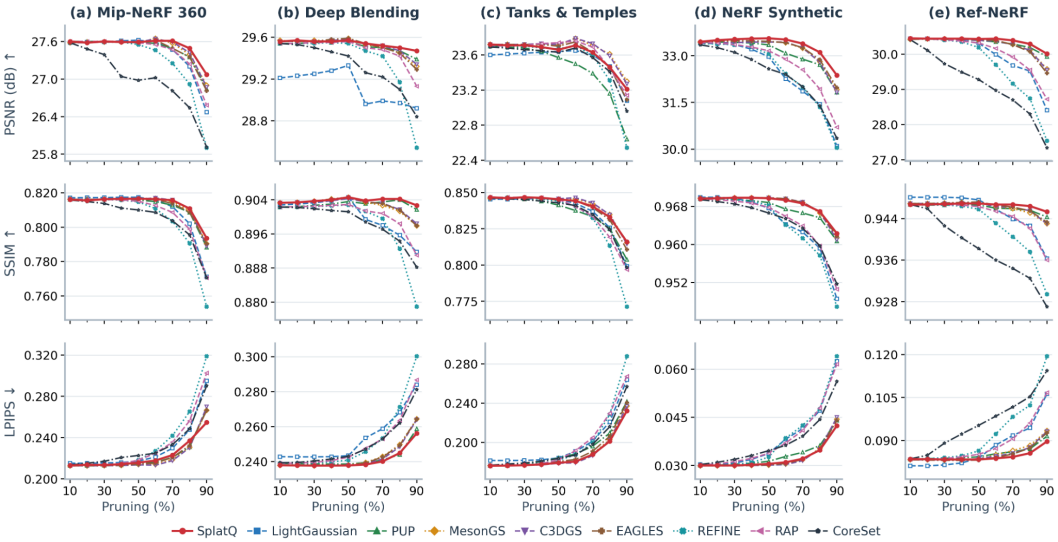

Figure 4: 2D pruning-percentage figure at 10k fine-tuning iterations. SplatQ is the SA variant.

Table 1: Main and picture-free comparisons at 90% pruning. Best among pruned methods is **bold** and second best is underlined.

Method	Mip-NeRF 360				Deep Blending				Tanks and Temples			
	PSNR↑	SSIM↑	LPIPS↓	FPS↑	PSNR↑	SSIM↑	LPIPS↓	FPS↑	PSNR↑	SSIM↑	LPIPS↓	FPS↑
3DGS	27.47	0.8123	0.2216	214.73	29.43	0.8984	0.2460	216.03	23.36	0.8378	0.1860	285.66
LightGaussian	26.47 26.17	0.7711 0.7671	0.2950 0.2994	600.25	28.92 28.99	0.8918 0.8926	0.2840 0.2828	792.80	23.08 22.80	0.7992 0.7973	0.2640 0.2624	1063.42
MesonGS	26.90 26.54	0.7903 0.7841	0.2665 0.2755	716.36	29.32 29.30	0.8979 0.8977	0.2644 0.2662	973.20	**23.30** 22.92	0.8154 0.8113	0.2371 0.2404	1286.88
EAGLES	26.81 26.43	0.7903 0.7841	0.2663 0.2753	707.75	29.28 29.29	0.8979 0.8977	0.2642 0.2664	1006.56	23.08 22.77	0.8107 0.8068	0.2405 0.2441	1254.02
C3DGS	26.87 26.47	0.7889 0.7817	0.2698 0.2797	747.66	29.34 29.34	0.8984 0.8982	0.2640 0.2658	1005.52	23.27 22.90	0.8143 0.8104	0.2369 0.2401	1299.13
PUP	26.84 26.39	0.7883 0.7837	0.2675 0.2754	896.26	29.39 29.37	0.9017 0.9022	0.2588 0.2572	1206.42	22.64 22.39	0.8038 0.7988	0.2416 0.2466	1452.20
RAP	26.58 26.05	0.7706 0.7591	0.3026 0.3163	781.16	29.13 29.00	0.8910 0.8883	0.2867 0.2931	1071.71	23.14 22.70	0.7969 0.7872	0.2674 0.2784	1392.78
REFINE	25.90 25.42	0.7536 0.7416	0.3188 0.3325	801.46	28.54 28.35	0.8789 0.8762	0.3003 0.3067	1078.79	22.54 22.13	0.7712 0.7616	0.2878 0.2982	1307.51
CoreSet	25.91 25.50	0.7705 0.7608	0.2903 0.3044	771.27	28.84 28.84	0.8882 0.8878	0.2812 0.2848	1005.39	22.96 22.58	0.7983 0.7906	0.2569 0.2649	1344.95
SplatQ (SA)	**27.07** **26.69**	**0.7932** **0.7870**	**0.2547** **0.2721**	**1022.23**	**29.47** **29.43**	**0.9027** **0.9028**	**0.2562** **0.2571**	**1232.28**	23.21 **22.93**	**0.8160** **0.8136**	**0.2322** **0.2349**	**1573.73**

Method	NeRF Synthetic				Ref-NeRF				Macro		
	PSNR↑	SSIM↑	LPIPS↓	FPS↑	PSNR↑	SSIM↑	LPIPS↓	FPS↑	PSNR↑	SSIM↑	LPIPS↓
3DGS	33.32	0.9695	0.0304	743.34	30.36	0.9470	0.0837	919.88	28.788	0.89300	0.15354
LightGaussian	30.10 30.18	0.9486 0.9494	0.0624 0.0613	2668.98	28.41 28.47	0.9363 0.9369	0.1063 0.1053	2392.75	27.396 27.322	0.86940 0.86866	0.20234 0.20224
MesonGS	31.91 31.84	0.9614 0.9613	0.0444 0.0445	3388.19	29.58 29.46	0.9430 0.9430	0.0934 0.0937	3002.43	28.202 28.012	0.88160 0.87948	0.18116 0.18406
EAGLES	31.96 31.89	0.9617 0.9615	0.0440 0.0445	3377.08	29.45 29.41	0.9432 0.9432	0.0927 0.0932	3060.99	28.116 27.958	0.88076 0.87866	0.18154 0.18470
C3DGS	31.82 31.75	0.9610 0.9609	0.0449 0.0452	3372.90	29.55 29.52	0.9430 0.9431	0.0931 0.0934	2984.86	28.170 27.996	0.88112 0.87886	0.18174 0.18484
PUP	31.83 31.63	0.9607 0.9595	0.0440 0.0456	3489.60	29.93 29.77	0.9443 0.9434	0.0915 0.0937	3026.85	28.126 27.910	0.87976 0.87752	0.18068 0.18370
RAP	30.70 30.26	0.9506 0.9476	0.0615 0.0653	3346.09	28.72 28.40	0.9360 0.9340	0.1067 0.1098	2740.48	27.654 27.282	0.86902 0.86324	0.20498 0.21258
REFINE	30.04 29.65	0.9469 0.9439	0.0640 0.0682	2985.89	27.54 27.07	0.9294 0.9263	0.1194 0.1247	2623.77	26.912 26.524	0.85600 0.84992	0.21806 0.22606
CoreSet	30.35 29.82	0.9517 0.9483	0.0562 0.0603	3294.38	27.34 26.89	0.9270 0.9249	0.1144 0.1178	2861.67	27.080 26.726	0.86714 0.86248	0.19980 0.20644
SplatQ (SA)	**32.37** **32.30**	**0.9623** **0.9623**	**0.0424** **0.0426**	**3595.66**	**30.01** **29.90**	**0.9453** **0.9451**	**0.0897** **0.0901**	**3180.37**	**28.426** **28.250**	**0.88390** **0.88216**	**0.17504** **0.17936**

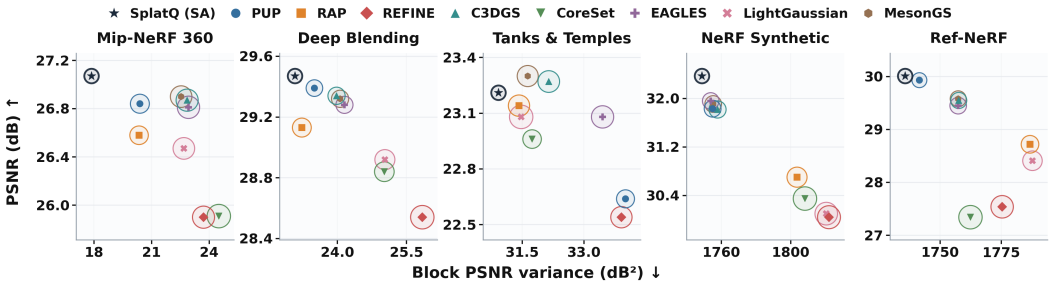

Figure 6: Overall reconstruction quality and spatial quality variation across five benchmarks. The horizontal axis shows the variance of PSNR over 256 image patches, and the vertical axis shows full-image PSNR. Bubble area represents the patch-PSNR range and is comparable only within each dataset. Higher positions, smaller horizontal values, and smaller bubbles indicate better results.

Qualitative Comparison Based on the results in Table 1, we select PUP, MesonGS, and EAGLES as three strong baselines for qualitative comparison. Figure 1(a) compares representative test views from SplatQ and these baselines. In these examples, SplatQ preserves detail more consistently across the highlighted regions, while several baselines show uneven degradation.

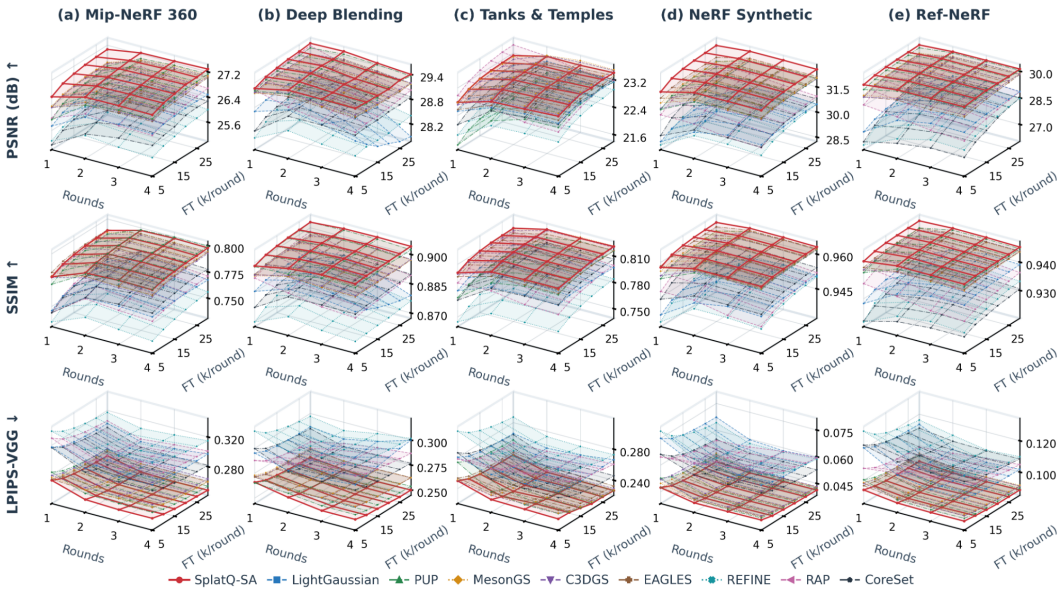

Figure 5: Pruning-round 3D figure at 90% final removal. SplatQ is the SA variant.

Table 2: Component and solver ablations at 90% pruning. Boldface marks SplatQ's default config.

Variant	Mip-NeRF 360			Deep Blending			Tanks and Temples		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Top-k	26.68	0.7787	0.2640	29.09	0.8944	0.2649	22.72	0.8034	0.2469
+ Uniform block quotas	26.52	0.7755	0.2685	28.98	0.8930	0.2668	22.64	0.8004	0.2505
+ Budget assignment	26.74	0.7889	0.2626	29.15	0.8958	0.2638	22.81	0.8050	0.2455
+ Quadratic, greedy	27.03	0.7928	0.2552	29.42	0.9020	0.2568	23.15	0.8135	0.2357
Quadratic, uniform quotas	26.79	0.7811	0.2602	29.36	0.8985	0.2585	23.07	0.8093	0.2424
+ Quadratic, SA	27.07	0.7932	0.2547	29.47	0.9027	0.2562	23.21	0.8160	0.2322
+ Quadratic, CIM	27.14	0.7960	0.2507	29.54	0.9041	0.2532	23.21	0.8173	0.2296

Variant	NeRF Synthetic			Ref-NeRF			Macro		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Top-k	31.85	0.9568	0.0503	29.55	0.9407	0.0972	27.978	0.87480	0.18466
+ Uniform block quotas	31.72	0.9557	0.0520	29.43	0.9396	0.0993	27.858	0.87284	0.18742
+ Budget assignment	31.96	0.9575	0.0496	29.64	0.9415	0.0961	28.060	0.87774	0.18352
+ Quadratic, greedy	32.31	0.9618	0.0427	29.96	0.9449	0.0906	28.374	0.88300	0.17620
Quadratic, uniform quotas	32.14	0.9596	0.0452	29.76	0.9431	0.0924	28.224	0.87832	0.17974
+ Quadratic, SA	32.37	0.9623	0.0424	30.01	0.9453	0.0897	28.426	0.88390	0.17504
+ Quadratic, CIM	32.48	0.9635	0.0409	30.07	0.9456	0.0887	28.488	0.88530	0.17262

4.3 Ablation Experiments

Component and solver ablations. Table 2 compares seven variants across datasets and reports their equally weighted macro-averages. Uniform block quotas lower macro PSNR from 27.978 to 27.858 dB relative to global first-order top-k; budget allocation raises it to 28.060 dB. Adding quadratic interactions with greedy selection yields 28.374 dB, versus 28.224 dB without budget allocation. The main SA variant achieves 28.426 dB, while CIM, reported separately, reaches 28.488 dB. Appendix B.2 compares CIM, SA, and greedy runtimes.

Qualitative ablations. Figure 1(b) illustrates the corresponding changes in local reconstruction quality. Uniform block quotas alone do not consistently recover the highlighted details. As budget allocation and quadratic selection with SA are progressively introduced, reconstruction quality improves in these examples.

5 Conclusion

We presented SplatQ, which combines regional budget allocation with interaction-aware quadratic selection for 3DGS pruning. Across five benchmarks at 90% pruning, SplatQ improves aggregate reconstruction quality, spatial quality balance, and rendering speed over the compared methods. Component ablations support the benefit of pairwise interactions before fine-tuning, while CIM provides a competitive alternative to SA for solving the selection problems.

AI USE STATEMENT

Generative AI tools assisted with manuscript drafting, language editing and translation, figure and table preparation, and LaTeX formatting. All reported numerical results originate from actual experimental records, not AI-generated data. The authors take responsibility for the final content and accuracy of the paper.

ETHICS STATEMENT

Our experiments use only existing public benchmark datasets. Potential downstream concerns include unauthorized sharing of reconstructed private spaces and loss of important visual details under aggressive pruning. Neither compression nor the picture-free setting provides a privacy guarantee. Use and redistribution of scene assets should respect applicable licenses and privacy requirements.

REPRODUCIBILITY STATEMENT

Section 3 and Appendix A describe the method, derivations, pseudocode, and solver postprocessing. Appendix B.1 specifies the datasets, baseline adaptations, hyperparameters, random seeds, fine-tuning and evaluation protocols, and hardware. Table 2 reports means and standard deviations across five complete-pipeline runs. Appendix B.2 provides computational costs and additional ablations. We plan to publicly release the source code.

REFERENCES

Muhammad Salman Ali, Maryam Qamar, Sung-Ho Bae, and Enzo Tartaglione. Trimming the fat: Efficient compression of 3D gaussian splats through pruning. In *BMVC*, 2024. URL https://bmva-archive.org.uk/bmvc/2024/papers/Paper_358/paper.pdf.

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, pp. 5470–5479, 2022. URL https://jonbarron.info/mipnerf360/.

Zhang Chen, Shuai Wan, Mengting Yu, Fuzheng Yang, and Junhui Hou. REFINE: Super-efficient 3D gaussian splatting pruning via rendering-free primitive importance. In *ECCV*, volume 17027 of *LNCS*, pp. 131–147. Springer, 2026. doi: 10.1007/978-3-032-37602-2_8.

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 *NeurIPS*, volume 37, pp. 140138–140158, 2024.

Guofeng Feng, Siyan Chen, Rong Fu, Zimu Liao, Yi Wang, Tao Liu, Boni Hu, Linning Xu, Zhilin Pei, Hengjie Li, Xiuhong Li, Ninghui Sun, Xingcheng Zhang, and Bo Dai. FlashGS: Efficient 3D Gaussian Splatting for large-scale and high-resolution rendering. In *CVPR*, pp. 26652–26662, 2025.

Sharath Girish, Kamal Gupta, and Abhinav Shrivastava. EAGLES: Efficient accelerated 3D gaussians with lightweight encodings. In *ECCV*, volume 15121 of *LNCS*, pp. 54–71, 2024. doi: 10.1007/978-3-031-73036-8_4.

Fred Glover, Gary Kochenberger, and Yu Du. A tutorial on formulating and using QUBO models. *arXiv preprint arXiv:1811.11538*, 2018. URL https://arxiv.org/abs/1811.11538.

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 *CVPR*, pp. 21537–21546, 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 *CVPR*, pp. 5949–5958, 2025b.

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM TOG*, 37(6), 2018. doi: 10.1145/3272127.3275084. URL http://visual.cs.ucl.ac.uk/pubs/deepblending/.

Shizeng Jiang, Hao Zhang, Xuerui Ma, Ying Hu, and Tao Zhang. QIRF: Quantum-inspired non-orthogonal function-space compression for 3D gaussian splatting, 2026. URL https://arxiv.org/abs/2607.18067.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for real-time radiance field rendering. *ACM TOG*, 42(4), 2023. doi: 10.1145/3592433. URL https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/.

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM TOG*, 36(4), 2017. doi: 10.1145/3072959.3073599. URL https://www.tanksandtemples.org/.

Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3D gaussian representation for radiance field. In *CVPR*, pp. 21719–21728, 2024.

Joo Chan Lee, Jong Hwan Ko, and Eunbyung Park. Optimized minimal 3D gaussian splatting. In *NeurIPS*, volume 38, 2025. doi: 10.52202/085713-4534.

Soonbin Lee, Yeong-Gyu Kim, Simon Sasse, Tomás M. Borges, Yago Sánchez, Eun-Seok Ryu, Thomas Schierl, and Cornelius Hellge. GaussianPOP: Principled simplification framework for compact 3D gaussian splatting via error quantification, 2026. URL https://arxiv.org/abs/2602.06830.

Jiaqi Liu and Zhizhong Han. Speeding up the learning of 3D gaussians with much shorter gaussian lists. In *CVPR*, 2026.

Xinpeng Liu, Zeyi Huang, Fumio Okura, and Yasuyuki Matsushita. HoGS: Unified near and far object reconstruction via homogeneous gaussian splatting. In *CVPR*, pp. 26714–26722, 2025a. URL https://arxiv.org/abs/2503.19232.

Yifei Liu, Zhihang Zhong, Yifan Zhan, Sheng Xu, and Xiao Sun. MaskGaussian: Adaptive 3D gaussian representation from probabilistic masks. In *CVPR*, pp. 681–690, 2025b.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. URL https://arxiv.org/abs/2003.08934.

Waseem Mousa and Alaa Maalouf. Provable pruning for efficient 3D gaussian splatting via coresets, 2026. URL https://arxiv.org/abs/2607.02721.

Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3D gaussian splatting for accelerated novel view synthesis. In *CVPR*, pp. 10349–10358, 2024.

Bert Ramlot, Martijn Courteaux, Peter Lambert, and Glenn Van Wallendael. POTR: Post-training 3DGS compression. *IEEE TCSVT*, 36(8):12168–12181, 2026. doi: 10.1109/TCSVT.2026.3685779.

Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. In *CVPR*, 2022. URL https://arxiv.org/abs/2112.03907.

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 *NeurIPS*, volume 38, 2025. doi: 10.52202/085713-5277.

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 *ECCV*, pp. 434–452, 2024. doi: 10.1007/978-3-031-73414-4_25.

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 *CVPR*, pp. 33323–33332, 2026.

Kai Zhang, Gernot Riegler, Noah Snavely, and Vladlen Koltun. NeRF++: Analyzing and improving neural radiance fields. *arXiv preprint arXiv:2010.07492*, 2020. URL https://arxiv.org/abs/2010.07492.

Yangming Zhang, Wenqi Jia, Wei Niu, and Miao Yin. GaussianSpa: An "Optimizing-Sparsifying" simplification framework for compact and high-quality 3D gaussian splatting. In *CVPR*, pp. 26673–26682, 2025.

Zhaoliang Zhang, Tianchen Song, Yongjae Lee, Li Yang, Cheng Peng, Rama Chellappa, and Deliang Fan. LP-3DGS: Learning to prune 3D gaussian splatting. In *NeurIPS*, volume 37, 2024. doi: 10.52202/079017-3891.

A METHOD DERIVATIONS AND IMPLEMENTATION DETAILS

A.1 NOTATION

We use the notation of Section 3. The current model has N Gaussians, partitioned into disjoint blocks $\mathcal{C}_b$ of size $n_b \le M$. The target retained count is K, and the final block budgets satisfy $\sum_b K_b = K$. During allocation, K_b is updated from its initial Top-K count to its final integer budget.

For each block, $\mathcal{P}_b$ contains selected view-pixel pairs, and $m_b = 3|\mathcal{P}_b|$ denotes the number of sampled color channels. Thus $J_b \in \mathbb{R}^{m_b \times n_b}$ and $R_b = J_b^\mathsf{T} J_b \in \mathbb{R}^{n_b \times n_b}$.

The depth index ℓ is distinct from Gaussian identity i. For a fixed view and pixel, g_ℓ denotes the global ID of the Gaussian at depth rank ℓ. In particular, $\alpha_\ell(\boldsymbol{\eta}) = \eta_{g_\ell}\alpha_\ell$. The reference for response evaluation is the current model at each pruning round, not a fixed model shared by all rounds.

Table 3 collects these main symbols.

Table 3: Principal method notation. Other scalar parameters are defined at their first use.

Symbol	Meaning
$\mathcal{G}, N, K$	Current Gaussian model, its Gaussian count, and target retained count for the current pruning round.
$\mathbf{x}_b, \mathbf{z}_b$	Block retention and deletion vectors in $\{0,1\}^{n_b}$, with $\mathbf{z}_b = \mathbf{1} - \mathbf{x}_b$. Subscripts are omitted in local derivations.
$\mathcal{C}_b, n_b, M$	Set of Gaussian indices in block b, its size $n_b = \lvert\mathcal{C}_b\rvert$, and maximum allowed block size.
B, K_b	Number of blocks and block retention budget, updated from the initial Top-K count to the final integer budget.
$\boldsymbol{\eta}$	Continuous opacity multipliers in $[0,1]^N$.
v, p, q	Camera index, pixel index, and RGB channel index.
L_{vp}, g_ℓ	Number of Gaussians covering pixel p in view v, and the global Gaussian ID at depth rank ℓ for that pixel.
$I_{vpq}(\boldsymbol{\eta})$	Rendered RGB channel q at pixel p in view v, evaluated with opacity multipliers $\boldsymbol{\eta}$ in the full scene.
$\mathcal{P}_b, m_b$	Selected view-pixel pairs for block b, and the sampled channel count $m_b = 3\lvert\mathcal{P}_b\rvert$.
J_b, R_b	Sampled opacity Jacobian in $\mathbb{R}^{m_b \times n_b}$ and its Gram matrix $R_b = J_b^\mathsf{T} J_b \in \mathbb{R}^{n_b \times n_b}$.
D_i	Allocation score $(R_b)_{ii}$ for $i \in \mathcal{C}_b$, equal to the summed squared opacity responses over sampled channels.
$\widehat{K}_b$	Continuous block quota from logarithmic smoothing.
ϵ_a, β	Pseudocount for handling zero initial budgets, and logarithmic smoothing parameter.
λ	Coefficient of the count penalty $(\mathbf{1}^\mathsf{T}\mathbf{x}_b - K_b)^2$.
I, I^{gt}	Rendered image and supervision image for a training view, used in the refinement loss.

A.2 SAMPLING AND GRAM ACCUMULATION

For each block and selected training view, we generate candidate pixels only from Gaussians whose projected radii reported by the rasterizer are positive. Let $\mathbf{u}_{vi}$ be a projected center and $h_{vi} > 0$ its radius in pixels. The candidate locations are

$$\mathbf{u}_{vi} + \max(h_{vi}, 1)\boldsymbol{\delta}, \qquad \boldsymbol{\delta} \in \mathcal{D},$$
$$\mathcal{D} = \{(0,0), (\pm 0.5, 0), (0, \pm 0.5), (\pm 0.35, \pm 0.35)\}. \tag{7}$$

This yields the projected center and eight neighboring locations for each Gaussian. We round their coordinates to the nearest integer pixels and discard locations outside the image boundaries. For each block, we remove duplicate pixels within each view and collect the selected pairs (v, p) across views into $\mathcal{P}_b$.

Responses are evaluated through the full scene rendering, with the Jacobian columns restricted to block members. Gaussians outside the block and the background therefore remain part of each response. Partitioning the sampled rows into batches gives

$$R_b = \sum_{t=1}^{T_b} (J_b^{[t]})^\mathsf{T} J_b^{[t]}, \qquad D_i = (R_b)_{ii} = \sum_{(v,p)\in\mathcal{P}_b} \sum_{q=1}^{3} (J_b)^2_{(v,p,q),i}. \tag{8}$$

Algorithm 1 Similarity guided partitioning

Require: Gaussians $\mathcal{G}$; block limit M; neighbors k; comparison-group size limit
Ensure: Disjoint blocks $\{\mathcal{C}_b\}_{b=1}^{B}$ covering all N Gaussians
1. Build the graph
for two spatial passes **do**
  Assign centers $\boldsymbol{\mu}_i$ to grid cells and sort cells by Morton code
  Split the sequence into the fewest balanced groups satisfying the comparison limit
  for each comparison group **do**
    Compute pairwise affinities a_{ij}
    Keep each Gaussian's top-k edges
  end for
  Shift the grid origin and enlarge the cell size for the second pass
end for
Merge edges from both passes, keep weight of repeated edges once
2. Grow blocks in parallel
Split the first-pass Morton sequence into $\lceil N/M \rceil$ balanced intervals,
seed each block with its first Gaussian, and leave the others unassigned.
for at most 16 iterations **do**
  Each unassigned Gaussian proposes the block of its strongest assigned neighbor
  for each block $\mathcal{C}_b$, in parallel **do**
    Rank proposals by descending edge weight, then ascending Gaussian index
    Add the first $\min(\text{number of proposals}, n_b - |\mathcal{C}_b|)$ Gaussians
  end for
  Stop if no Gaussian was assigned in this iteration
end for
3. Fill unused capacity
Assign remaining Gaussians to fill each block to n_b in the first-pass Morton order.
return $\{\mathcal{C}_b\}_{b=1}^{B}$

Algorithm 2 Representative pixel selection

Require: Block $\mathcal{C}_b$ and selected training views
Ensure: Sampled view-pixel pairs $\mathcal{P}_b$
$\mathcal{P}_b \leftarrow \varnothing$
$\mathcal{D} \leftarrow \{(0,0), (\pm 0.5, 0), (0, \pm 0.5), (\pm 0.35, \pm 0.35)\}$
for each selected view v **do**
  Obtain projected centers $\mathbf{u}_{vi}$ and screen radii h_{vi} for $i \in \mathcal{C}_b$
  Generate $p = \mathrm{round}(\mathbf{u}_{vi} + \max(h_{vi}, 1)\boldsymbol{\delta})$ for $h_{vi} > 0$ and $\boldsymbol{\delta} \in \mathcal{D}$
  Remove duplicate pixels and discard those outside the image
  Add the remaining view-pixel pairs (v, p) to $\mathcal{P}_b$
end for
return $\mathcal{P}_b$

Here T_b is the number of batches, and each batch contains all three channels of its sampled pixels. Batch accumulation equals the full Gram product up to floating-point rounding and does not require storing the full Jacobian. If no valid pixels are sampled across the selected views, the zero-initialized matrix and its diagonal scores remain zero.

A.3 Derivation of the opacity response

For a fixed view v, pixel p, and color channel q, consider Gaussian g_ℓ at depth rank ℓ. We keep the projected footprints, depth order, and all other opacity multipliers fixed. We then rewrite Equation (3) as the sum of contributions from g_ℓ, the Gaussians in front of it, and the content behind it:

$$I_{vpq}(\boldsymbol{\eta}) = A^{<\ell} + T_\ell \left[\alpha_\ell(\boldsymbol{\eta}) c_\ell + (1 - \alpha_\ell(\boldsymbol{\eta})) C_\ell^{\mathrm{back}}\right]. \tag{9}$$

Here $A^{<\ell}$ is the foreground contribution and C_ℓ^{back} is the suffix color composited with unit incoming transmittance. Neither term, nor T_ℓ, depends on η_{g_ℓ}. At the reference model, the suffix is

$$C_\ell^{\mathrm{back}} = \sum_{r=\ell+1}^{L_{vp}} \left[\prod_{u=\ell+1}^{r-1} (1 - \alpha_u)\right] \alpha_r c_r + \left[\prod_{u=\ell+1}^{L_{vp}} (1 - \alpha_u)\right] c_q^{\mathrm{bg}}. \tag{10}$$

Empty products equal one, so the suffix after the last Gaussian is the background color. Since $\partial\alpha_\ell(\boldsymbol{\eta})/\partial\eta_{g_\ell} = \alpha_\ell$, for $i = g_\ell \in \mathcal{C}_b$ we obtain

$$(J_b)_{(v,p,q),i} = \left.\frac{\partial I_{vpq}(\boldsymbol{\eta})}{\partial \eta_i}\right|_{\boldsymbol{\eta}=\mathbf{1}} = T_\ell \alpha_\ell (c_\ell - C_\ell^{\text{back}}). \tag{11}$$

All quantities on the right are evaluated at the reference model. The first color term is the Gaussian's direct response; the suffix term captures the change in visibility of everything behind it.

When the projected footprints remain fixed and alpha values are not clamped, the rendered channel is an affine function of each opacity multiplier, with all others held fixed. A single deletion is therefore represented exactly by its derivative under these assumptions. Multiple deletions introduce products of multiplier changes through transmittance, so their combined first-order response is generally approximate. In practice, alpha clamping, contribution thresholds, and early ray termination can invalidate the exact correspondence between the derivative and the effect of deleting one Gaussian.

A.4 Capacity handling

Equation (5) produces continuous quotas summing to K. To enforce capacities during integer conversion, the implementation first clips them and initializes integer budgets:

$$\widetilde{K}_b = \min\{n_b, \max(0, \widehat{K}_b)\}, \qquad K_b \leftarrow \lfloor \widetilde{K}_b \rfloor. \tag{12}$$

It then recomputes the remaining budget and residuals as

$$H = K - \sum_b K_b, \qquad f_b = \widetilde{K}_b - K_b. \tag{13}$$

While $H > 0$, eligible blocks are those with $K_b < n_b$. They are sorted by descending f_b, with ties broken by block index. Each of the first $\min(H, \text{number of eligible blocks})$ blocks receives one unit. The remaining budget and residuals are then updated, and the procedure repeats until $H = 0$.

If no quota is clipped, one pass reduces to largest-remainder rounding. Clipping can create a larger deficit, requiring multiple passes. Each increment respects capacity and reduces the remaining deficit. The procedure therefore terminates with $0 \le K_b \le n_b$ and $\sum_b K_b = K$. For $K = 0$ or disabled smoothing, the initial budgets are returned unchanged.

A.5 Joint deletion

Let $\mathbf{f}_b(\boldsymbol{\xi}) \in \mathbb{R}^{m_b}$ stack the sampled RGB channels when block-member opacity multipliers are $\boldsymbol{\xi} \in \mathbb{R}^{n_b}$. All multipliers outside the block remain one. For a small perturbation $\mathbf{u}$,

$$\mathbf{f}_b(\mathbf{1} - \mathbf{u}) - \mathbf{f}_b(\mathbf{1}) = -J_b\mathbf{u} + O(\|\mathbf{u}\|_2^2). \tag{14}$$

The squared magnitude of its linear term is

$$\|J_b\mathbf{u}\|_2^2 = \mathbf{u}^\mathsf{T} R_b \mathbf{u}. \tag{15}$$

With binary deletion variables $\mathbf{z}_b = \mathbf{1} - \mathbf{x}_b$, the approximate deletion cost is:

$$E_{\text{del}}(\mathbf{z}_b) = \mathbf{z}_b^\mathsf{T} R_b \mathbf{z}_b = \sum_i (R_b)_{ii} z_{b,i} + 2\sum_{i<j} (R_b)_{ij} z_{b,i} z_{b,j}. \tag{16}$$

The diagonal is the individual squared response. The off-diagonal terms measure reinforcement or cancellation between response columns. Substituting $\mathbf{z}_b = \mathbf{1} - \mathbf{x}_b$ and adding the count penalty yields Equation (6) in the main text.

Our objective approximates changes from the current rendering rather than error relative to ground truth. This approximation may become less accurate when many Gaussians are deleted together. We therefore recompute the responses after model refinement in each subsequent pruning round.

A.6 QUBO coefficients and the Ising mapping

Within this subsection, write $n = n_b$, $K = K_b$ and $R = R_b$ for local quantities. Expanding Equation (6) and using $x_i^2 = x_i$ gives

$$\begin{aligned} E(\mathbf{x}) &= C + \sum_i a_i x_i + \sum_{i<j} b_{ij} x_i x_j, \\ a_i &= R_{ii} - 2\sum_j R_{ij} + \lambda(1 - 2K), \\ b_{ij} &= 2R_{ij} + 2\lambda, \qquad C = \mathbf{1}^\mathsf{T} R \mathbf{1} + \lambda K^2. \end{aligned} \tag{17}$$

The constant does not affect selection. We store the rendering objective as a vector of linear coefficients and a sparse list of quadratic coefficients. Each pair (i, j) is stored once with $i < j$.

For spins $\sigma_i \in \{-1, +1\}$, substitute $x_i = (1 + \sigma_i)/2$:

$$E(\boldsymbol{\sigma}) = C' + \sum_i h_i \sigma_i + \sum_{i<j} \mathcal{J}_{ij} \sigma_i \sigma_j,$$
$$\mathcal{J}_{ij} = b_{ij}/4, \qquad h_i = a_i/2 + \tfrac{1}{4} \sum_{j \neq i} b_{ij},$$
$$C' = C + \tfrac{1}{2} \sum_i a_i + \tfrac{1}{4} \sum_{i<j} b_{ij}. \tag{18}$$

Here $b_{ji} = b_{ij}$ when evaluating the field sum. For the backend convention $H(\boldsymbol{\sigma}) = -\boldsymbol{\sigma}^\mathsf{T} W \boldsymbol{\sigma}$, the symmetric pair entries are $W_{ij} = -\mathcal{J}_{ij}/2$. Fields are encoded using one reference spin σ_0, with $W_{i0} = W_{0i} = -h_i/2$ and zero diagonal entries. Decoding uses $x_i = (1 + \sigma_i \sigma_0)/2$, so an n-variable block requires $n + 1$ spins in this encoding.

A.7 Greedy selection, Simulated Annealing and CIM Solvers

Again use local R, n, K and deletion cost $E_{\mathrm{del}}(\mathbf{z}) = \mathbf{z}^\mathsf{T} R \mathbf{z}$.

Greedy selection. Starting with $\mathbf{z} = \mathbf{0}$, deleting a retained member j changes the cost by

$$\Delta_{\mathrm{add}}(j) = 2(R\mathbf{z})_j + R_{jj}. \tag{19}$$

Greedy selection takes the smallest increment at each step until exactly $n - K$ members are deleted. It satisfies the count constraint by construction and does not require a count penalty.

Simulated annealing. Simulated annealing (SA) minimizes the penalized objective with the count-penalty coefficient $\lambda = 1$, using the Ising encoding in Appendix A.6. We initialize multiple random spin configurations and apply Metropolis updates while reducing the temperature geometrically. For a proposed spin flip with energy change ΔH, the acceptance probability at temperature $\theta > 0$ is

$$p_{\mathrm{accept}} = \min\{1, \exp(-\Delta H/\theta)\}. \tag{20}$$

Energy decreases are always accepted, whereas temporary increases allow the search to leave local minima. Lower temperatures reduce the probability of accepting such increases. By default, the initial and final temperatures are 100 and 10^{-3}, respectively, with a cooling factor of 0.99. We use 10 iterations per temperature and a solution limit of 100 per block. We select the returned state with the lowest Ising energy and decode its retention variables using the reference spin.

Coherent Ising Machine (CIM). The coherent Ising machine (CIM) uses the same Ising formulation and reference-spin encoding as SA, but with a different count-penalty coefficient, $\lambda = 0.0625$. Before submission, we rescale and truncate the matrix coefficients to a signed 14-bit representation. We request multiple samples and select the returned state with the lowest energy under the submitted matrix. The selected state is then decoded into binary retention variables. Neither SA nor CIM guarantees an exact retained count, we therefore apply the following postprocessing.

A.8 Postprocessing

Capacity repairing. For an infeasible solver output, count repair changes one bit at a time. If too many Gaussians are retained, it deletes the retained member with the smallest increase in unpenalized energy. If too few are retained, it restores the deleted member with the smallest increase. The coefficients and current state determine each increment exactly, and the process stops when the retained count equals K. Repair can increase the unpenalized cost because feasibility takes priority.

One-swap refinement. For a deleted member i and retained member j, a one-swap move restores i and deletes j. Its energy change is

$$\Delta_{\mathrm{swap}}(i, j) = 2\big[(R\mathbf{z})_j - (R\mathbf{z})_i\big] + R_{ii} + R_{jj} - 2R_{ij}. \tag{21}$$

When enabled, refinement repeatedly selects the best exchange and accepts it only when $\Delta_{\mathrm{swap}} < -\tau$, for a numerical tolerance $\tau \geq 0$. It stops when no qualifying exchange remains or the configured swap limit is reached. Every accepted swap preserves the count and decreases the objective. Without an active iteration limit, termination gives a one-swap local optimum within tolerance. For $K = 0$ or $K = n$, the feasible state is unique and no search is needed.

We use one-swap refinement as a shared local-search step rather than a separate methodological contribution. The same refinement rule and stopping criteria are applied after greedy selection, SA,

Table 4: Scoring time, solver time, and peak GPU memory across three datasets. Times are reported in seconds and memory in GB. For SplatQ, scoring denotes R_b calculation. A dash indicates that no separate QUBO solver is used.

	Mip-NeRF 360			Deep Blending			Tanks & Temples		
Method	Scoring	Solving	Memory	Scoring	Solving	Memory	Scoring	Solving	Memory
LightGS	4.63	–	9.91	5.26	–	7.74	6.12	–	5.52
MesonGS	6.17	–	7.64	20.34	–	6.50	8.59	–	4.02
EAGLES	1.10	–	8.36	1.12	–	7.80	1.01	–	5.00
C3DGS	4.37	–	10.05	4.31	–	8.48	3.24	–	5.47
PUP	17.5	–	10.1	16.3	–	8.7	20.1	–	5.5
REFINE	0.26	–	1.5	0.25	–	1.4	0.27	–	1.3
RAP	89.4	–	16.4	88.5	–	14.6	41.7	–	9.2
CoreSet	1240.4	–	4.0	876.3	–	4.4	638.9	–	4.1
SplatQ (greedy)	1005.0	101.1	10.41	1237.1	99.1	8.56	3456.3	58.9	5.45
SplatQ (SA)	1005.0	845.9	10.41	1237.1	748.4	8.56	3456.3	448.9	5.45
SplatQ (CIM)	1005.0	581.4	10.41	1237.1	514.5	8.56	3456.3	309.6	5.45

and CIM. We therefore keep this step fixed in the solver comparisons, which evaluate each solver together with the shared postprocessing.

B Experiment

This section supplements the experimental results in Section 4.

B.1 Experiment Settings

Datasets and Metrics We evaluate 27 scenes from five benchmarks: Mip-NeRF 360 (9), Deep Blending (2), Tanks and Temples (2), NeRF Synthetic (8), and Ref-NeRF/Shiny Blender (6) (Barron et al., 2022; Hedman et al., 2018; Knapitsch et al., 2017; Mildenhall et al., 2020; Verbin et al., 2022). We compare against the unpruned 3DGS reference and eight pruning baselines, reporting dataset-average PSNR, SSIM, LPIPS-VGG, and measured rendering FPS. Macro scores average the five dataset means equally.

Evaluation settings. FPS measurements were conducted on an Ubuntu 22.04 server equipped with NVIDIA GeForce RTX 5090 GPUs and Intel Xeon Gold 6459C processors. The inspected evaluation pipeline saves test-view renderings and computes PSNR, SSIM, and LPIPS-VGG. The FPS testing procedure warms up with ten test views and times three passes over all test views.

Implementation Details. For each scene, all pruning methods start from the same 3DGS model pretrained for 30k iterations and retain 10% of its original Gaussians. All methods follow the same two-round schedule: removing 80% of the original Gaussians, then 50% of the survivors. Each round includes 10k fine-tuning iterations, for 20k in total, using the same loss $0.8\mathcal{L}_1 + 0.2(1 - \mathrm{SSIM})$ with densification disabled. We adapt the baselines by retaining their Gaussian selection mechanisms within this shared pruning and fine-tuning pipeline. Methods originally evaluated without fine-tuning also receive the same fine-tuning schedule, and attribute compression is disabled for all methods. SplatQ uses simulated annealing (SA) in the main comparison, with greedy and CIM evaluated as solver ablations. Further details are provided in Appendix A.1 and Table 5.

Ablation settings. Table 6 compares budget assignment and selection variants. Top-k globally retains the K Gaussians with the largest diagonal scores D_i. Uniform block quotas assigns the same retention fraction to all blocks, whereas Budget allocation uses our logarithmically smoothed quotas. Both variants select the highest-scoring Gaussians within each block using D_i. The Quadratic variants use the full objective with smoothed quotas and are solved using greedy selection, SA, or CIM. Quadratic, no allocation variant instead uses uniform block quotas and greedy selection. For each variant in this table, we repeat the complete pruning and fine-tuning pipeline five times using seeds 0 through 4, starting from the same pretrained models.

Algorithm settings. Table 5 lists the partitioning, sampling, and allocation settings. We use $M = 512$ and $k = 4$, with at most 512 Gaussians in each temporary comparison group. Successive spatial passes change the grid origin using half-cell offsets. Graph construction uses two passes, and parallel block growth runs for at most 16 iterations. For each block, we use all training views with image width and height reduced to one quarter of their input dimensions. At this resolution, we

Table 5: Partitioning, response, and budget settings. Parameters without dedicated symbols in Section 3 are described by name.

Parameter	Setting
Maximum block size M	512
Neighbors per Gaussian per pass k	4
Comparison group size limit	512
Spatial passes	2
maximum growth iterations	16
Response views	All training views
Response image width and height	Input dimensions divided by 4
Pixel cap per block per view	256
Budget smoothing β	3
Budget pseudocount ϵ_a	1
Spherical-harmonic degree	3

generate candidate pixels using the projected-center offsets in Algorithm 2. After removing duplicate locations and those outside the image, We evaluate J_b at the selected pixels. We use $\beta = 3$ and $\epsilon_a = 1$ for budget assignment.

QUBO solver settings. Greedy selection enforces K_b directly and therefore does not use a count penalty. The SA backend uses Kaiwu's simulated annealing optimizer with an initial temperature of 100, a cutoff of 10^{-3}, and a cooling factor of 0.99. It uses 10 iterations per temperature, eight worker processes, and a solution limit of 100 per block. The SA random seed is the base seed 0 plus the block index. After decoding the lowest-energy returned state, the implementation repairs its count and applies up to n_b improving one-swap exchanges. The CIM backend additionally truncates its Ising coefficients to signed 14-bit precision and applies the same repair and refinement.

Fine-tuning settings. Fine-tuning uses Adam with numerical stabilizer 10^{-15} and one randomly selected training view per update. The initial learning rates for DC coefficients, higher-order coefficients, opacity, scale, and rotation are 0.0025, 0.000125, 0.05, 0.005, and 0.001, respectively. These rates decay by a factor of 0.95 every 400 refinement steps. Position updates use the inherited exponential schedule, with endpoints 1.6×10^{-4} and 1.6×10^{-6} multiplied by the scene scale. Refinement and final evaluation preserve input resolution when image width is at most 1600 pixels. Wider images are resized to approximately 1600 pixels in width while preserving aspect ratio. In the picture-free setting, neither pruning nor fine-tuning uses the original training images. We retain the training camera parameters and supervise fine-tuning with images rendered by the fixed, original pretrained model. Ground-truth test images are used only for evaluation.

Metrics methods. For Figure 6, we divide each test image into a 16×16 grid of 256 non-overlapping patches. We compute PSNR for each patch against its corresponding ground-truth region, including pure background patches in synthetic scenes. For each image, we calculate the variance and range (maximum minus minimum) of these patch PSNR values. We average each statistic over test views within each scene, then equally over scenes within each benchmark. Full-image PSNR is reported alongside these statistics to distinguish overall reconstruction quality from spatial quality variation.

For Figure 3, we normalize Gaussian volumes using a shared spatial reference for each scene. We construct an axis-aligned bounding box from the coordinate-wise minima and maxima of Gaussian centers across all nine pruned models. Its diagonal length is

$$D = \sqrt{(x_{\max} - x_{\min})^2 + (y_{\max} - y_{\min})^2 + (z_{\max} - z_{\min})^2}.$$

Each Gaussian's volume is divided by D^3, so all methods use the same normalization within a scene. For each method, we compute normalized volume quantiles separately for each scene. At each cumulative fraction, we average the corresponding quantiles equally across scenes within each benchmark. The resulting curves summarize scene-level volume distributions without pooling Gaussians across scenes.

B.2 Additional Experimental Results

Ablations with standard deviations. Table 6 supplements the mean results in Table 2 with standard deviations. For each variant, we repeat the complete pruning and fine-tuning pipeline five times using random seeds 0 through 4, starting from the same pretrained models. We first average scene-level metrics within each dataset, then report the mean and standard deviation of these dataset averages across runs.

Table 6: Component and solver ablations at 90% pruning. Each dataset cell reports the mean above its ± standard deviation across five runs of the complete pruning and fine-tuning pipeline.

Variant	Mip-NeRF 360			Deep Blending			Tanks and Temples		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Top-k	26.68 ±4.1e-3	0.7787 ±2.50e-5	0.2640 ±7.40e-5	29.09 ±1.31e-2	0.8944 ±5.85e-5	0.2649 ±7.42e-5	22.72 ±7.6e-3	0.8034 ±8.67e-5	0.2469 ±1.158e-4
+ Uniform block quotas	26.52 ±5.5e-3	0.7755 ±2.75e-5	0.2685 ±8.31e-5	28.98 ±1.42e-2	0.8930 ±5.37e-5	0.2668 ±8.32e-5	22.64 ±8.1e-3	0.8004 ±9.12e-5	0.2505 ±1.269e-4
+ Budget assignment	26.74 ±4.8e-3	0.7889 ±2.79e-5	0.2626 ±8.07e-5	29.15 ±1.35e-2	0.8958 ±5.87e-5	0.2638 ±7.92e-5	22.81 ±8.3e-3	0.8050 ±9.03e-5	0.2455 ±1.240e-4
+ Quadratic, greedy	27.03 ±5.2e-3	0.7928 ±2.82e-5	0.2552 ±8.27e-5	29.42 ±1.38e-2	0.9020 ±5.26e-5	0.2568 ±7.05e-5	23.15 ±7.5e-3	0.8135 ±8.77e-5	0.2357 ±1.141e-4
Quadratic, uniform quotas	26.79 ±4.7e-3	0.7811 ±2.65e-5	0.2602 ±8.01e-5	29.36 ±1.53e-2	0.8985 ±5.65e-5	0.2585 ±7.81e-5	23.07 ±8.6e-3	0.8093 ±8.45e-5	0.2424 ±1.374e-4
+ Quadratic, SA	27.07 ±4.4e-3	0.7932 ±2.56e-5	0.2547 ±7.90e-5	29.47 ±1.37e-2	0.9027 ±5.34e-5	0.2562 ±7.08e-5	23.21 ±7.3e-3	0.8160 ±8.41e-5	0.2322 ±1.393e-4
+ Quadratic, CIM	27.14 ±2.0e-3	0.7960 ±1.12e-4	0.2507 ±9.12e-5	29.54 ±1.24e-2	0.9041 ±1.22e-4	0.2532 ±1.20e-4	23.21 ±1.27e-2	0.8173 ±3.5e-5	0.2296 ±1.25e-4

Variant	NeRF Synthetic			Ref-NeRF			Macro		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Top-k	31.85 ±9.9e-3	0.9568 ±1.43e-5	0.0503 ±6.86e-5	29.55 ±8.7e-3	0.9407 ±4.31e-5	0.0972 ±5.88e-5	27.978	0.87480	0.18466
+ Uniform block quotas	31.72 ±8.5e-3	0.9557 ±1.48e-5	0.0520 ±7.12e-5	29.43 ±7.8e-3	0.9396 ±3.77e-5	0.0993 ±6.60e-5	27.858	0.87284	0.18742
+ Budget assignment	31.96 ±8.9e-3	0.9575 ±1.29e-5	0.0496 ±7.51e-5	29.64 ±9.0e-3	0.9415 ±3.97e-5	0.0961 ±6.79e-5	28.060	0.87774	0.18352
+ Quadratic, greedy	32.31 ±9.1e-3	0.9618 ±1.45e-5	0.0427 ±6.62e-5	29.96 ±9.4e-3	0.9449 ±3.73e-5	0.0906 ±7.08e-5	28.374	0.88300	0.17620
Quadratic, uniform quotas	32.14 ±9.1e-3	0.9596 ±1.40e-5	0.0452 ±6.97e-5	29.76 ±8.9e-3	0.9431 ±4.22e-5	0.0924 ±6.95e-5	28.224	0.87832	0.17974
+ Quadratic, SA	32.37 ±8.3e-3	0.9623 ±1.44e-5	0.0424 ±7.42e-5	30.01 ±8.7e-3	0.9453 ±4.13e-5	0.0897 ±6.30e-5	28.426	0.88390	0.17504
+ Quadratic, CIM	32.48 ±7.0e-3	0.9635 ±3.2e-5	0.0409 ±6.5e-5	30.07 ±3.6e-3	0.9456 ±3.3e-5	0.0887 ±1.27e-4	28.488	0.88530	0.17262

Pruning Time and Memory. Table 4 reports mean per-scene scoring time, QUBO solving time, and peak GPU memory. We average scenes equally within each dataset: nine for Mip-NeRF 360 and two each for Deep Blending and Tanks and Temples. We perform one pruning pass, removing 80% of the Gaussians from the original 3D-GS. These measurements use RTX 5090 GPUs on the same servers. For SplatQ, scoring includes partitioning and R_b calculation, taking 1005.0, 1237.2, and 3456.3 s on the three datasets, respectively. Selection includes budget allocation and solving QUBOs including necessary postprocessing. Greedy selection takes 101.1, 99.1, and 58.9 s, whereas SA takes 845.9, 748.4, and 448.9 s. Greedy therefore reduces solver time by approximately 7.6–8.4×, although R_b calculation remains the main computational cost. The reported peak GPU memory is the memory peak over scoring and QUBO solving stages. We average these per-scene peaks. SplatQ's peak GPU memory is similar to that of LightGS and C3DGS. These measurements show the substantial offline cost of SplatQ's interaction-aware selection.

Pruning latency should be considered within the complete compression pipeline, which also includes 20k fine-tuning iterations in our main setting. Our refinement sweeps show that longer fine-tuning generally improves reconstruction quality within the tested range. For a fixed pruning workload, longer fine-tuning reduces the relative contribution of pruning to total preparation time. Moreover, this cost is incurred offline and is not repeated during rendering. For applications that prioritize final reconstruction quality, the observed quality gains can therefore justify additional pruning time.

Effect of Pairwise Interactions without Fine-tuning. To isolate the contribution of pairwise interactions, we compare a diagonal-only objective with the full quadratic objective. We solve the diagonal-only problem exactly by retaining the K_b Gaussians with the largest D_i in each block. The full quadratic problem is solved using SA with the settings described above. Both variants use the same pretrained models, partitions, and block quotas. We perform one run per variant per scene, removing 80% of the Gaussians in a single round without fine-tuning. Table 7 reports dataset-average metrics from these runs. Including pairwise interactions improves PSNR by 0.34, 0.30, and 0.28 dB on Mip-NeRF 360, Deep Blending, and Tanks and Temples, respectively. Figure 7 provides corresponding visual comparisons.

Table 7: Effect of pairwise interactions after a single round of 80% pruning without fine-tuning. Both variants use identical pretrained models, partitions, and block quotas. Values are dataset averages; PSNR is measured in dB and LPIPS uses the VGG backbone. Better results are shown in bold.

	Mip-NeRF 360			Deep Blending			Tanks & Temples		
Objective	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Diagonal	21.57	0.6848	0.3330	26.51	0.8543	0.3018	18.71	0.7195	0.2871
Quadratic	**21.91**	**0.6890**	**0.3272**	**26.81**	**0.8572**	**0.2984**	**18.99**	**0.7215**	**0.2844**

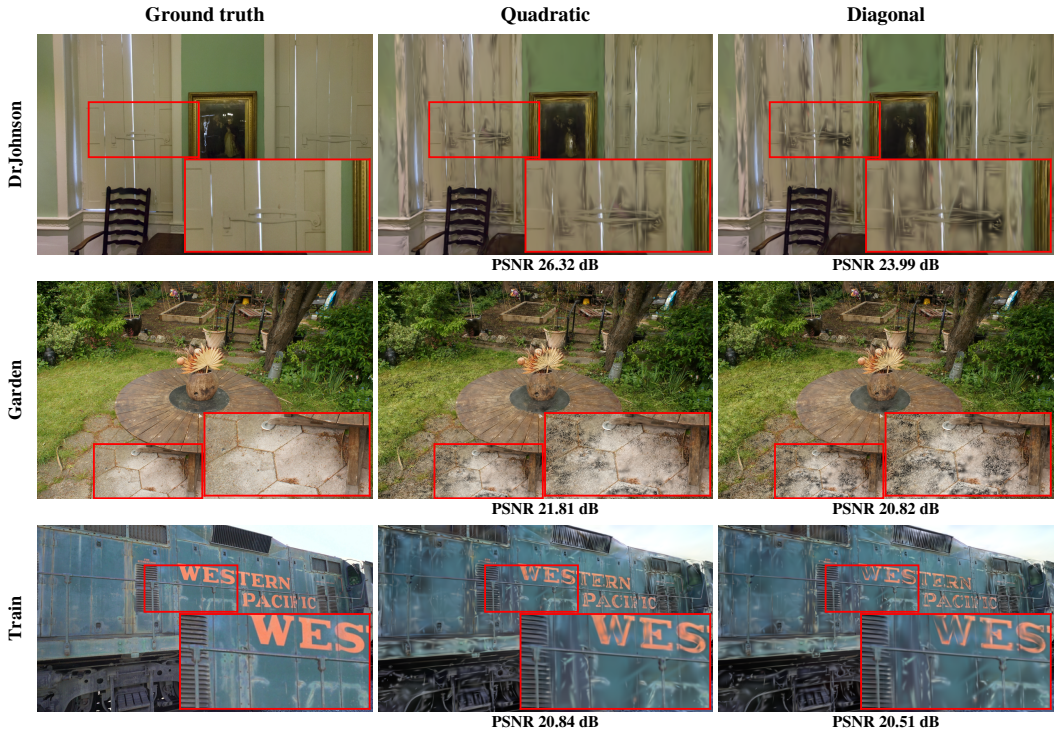

Figure 7: Visual comparison after a single round of 80% pruning without fine-tuning.

Sensitivity to Block Size and Smoothing. Table 8 evaluates block size M and smoothing strength β on Mip-NeRF 360 using SA. We vary one parameter at a time, fixing $\beta = 3$ for the block-size sweep and $M = 512$ for the smoothing sweep. Across $M \in \{256, 512, 1024, 2048\}$, PSNR varies by only 0.07 dB, while SSIM increases and LPIPS decreases as blocks grow. The largest tested size, $M = 2048$, achieves the best quality across all three metrics. Increasing M from 256 to 2048 also reduces scoring time from 1636.8 to 310.2 s and solver time from 940.4 to 536.3 s under the tested configuration. We use $M = 512$ by default to maintain compatibility with the CIM backend used in our experiments, which supports Ising matrices with fewer than 1,000 variables. Across $\beta \in \{1, 2, 3, 4, 5\}$, PSNR varies by only 0.05 dB. The default $\beta = 3$ achieves the highest PSNR and SSIM, and ties with $\beta = 2$ for the lowest LPIPS at the reported precision.

Table 8: Sensitivity to block size M and smoothing strength β on Mip-NeRF 360 using SA. We fix $\beta = 3$ when varying M and $M = 512$ when varying β. PSNR is measured in dB, and scoring and solver times are reported in seconds. Bold indicates the best quality result within each sweep, including ties at the reported precision. For the *beta* part, we report quality only and omit timing results.

	Block size M				Smoothing strength β				
Metric	256	512	1024	2048	1	2	3	4	5
PSNR↑	27.02	27.07	27.05	**27.09**	27.02	27.04	**27.07**	27.06	27.05
SSIM↑	0.7927	0.7932	0.7934	**0.7937**	0.7926	0.7929	**0.7932**	0.7929	0.7925
LPIPS↓	0.2556	0.2547	0.2539	**0.2528**	0.2549	**0.2547**	**0.2547**	0.2550	0.2553
Scoring time	1636.8	1005.0	541.6	310.2	–	–	–	–	–
Solver time	940.4	845.9	784.0	536.3	–	–	–	–	–

Sensitivity to Pruning and Fine-Tuning Figure 8 extends the pruning-ratio comparison at 10k fine-tuning steps in Figure 4 to additional nonzero fine-tuning budgets. Figure 9 shows the two-round slice of Figure 5. In the round study, the reported fine-tuning budget is per round: R rounds with T updates each require RT updates in total.

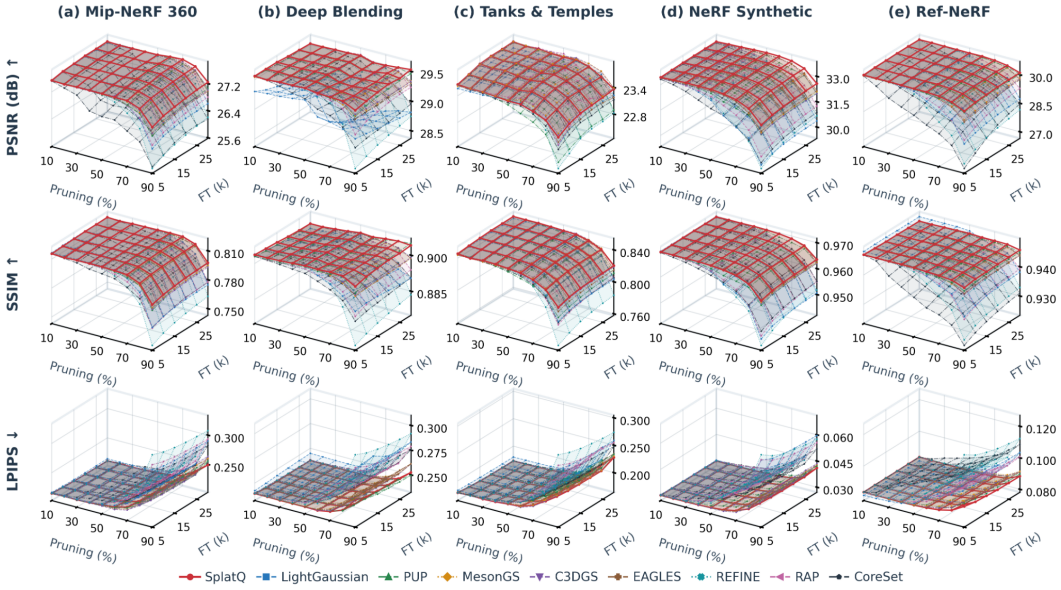

Figure 8: Reconstruction quality across pruning percentages and fine-tuning budgets. Columns correspond to Mip-NeRF 360, Deep Blending, Tanks and Temples, NeRF Synthetic, and Ref-NeRF; rows show PSNR, SSIM, and LPIPS-VGG. Axes indicate removal percentage, fine-tuning iterations in thousands, and the corresponding metric. SplatQ uses SA.

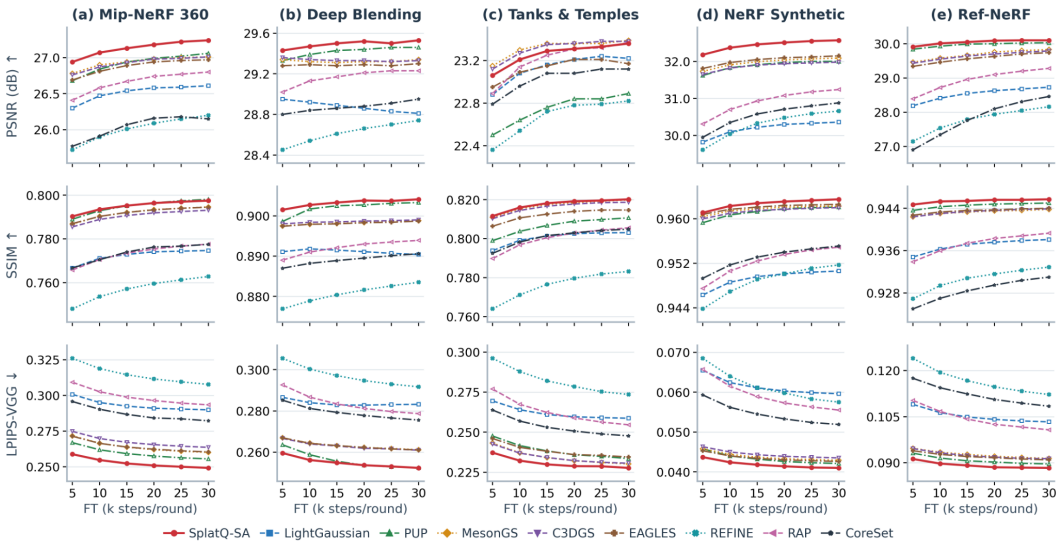

Figure 9: Effect of fine-tuning duration with two pruning rounds and 90% final removal. Columns correspond to the five datasets, and rows show PSNR, SSIM, and LPIPS-VGG. Fine-tuning steps on the horizontal axis are per round. The SplatQ curves use SA.