Prune Wisely, Reconstruct Sharply: Compact 3D Gaussian Splatting via Adaptive Pruning and Difference-of-Gaussian Primitives

Supplementary Material

S1. Additional Implementation Details

S1.1. Training Settings

We basically follow the backbone training settings of 3DGS [23]. Pruning is activated only after densification has finished. During training, we iteratively compute the average value of L_1 over the entire training set, and the Reconstruction-aware Pruning Scheduler uses this statistic to control both the timing and the ratio of pruning.

For the 3D-DoG primitives, we initialize the scaling factors $[f_x, f_y, f_z]$ to 0.5 and f_α to 0.1. Any 3D-DoG primitive with $f_\alpha < 0.01$ is degraded to a standard 3D Gaussian primitive.

S1.2. Baseline Configuration

In this paper, we avoid methods that rely on additional compact encodings, such as learned codebooks, to reduce the overall model size. This ensures a fair comparison that focuses purely on compressing the number of Gaussian primitives. We therefore select baselines that operate directly on 3D Gaussians.

For each baseline, we use the official implementation released by the respective authors. For MaskGaussian [27], we adopt the recommended value $\lambda_m = 0.0005$. For GaussianSpa, we use its 3DGS-based implementation [23] rather than MiniSplatting [9] to ensure a consistent densification strategy. We follow the recommended settings of the published Speedy-Splat implementation [13]. In addition, we directly report the evaluation results provided in the PUP-3DGS paper [14].

All baselines are trained with the same dataset splits, image resolution, number of iterations, and optimization settings as the original 3DGS.

S1.3. 3D-DoG Rasterization

Forward: Following subsection 3.3, we compute the scalings S_p and α_p for the pseudo-Gaussians. We then follow the procedure introduced in subsection 3.1 to obtain the 2D covariance matrices Σ_i' and $\Sigma_i^{p'}$. The weights for the primary Gaussians, α_i', are computed as in Equation 3, while the pseudo-Gaussian weights $\alpha_i^{p'}$ are obtained as

$$\alpha_i^{p'} = \alpha_p \exp\left(-\tfrac{1}{2}(x-\mu_i')^T(\Sigma_i^{p'})^{-1}(x-\mu_i')\right). \quad (15)$$

Next, we define the effective weight β_i as

$$\beta_i = \alpha_i' - \alpha_i^{p'}, \quad (16)$$

and accordingly modify the blending rule to

$$C = \sum_i c_i\,\beta_i \prod_{j=1}^{i-1}(1-\beta_j). \quad (17)$$

Backward: Recall that in subsection 3.3, the 3D-DoG model is defined as the subtraction of a pseudo-Gaussian from a primary Gaussian. To optimize the additional pseudo-Gaussian parameters, we modify the original rasterization implementation. The gradients of the opacity parameter α and the opacity factor f_α are given by

$$\frac{\partial L}{\partial \alpha} = \left(G - f_p G_p\right)\frac{\partial L}{\partial \beta}, \quad (18)$$

$$\frac{\partial L}{\partial f_\alpha} = -\alpha\, G_p\, \frac{\partial L}{\partial \beta}, \quad (19)$$

and the gradients for the scaling factors $[f_x, f_y, f_z]$ are

$$\frac{\partial L}{\partial f_x} = -0.5\,\frac{\partial L}{\partial G_p}\,G_p\,d_x^2, \quad (20)$$

$$\frac{\partial L}{\partial f_y} = -0.5\,\frac{\partial L}{\partial G_p}\,G_p\,d_y^2, \quad (21)$$

$$\frac{\partial L}{\partial f_z} = -0.5\,\frac{\partial L}{\partial G_p}\,G_p\,d_x d_y, \quad (22)$$

where $[d_x, d_y]$ denote the 2D Gaussian pixel offsets, and $\frac{\partial L}{\partial G_p}$ is a temporary variable defined as

$$\frac{\partial L}{\partial G_p} = -\alpha f_\alpha \frac{\partial L}{\partial \beta}. \quad (23)$$

S2. Additional Results

By adopting the Reconstruction-aware Pruning Scheduler, our method can compress 3DGS models to meet any predefined pruning target. Here, we additionally present results for pruning 50% and 80% of primitives. Evaluation results are reported in Table S1. The results show that our method achieves the best performance on the Deep Blending dataset [15] and that, across different pruning ratios, the metrics of the compressed models can match those of the 3DGS backbone. Moreover, our method can retain the original performance on Mip-NeRF 360 [4] and Tanks and Temples [24] when pruning 50% of primitives. Finally, note that our 20% model can outperform GaussianSpa [44] while using less memory.

Table S1. Quantitative evaluation of our method using different pruning ratios, computed over three datasets: Mip-NeRF 360, Deep Blending, and Tanks and Temples. ↑ indicates that larger values are better, while ↓ indicates the opposite.

Method	Mip-NeRF 360 [4]					Deep Blending [15]					Tanks and Temples [24]				
	Size ↓	PSNR↑	SSIM↑	LPIPS↓	Time↓	Size ↓	PSNR↑	SSIM↑	LPIPS↓	Time↓	Size ↓	PSNR↑	SSIM↑	LPIPS↓	Time↓
3DGS [23]	645.2	27.47	0.826	0.201	17m1s	592.7	29.75	0.904	0.244	13m51s	381.0	23.77	0.847	0.177	17m50s
Ours-50%	328.3	27.51	0.815	0.217	15m42s	300.5	29.89	0.905	0.242	12m3s	192.2	23.81	0.851	0.171	11m43s
Ours-20%	131.5	27.47	0.809	0.241	14m10s	120.2	29.87	0.905	0.241	10m41s	76.8	23.71	0.843	0.195	8m42s
Ours-10%	65.3	27.16	0.789	0.285	13m48s	59.9	29.87	0.904	0.254	10m19s	38.4	23.79	0.823	0.229	8m9s