# RaPTGS: Render-Agnostic Post-Training Compression of 3D Gaussian Splatting

Asif Jawad[1] and K. M. Azwad Hossain[1]

Samsung R&D Institute Bangladesh, Dhaka 1205, Bangladesh

**Abstract.** 3D Gaussian Splatting (3DGS) has emerged as a powerful technique for real-time, high-fidelity novel view synthesis. Despite these advantages, its multi-million point representation leads to large storage footprints that hinder asset distribution and on-device deployment. Existing compression approaches typically rely on access to training images, camera poses, or the rendering pipeline for retraining or iterative fine-tuning. However, in realistic post-training and archival scenarios, often only the optimized model parameters are available. To address this limitation, we propose a render-agnostic, post-training compression pipeline for 3DGS operating strictly under a model-only constraint. We first introduce a camera-independent, multi-criteria importance score to prune redundant Gaussians based on a combination of geometric, spatial, and appearance-related cues. Following this pruning, a lightweight, training-free refinement step conservatively restores local coverage. To further reduce the storage footprint, we apply degree-wise vector quantization to the spherical harmonic coefficients and compress the remaining attributes via entropy coding. The compression performance is significantly enhanced by a custom spatial reordering to maximize local data redundancy. Experiments on standard benchmarks show that our approach achieves an average of 30× compression rate while maintaining competitive visual quality against existing methods.

**Keywords:** Compression · Gaussian Splatting · Novel View Synthesis

## 1 Introduction

Novel view synthesis (NVS) generates photorealistic images from unseen viewpoints. Neural Radiance Fields (NeRF) [19] achieve this using continuous, network parameterized radiance fields and volumetric ray integration, achieving high visual quality but incurring significant computational costs.

To address these limitations, 3D Gaussian Splatting (3DGS) [13] has emerged as an efficient, high-fidelity alternative. By jointly optimizing explicit Gaussian parameters (position, scale, rotation, opacity) and view-dependent appearance (spherical harmonics), 3DGS achieves interactive rendering via fast rasterization. This makes it ideal for latency-sensitive applications such as immersive content, digital twins, and interactive scene exploration.

Despite its speed, 3DGS is storage-intensive: realistic scenes contain millions of multi-attribute Gaussians, resulting in massive model sizes that impede

distribution, caching, and on-device use [20, 32]. Consequently, recent compression works focus on pruning primitive counts or learning compact representations [1, 12, 30]. However, these approaches typically require retraining or fine-tuning, demanding access to the original training data and substantial compute. Furthermore, many pruning methods rely on rendering or camera statistics, tightly coupling compression to specific view configurations and repeated rendering passes.

However, *post-training compression in the wild* often faces stricter, more realistic constraints, where only the trained 3DGS model (Gaussian parameters) is available, without access to training images, camera poses, or SfM/COLMAP outputs. In asset distribution and archival scenarios, provenance metadata is frequently missing, proprietary, or impractical to ship alongside the model. Under this *model-only* assumption, methods that rely on rendering supervision, training-time views, or iterative refinement become difficult to apply, even if they are highly effective when full training data is available [8, 20].

In this work, we propose *RaPTGS*, a render-agnostic, post-training compression pipeline for 3DGS that operates entirely in the *model-only* setting. Our approach serves as a lightweight post-process over the Gaussian parameters, requiring no iterative fine-tuning or view-dependent statistics. In summary, our main contributions are:

- We propose a novel, rendering-independent metric to evaluate the significance of individual Gaussians without requiring camera poses or rendering loops. By analyzing the intrinsic geometric and visual properties of the model, we extract an importance score based on a Gaussian's opacity salience, shape anisotropy, and scale-aware appearance energy, allowing for accurate and efficient pruning.
- To mitigate fidelity loss from pruning without reverting to iterative optimization, we propose *Voxel Mass Compensation (VMC)*. This grid-based technique conservatively scales the remaining primitives to restore local volume coverage efficiently.
- By coupling our pruning and refinement steps with degree-wise Spherical Harmonics (SH) quantization and spatially-reordered entropy coding, we achieve an average compression rate of 30×. Extensive experiments on standard benchmarks demonstrate that RaPTGS maintains highly competitive visual quality, outperforming recent state-of-the-art baselines while offering significantly faster compression times.

## 2 Related Work

The introduction of 3DGS [13] has garnered significant interest in the field of 3D reconstruction, leading to extensive research on optimization, rendering quality, dynamic scenes, and large-scale reconstruction [5,7,17,34]. To reduce the storage and memory footprint of 3DGS models while preserving visual quality, numerous compression methods have been proposed [1]. At the level of technique, two directions dominate: redundancy reduction (pruning) and parameter compactness

(quantization/encoding). These methods are applied during training, as post-processing, or through feed-forward networks, with the latter two being most relevant to the model-only setting considered in this work.

**Redundancy Reduction (Pruning):** Pruning methods aim to identify and remove redundant Gaussians that contribute little to the overall scene representation. These approaches can be divided into two categories: those that include pruning as part of the training process [20, 23, 37], and those that apply pruning as a post-processing step after training [8, 25, 27, 32]. LightGaussian [6] introduces a Global Significance Score that evaluates the contribution of each Gaussian to the overall scene representation by considering its visibility and physical properties across all training views. Many subsequent works [11, 27] use this score as a basis for pruning. FlexGaussian [27], for example, adopts it for attribute-discriminative pruning and still loads camera parameters during compression. This reliance on rendering-based visibility and contribution metrics makes such scores inapplicable in the model-only setting we consider. A simpler opacity-only threshold avoids per-view computation but tends to discard perceptually important fine detail, noticeably reducing rendering quality [6]. The post-training pruning pipelines in PUP 3D-GS [8] and MesonGS [32] likewise rely on rendering-based importance scores, the former computing a second-order sensitivity of the reconstruction error on the training views, and the latter deriving its pruning importance, in both its fine-tuning-free and fine-tuned variants, from a forward-rendering pass over the training views. More recently, POTR [25] employs a modified 3DGS rasterizer to compute, in parallel, each splat's contribution and the change in rendering quality caused by its removal. This criterion is again obtained through rasterization and therefore presupposes the rendering pipeline and camera views that our setting excludes.

**Parameter Compactness (Quantization/Encoding):** Quantization approaches focus on reducing the precision of Gaussian parameters, such as position, scale, rotation, and appearance attributes, to save storage space. This includes reduced-precision representations, such as codebook and half-float storage [23], K-means-based vector quantization with learned codebooks [21,22], and residual vector quantization with learned codebooks [15]. More sophisticated encoding schemes further exploit parameter distributions for improved compression [3, 4, 16]. A complementary strategy is mixed-precision quantization, which assigns different bit-widths to different attribute channels. FlexGaussian [27] combines attribute-discriminative pruning with channel-wise mixed-precision quantization for training-free compression. SizeGS [31] formulates the Gaussian reserve ratio and per-attribute bit-width allocation as a mixed-integer optimization problem, enabling compression to a target model size. Several recent methods additionally employ entropy coding to compress 3DGS parameters [3,4,11,16,28]. In particular, CAT-3DGS [35] performs rate-distortion-optimized coding on top of Scaffold-GS [18] by projecting primitives onto multi-scale triplanes aligned with their principal axes. These triplanes serve as a context-adaptive hyperprior for spatial and channel-wise autoregressive entropy modeling. Another line of work imposes a structured 2D layout so that conventional image or video codecs

can be applied. Self-Organizing Gaussians [20] arrange parameters into a smooth grid through parallel assignment, whereas LGSCV [33] replaces this expensive sorting process with a two-stage Morton scan, a lightweight MiniPLAS, and PCA-reduced spherical harmonics. The resulting blockwise 2D maps are then compressed using standard video codecs. A further line applies a learned entropy codec in a feed-forward manner: a single network, trained once, encodes any input model in one inference pass, amortizing compression across scenes and trading some rate-distortion performance for a large speedup over per-scene optimization. FCGS [3] is the principal such codec, pairing adaptive attribute processing with inter- and intra-Gaussian context models for entropy coding. It is optimization-free but depends on a pretrained network and, owing to the stored intermediate representations and context models, can be memory-prohibitive for large scenes [27].

Across these groups, most methods presuppose rendering supervision or the original training inputs, and many of the remaining approaches rely on a learned compression model. Instead, we consider the strict model-only regime, compressing a trained 3DGS scene solely from its parameters using standard off-the-shelf coding tools. Our approach requires neither training images, camera poses, a rendering pipeline, nor a learned codec.

## 3 Method

### 3.1 Problem Formulation

Let the trained 3DGS scene be represented as a set of $N$ Gaussians

$$\mathcal{G} = \{g_i\}_{i=1}^{N} \tag{1}$$

Each Gaussian $g_i$ consists of a 3D position $\mathbf{x}_i \in \mathbb{R}^3$, an opacity $\alpha_i \in [0, 1]$, axis-wise scales $\mathbf{s}_i = (s_{i,x}, s_{i,y}, s_{i,z}) \in \mathbb{R}^3_+$, rotation $\mathbf{r}_i \in \mathbb{S}^3$ and spherical-harmonic (SH) coefficients $c_i = \{c_i^{(\ell)}\}_{\ell=0}^{3}$ describing view-dependent appearance. Based on two hyperparameters, a pruning ratio $p \in [0, 1)$ and a bit budget $B$, our objective is to construct a compressed 3DGS scene $\mathcal{G}'$ such that $|\mathcal{G}'| = \lfloor (1-p)N \rfloor$, while the AC SH coefficients ($\ell \in \{1, 2, 3\}$) of each Gaussian in $\mathcal{G}'$ are quantized with a discrete representation controlled by $B$.

### 3.2 Overview

An overview of our pipeline is shown in Fig. 1. Starting from a trained scene $\mathcal{G}$, we produce the compressed scene $\mathcal{G}'$ through four stages. *(1) Render-Agnostic Pruning* ranks Gaussians by view-independent importance proxies and keeps the top $|\mathcal{G}'|$. *(2) Post-Pruning Refinement* applies Voxel Mass Compensation (VMC) to rescale retained Gaussians and recover lost local coverage. *(3) Degree-wise SH Quantization* vector-quantizes the AC SH coefficients band by band controlled by the bit budget $B$. *(4) Attribute Compression* reorders the Gaussians and entropy-codes the uniformly quantized attributes. Each stage is detailed in the following subsections.

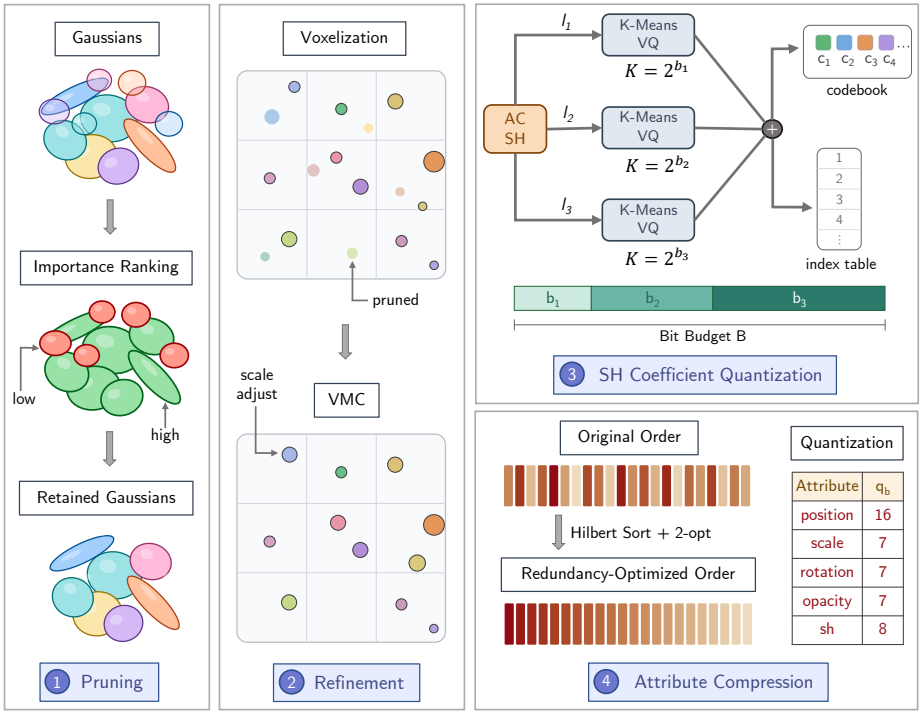


**Fig. 1:** Overview of our proposed method. (1) Pruning: Gaussians are scored by view-independent importance proxies and only the top-ranked ones are retained. (2) Refinement: the scene is voxelized and surviving Gaussians' scales are adjusted (VMC) to compensate for lost spatial mass. (3) SH Coefficient Quantization: AC SH coefficients are split by degree $(\ell_1, \ell_2, \ell_3)$ and vector-quantized via K-means codebooks controlled by a per gaussian bit budget $B$. (4) Attribute Compression: Gaussians are reordered (Hilbert sort + 2-opt) for stream locality, then uniformly quantized and entropy-coded.

### 3.3 Render Agnostic Pruning

**Gaussian Importance Estimation.** Operating strictly within the *model-only* compression paradigm means our method relies exclusively on the optimized 3D Gaussian parameters. This intrinsic restriction precludes importance measures derived from reconstruction error, including view-aggregated hit statistics, transmittance-aware visibility, or second-order sensitivity scores. Without these view-dependent signals, a standard heuristic is to adopt opacity as a proxy for importance [20] by pruning low-opacity Gaussians. Yet, as highlighted by prior work [6], opacity-centric pruning frequently sacrifices fine-grained structures like thin geometry, foliage, or text, resulting in a substantial loss of quality.

Motivated by these limitations, we adopt a multi-criteria approach to assess Gaussian significance. Specifically, a Gaussian is considered important if it excels in at least one of three complementary aspects: (i) visibility potential, (ii) geometric specificity, or (iii) scale-aware appearance complexity.

We first quantify the spatial footprint of each primitive, a prerequisite for assessing scale-aware appearance. Let the geometric extent of a Gaussian be the sum of its activated scales: $e_i = s_{i,x} + s_{i,y} + s_{i,z}$. Unlike volume ($s_{i,x}s_{i,y}s_{i,z}$), which erroneously vanishes for highly anisotropic splats, this linear sum robustly captures maximum spatial reach. We then evaluate Gaussian importance via three complementary proxies.

*(1) Opacity Salience (OS).* Opacity is a direct, albeit rudimentary, proxy for a primitive's potential influence under alpha compositing:

$$\mathrm{OS}_i \;=\; \alpha_i. \tag{2}$$

*(2) Shape Anisotropy Strength (SAS).* Anisotropy serves as a proxy for geometric specificity, preserving structural elements like thin splats or aligned surfaces. We measure this via the ratio of the largest to smallest scale components. To mitigate sensitivity to extreme outliers and suppress highly anisotropic but invisible primitives, we apply a logarithm and weight by opacity:

$$\mathrm{SAS}_i \;=\; \alpha_i \,\log\!\left(\frac{\max(s_{i,x}, s_{i,y}, s_{i,z})}{\min(s_{i,x}, s_{i,y}, s_{i,z}) + \varepsilon}\right) \tag{3}$$

where $\varepsilon > 0$ is a small constant for numerical stability.

*(3) Scale-Aware Appearance Energy (SAE).* To quantify view-dependent complexity, we weight higher-order Spherical Harmonic (SH) energy ($\ell \geq 1$) by opacity and spatial extent. Because strictly linear scaling with $e_i$ disproportionately penalizes small primitives that encode critical high-frequency details, we establish a lower bound $e_{\min}$. This threshold is set such that the smallest primitives ($e_i < e_{\min}$) collectively account for 15% of the scene's total spatial footprint. Using the clamped extent $\tilde{e}_i = \max(e_i, e_{\min})$, our appearance proxy is:

$$\mathrm{SAE}_i \;=\; \alpha_i\, \tilde{e}_i \,\log\!\left(1 + \sum_{\ell=1}^{3} \left\|\mathbf{c}_i^{(\ell)}\right\|_2^2\right) \tag{4}$$

This clamping naturally preserves fine-grained details by avoiding premature culling.

*Combined Importance Score.* Because the three proxies possess different units and dynamic ranges, we normalize them via percentile ranking across all $N$ Gaussians, denoted by $\pi(\cdot) \in [0, 1]$. The final importance score is defined as:

$$\mathcal{I}_i \;=\; \max\Big(\pi(\mathrm{OS}_i),\; \pi(\mathrm{SAS}_i),\; \pi(\mathrm{SAE}_i)\Big) \tag{5}$$

The maximum pooling operator acts as a logical OR for primitive preservation. Gaussians salient in visibility, geometric structure, or view-dependent appearance are retained even if they remain inconspicuous under alternative criteria. Conversely, this formulation restricts pruning to primitives that are universally uninformative across all three dimensions, safely culling them without degrading fine-grained details.

**Global Top-$K$ Importance Pruning.** Given per-Gaussian importance scores $\mathcal{I}_i$, we retain the top-$K$ Gaussians with highest importance, where $K = |\mathcal{G}'|$. The pruned set is defined as

$$\mathcal{G}' = \mathrm{TopK}(\mathcal{G}, \{\mathcal{I}_i\}, K)\,. \tag{6}$$

### 3.4 Post-Pruning Refinement

Pruning reduces Gaussian density and may introduce local coverage gaps, particularly in regions where many splats receive low importance scores. To mitigate this effect without retraining, we apply a lightweight post-processing step termed *Voxel Mass Compensation (VMC)*.

**Voxelization.** Given voxel size $h$, we partition the scene into a regular grid with $V(h)$ voxels and assign each Gaussian to a voxel according to its 3D center. We select $h$ such that the number of voxels approximately matches the number of retained Gaussians, $V^* = |\mathcal{G}'|$, promoting a locally balanced granularity for compensation. Concretely, we determine

$$h^* = \operatorname*{argmin}_h \; |V(h) - V^*|, \tag{7}$$

which we solve efficiently via binary search.

**Voxel Mass Compensation (VMC).** To quantify the spatial occupancy within a given region, we first define the Spatial Support Mass (SSM) of an individual Gaussian as its opacity-weighted geometric extent: $\mathrm{SSM}_i = \alpha_i e_i$. For a given voxel $v$, we then define the total voxel mass $\mathcal{M}_v$ as,

$$\mathcal{M}_v \;=\; \sum_{g_i \in \mathcal{V}_v} \mathrm{SSM}_i \;=\; \sum_{g_i \in \mathcal{V}_v} \alpha_i (s_{i,x} + s_{i,y} + s_{i,z}), \tag{8}$$

where $\mathcal{V}_v$ denotes the set of Gaussians whose centers lie within voxel $v$.

To locally adjust Gaussian scales after pruning, we compute a compensation factor for each non-empty voxel based on the mass lost:

$$\rho_v = \frac{\mathcal{M}_v^{\mathrm{pre}}}{\mathcal{M}_v^{\mathrm{post}}}, \tag{9}$$

where $\mathcal{M}_v^{\mathrm{pre}}$ and $\mathcal{M}_v^{\mathrm{post}}$ are the voxel mass before and after pruning, respectively. We apply $\rho_v$ uniformly to the scale components of all retained Gaussians in voxel $v$, so voxels that lost more mass receive stronger local inflation.

To prevent excessive growth in sparsely populated voxels (e.g., when $\mathcal{M}_v^{\mathrm{post}}$ is small), we restrict the final scaled dimensions of the retained Gaussians. Specifically, we independently compute the maximum scale along each axis $(x, y, z)$ across all Gaussians present in voxel $v$ prior to pruning. After scaling the retained Gaussians by $\rho_v$, we clamp each axis of the new scale vectors to these pre-computed voxel-wise maximums. This conservative design is crucial in a post-training pipeline without access to training views or iterative refinement, avoiding oversmoothing or floating artifacts while restoring coverage locally.

### 3.5 Degree-wise Quantization of SH Coefficients

AC SH coefficients at different degrees exhibit distinct statistical characteristics. To accommodate these differences, we perform degree-wise quantization, enabling independent rate allocation across SH bands. We group the AC SH coefficients by degree and vectorize them as

$$L_1 \equiv c_i^{(1)} \in \mathbb{R}^9, \quad L_2 \equiv c_i^{(2)} \in \mathbb{R}^{15}, \quad L_3 \equiv c_i^{(3)} \in \mathbb{R}^{21},$$

where each dimensionality follows $3(2\ell{+}1)$, corresponding to the $(2\ell{+}1)$ spherical harmonic basis functions at degree $\ell \in \{1, 2, 3\}$ across three color channels.

For each group $L_i$, we learn a vector quantization codebook of size $K_i = 2^{b_i}$ via K-means clustering and assign each Gaussian to its nearest centroid using an index mapping table. We impose a total bit budget $B$ per Gaussian for the three AC groups. Under this per-Gaussian budget, we compute a single global bit allocation $(b_1, b_2, b_3)$ that is shared across all Gaussians. The allocation is obtained by minimizing a dimension-weighted mean squared reconstruction error aggregated over the sampled Gaussians used for codebook training:

$$\min_{b_1+b_2+b_3=B} \left\{ \sum_{i=1}^{3} d_i \, \mathrm{MSE}_{L_i}(b_i) \right\}, \quad d_i = 3\,(2\ell_i + 1). \tag{10}$$

The allocation is constrained to $b_i \in \{8, \ldots, 14\}$ to ensure practical codebook sizes and manageable search complexity. We estimate rate-distortion behavior using random subsampling of Gaussians during codebook training and evaluation. Codebook search across candidate bit-widths is further accelerated using a centroid-doubling warm start with limited Lloyd iterations.

### 3.6 Attribute Compression

We store each Gaussian attribute separately on disk, while enforcing a shared ordering across all of them enabling an unambiguous reconstruction without additional per-Gaussian metadata. We quantize the Gaussian attributes and compress them using off-the-shelf entropy coders. To improve entropy coding efficiency, we apply a spatial reordering to the Gaussians before quantization.

**Quantization.** All attributes are linearly normalized to $[0, 1]$ and uniformly quantized by rounding to the nearest value among $2^q$ discrete levels. We use $q_{\text{position}} = 16$, $q_{\text{scale}} = q_{\text{rotation}} = q_{\text{opacity}} = 7$, and $q_{\text{sh}} = 8$ (DC uniformly quantized; AC codebook centroids uniformly quantized to 8 bits). The quantized attributes are then compressed with lossless JPEG XL, following the approach of [20], while the index mapping tables are stored as bit-packed binary arrays in separate ZIP-compressed files. Note that our method is not restricted to this particular codec and can be readily combined with other existing compression techniques.

**Reordering for Efficient Entropy Coding.** Entropy coding is more efficient when consecutive elements exhibit strong value similarity. Since positions are quantized at higher precision (16 bits), we reorder Gaussians using only positional information to increase locality in the *position stream*. We seek a permutation $\pi$ minimizing the $\ell_1$ path length

$$L(\pi) = \sum_{i=1}^{|\mathcal{G}'|-1} \left\| x_{\pi_i} - x_{\pi_{i+1}} \right\|_1, \tag{11}$$

The choice of $\ell_1$ as the distance metric is motivated by its separability across dimensions, computational efficiency and its consistency with Laplacian-like residual distributions commonly exploited by entropy coders.

Direct minimization of $L(\pi)$ is infeasible at the scale of millions of Gaussians, so we approximate it with a two-stage heuristic. First, we globally sort Gaussians using Hilbert keys [10] computed via the *Axes-to-Transpose* [26] method, which maps spatially nearby Gaussians to adjacent indices in a 1D sequence. Second, we refine the ordering using a *windowed 2-opt* heuristic, which considers swapping edges $(i, i{+}1)$ and $(j, j{+}1)$ for index pairs $(i, j)$ satisfying $i < j < i + W$.

For efficiency, we propose a *parallel windowed 2-opt* strategy by observing that indices satisfying $i \equiv r \pmod{W+1}$ produce non-overlapping swap regions and can be processed independently. For each processed $i$, we select the $j$ that produces the largest reduction in path length and apply the swap only if it improves the ordering. The procedure is repeated until the improvement rate of $L(\pi)$ between iterations falls below $\tau$. In our implementation, we set the window size to $W = |\mathcal{G}'|^{1/3}$ to balance computational complexity with sufficient neighborhood coverage for effective ordering refinement. We set the stopping threshold to $\tau = 0.1\%$ to ensure scale-invariant convergence while avoiding marginal improvements in entropy coding efficiency.

## 4 Experiments

We evaluate on three standard 3DGS benchmarks: Mip-NeRF 360 [2], Tanks & Temples [14], and Deep Blending [9], following the 3DGS [13] protocols and splits. We report PSNR, SSIM [29], LPIPS [36], storage size, and encoding time.

**Baselines.** We compare against the uncompressed 3DGS [13] and a representative set of recent compression methods: the fine-tuning-free FlexGaussian [27] and MesonGS [32], the fine-tuning-based MesonGS-FT [32] and PUP 3D-GS [8], the video-codec-based LGSCV [33], and the feed-forward learned codec FCGS [3].

**Hardware and Implementation.** Experiments ran on an Intel Core Ultra 9 285K (24 cores) with 32 GB RAM and an NVIDIA RTX 5080 (16 GB VRAM). The pipeline is implemented in PyTorch [24], with CUDA kernels accelerating the windowed 2-opt and degree-wise SH quantization stages.

**Table 1: Quantitative comparison on Mip-NeRF 360, Tanks & Temples, and Deep Blending**. All sizes are in MB. The best results in each metric are **bolded**, and the second-best results are underlined.

| Method | Mip-NeRF 360 | | | | Tanks & Temples | | | | Deep Blending | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PSNR↑ | SSIM↑ | LPIPS↓ | Size↓ | PSNR↑ | SSIM↑ | LPIPS↓ | Size↓ | PSNR↑ | SSIM↑ | LPIPS↓ | Size↓ |
| 3D-GS [13] | 27.09 | 0.805 | 0.226 | 795.26 | 23.38 | 0.841 | 0.182 | 421.90 | 29.52 | 0.902 | 0.243 | 703.77 |
| FlexGaussian [27] | 26.31 | 0.775 | 0.254 | 42.40 | 22.52 | 0.807 | 0.215 | 16.31 | 28.65 | 0.887 | 0.265 | 25.60 |
| MesonGS [32] | 26.30 | 0.785 | 0.250 | 28.29 | 23.02 | 0.828 | 0.198 | 16.51 | 29.22 | 0.898 | 0.250 | 27.62 |
| MesonGS-FT [32] | **26.98** | **0.799** | **0.240** | 28.34 | **23.28** | **0.834** | **0.194** | 16.56 | **29.45** | **0.902** | **0.247** | 27.70 |
| PUP 3D-GS [8] | 26.66 | 0.788 | 0.267 | 79.52 | 22.46 | 0.798 | 0.248 | 42.19 | 29.39 | 0.902 | 0.255 | 70.37 |
| Ours | 26.41 | 0.782 | 0.252 | **25.95** | 22.97 | 0.823 | 0.203 | **13.64** | 29.36 | 0.899 | 0.248 | **24.51** |

**Table 2: Encoding time.** Our method is the fastest. All methods are evaluated on the same hardware to ensure a fair comparison.

| | Mip-NeRF 360 | | | Tanks & Temples | | | Deep Blending | | |
|---|---|---|---|---|---|---|---|---|---|
| Method | FlexGaussian | MesonGS | Ours | FlexGaussian | MesonGS | Ours | FlexGaussian | MesonGS | Ours |
| Time(s) | 42.44 | 199.97 | **7.60** | 30.25 | 227.43 | **5.78** | 44.20 | 159.23 | **7.70** |

### 4.1 Experimental Results

**Quantitative Results.** In Table 1, we present the quantitative performance of *RaPTGS*. To match the results of other works, we adopt scene-specific hyperparameter settings. The pruning ratio $p$ is varied between 0.35 and 0.45, while the bit budget $B$ ranges from 28 to 36.

Our approach achieves the highest level of compactness across all evaluated benchmarks, providing a 30× reduction in memory footprint compared to the uncompressed 3DGS [13]. Quantitative results on Mip-NeRF 360, Tanks & Temples, and Deep Blending demonstrate that our method maintains high visual fidelity while consistently requiring the least storage. Notably, on Mip-NeRF 360 and Deep Blending, our model achieves higher PSNR than both FlexGaussian [27] and MesonGS [32] despite its smaller size. While MesonGS-FT [32] and PUP 3D-GS [8] yield the highest reconstruction metrics through their respective iterative fine-tuning and multi-round prune-refine pipelines, our method maintains comparable quality using only a one-pass analytical scale adjustment. Because our refinement is render-free and requires no gradient updates, it avoids costly post-compression optimization and yields substantially faster encoding times than all competing baselines (Table 2).

**Comparison with FCGS.** FCGS [3] is among the closest learned baselines to our model-only setting: it compresses a pretrained 3DGS in a single feed-forward pass without per-scene fine-tuning or access to training images. However, its autoencoder-based architecture is memory-intensive [27]. On our hardware (RTX 5080, 16 GB), it ran out of memory on all 13 fully trained (30k-iteration) scenes

**Table 3: Quantitative comparison between our method and FCGS.**

| Scene | Method | PSNR↑ | SSIM↑ | LPIPS↓ | Size(MB)↓ | Time(s)↓ | Memory(GB)↓ |
|---|---|---|---|---|---|---|---|
| bonsai-7k | 3D-GS | 29.74 | 0.924 | 0.215 | 273.67 | - | - |
| | FCGS $1e^{-4}$ | 29.69 | 0.923 | 0.216 | 17.78 | 7.00 | 13.43 |
| | Ours | 29.12 | 0.915 | 0.227 | 8.61 | 4.32 | 2.83 |
| kitchen-7k | 3D-GS | 28.87 | 0.905 | 0.151 | 398.44 | - | - |
| | FCGS $1e^{-4}$ | 28.77 | 0.904 | 0.153 | 25.23 | 9.51 | 13.53 |
| | Ours | 28.43 | 0.898 | 0.159 | 12.92 | 5.08 | 2.95 |
| train-7k | 3D-GS | 18.97 | 0.690 | 0.350 | 132.27 | - | - |
| | FCGS $1e^{-4}$ | 18.96 | 0.688 | 0.351 | 8.30 | 3.75 | 10.58 |
| | Ours | 18.84 | 0.682 | 0.358 | 4.21 | 3.15 | 2.69 |

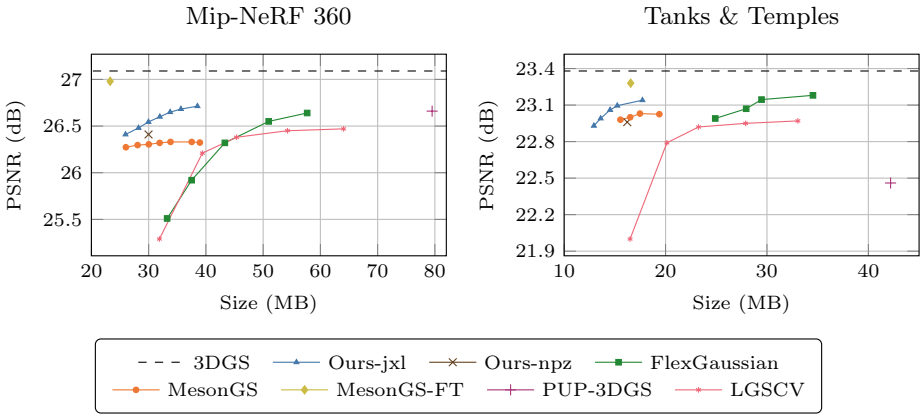


**Fig. 2: Rate-distortion curves.** Our method achieves favorable rate-distortion trade-offs across different datasets and size budgets.

from Mip-NeRF 360, Tanks & Temples, and Deep Blending. Re-evaluating on lighter 7k-iteration checkpoints, it still failed on 10 of 13 scenes; Table 3 reports the 3 in which it succeeded. On these reduced scenes, FCGS achieves marginally higher fidelity, as expected from its learned neural compressor. In contrast, our method yields roughly 2× smaller models, is nearly 4.5× more memory-efficient, and compresses in comparable or less time. Critically, this makes our method substantially more practical when datacenter-class GPUs are unavailable.

**Rate-Distortion Analysis.** Figure 2 reports the rate-distortion behavior of our method. To probe the high-bitrate regime, we vary the hyperparameters $p$ and $B$ and raise the maximum bit allocation $b_i$ from 14 to 16 (Eq. (10)). Across both datasets and all size budgets, our method attains compression-quality trade-offs competitive with the baselines. Specifically, within our operating range, our method achieves the best rate-distortion trade-off among all methods except MesonGS-FT. Since MesonGS-FT relies on fine-tuning, our approach offers the best trade-off among fine-tuning-free methods.


**Fig. 3:** Qualitative comparison of our compression method with ground truth, 3DGS and existing works on different scenes.

**Choice of Compression Format.** To assess the impact of the compression format, we additionally report our method under standard archive-based entropy coding (zip/npz). On Mip-NeRF 360, the compressed size is 29.98 MB (w/ reordering, Ours-npz in Fig. 2) vs. 34.91 MB (w/o reordering), demonstrating that the proposed reordering consistently improves compressibility even under archive-based coding. However, Fig. 2 also shows that JPEG-XL (Ours-jxl) consistently provides a better rate-distortion trade-off than npz, further motivating our choice of JPEG-XL over archive-based codecs.

**Qualitative Results.** Figure 3 shows a visual comparison of our method against the ground truth, uncompressed 3DGS and existing compression methods on multiple scenes side by side. Our approach effectively preserves fine details and overall structure closely matching the original 3DGS quality. In contrast, both FlexGaussian and MesonGS exhibit noticeable deviation and loss of detail, particularly in complex regions and far-away areas.

**Pruning Strategy.** We compare our strategy with opacity-based pruning [6,20] and MesonGS [32] (Fig. 4). Across all pruning ratios, Ours consistently preserves higher PSNR and SSIM than the opacity baseline. This advantage is most pronounced at moderate-to-aggressive pruning levels, where opacity-only ranking degrades rapidly. This suggests our scoring criteria better retains structurally and appearance-critical Gaussians.

While MesonGS achieves the best overall performance due to its render-dependent importance score, our method provides a strong render-agnostic al-

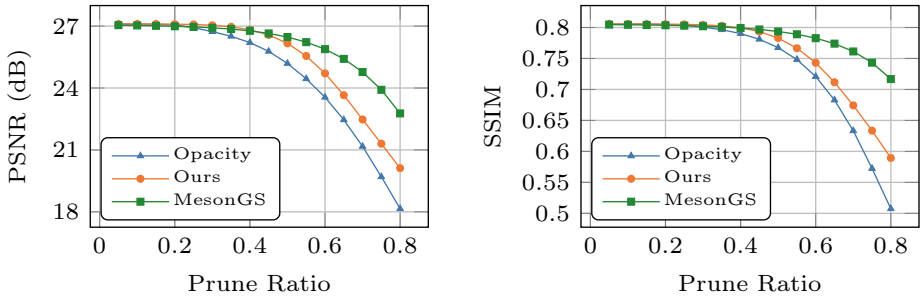


**Fig. 4: Comparison of pruning strategies (Mip-NeRF 360).** Our render-agnostic importance score degrades more gracefully than opacity-based pruning, while MesonGS' render-dependent score remains the most robust at higher prune ratios.

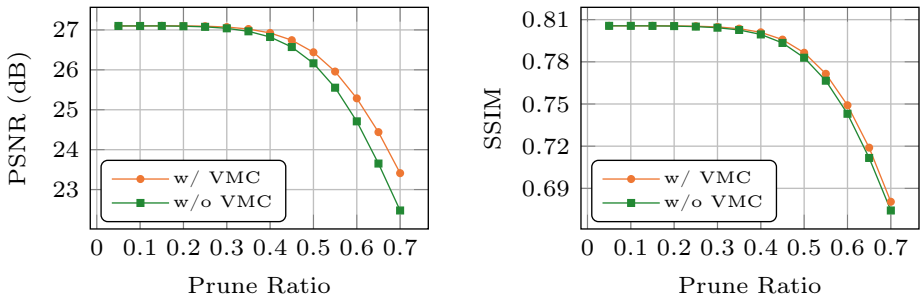


**Fig. 5: Impact of VMC on reconstruction quality (Mip-NeRF 360).** Spatial coverage is moderately recovered without compromising structural similarity.

ternative. The performance gap at high ratios likely stems from our lack of view-dependent signals, such as occlusion. That said, at low pruning ratios, all approaches perform similarly, as they primarily remove redundant primitives.

A key limitation of our approach is the absence of camera or view-dependent rendering signals, such as occlusion and view-specific contribution. This likely explains the remaining gap to MesonGS under high pruning ratios. Finally, at extreme pruning levels, all methods exhibit substantial quality degradation, indicating an inherent limit when too few primitives remain.

**Refinement.** *Voxel Mass Compensation (VMC)* is designed to conservatively restore spatial coverage lost to pruning. Empirical results indicate that VMC effectively fulfills this role: it improves reconstruction quality (PSNR) without compromising structural similarity (SSIM), particularly at prune ratios above 0.4 (see Fig. 5). However, at lower prune ratios (0.0 - 0.3), its effect is marginal, which is expected given the limited coverage loss in this range. Moreover, because VMC operates without any rendering feedback, it cannot fully compensate for severe information loss as sparsity increases, thereby limiting its corrective capacity

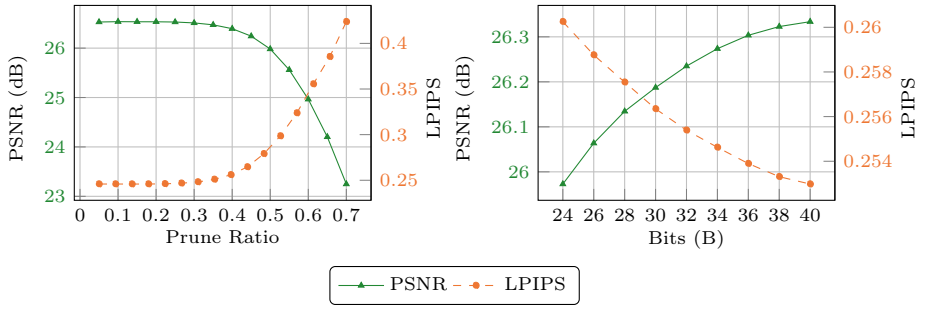


**Fig. 6: Hyperparameter sensitivity (Mip-NeRF 360).** PSNR and LPIPS across variations of $p$ (left, $B = 32$) and $B$ (right, $p = 0.45$). Ranges: $p \in [0.05, 0.7]$ and $B \in \{24, \ldots, 40\}$.

relative to rendering-aware alternatives under extreme pruning conditions. Nevertheless, this feedback-free design makes VMC computationally efficient while still providing a practical safeguard under aggressive pruning.

**Hyperparameter Sensitivity Analysis.** Figure 6 analyzes the sensitivity of RaPTGS to the pruning ratio ($p$) and the Spherical Harmonics (SH) bit budget ($B$). As shown on the left, the model demonstrates robustness to initial pruning, maintaining stable PSNR and LPIPS up to $p \approx 0.35$. Beyond this threshold, PSNR begins to degrade gracefully until $p \approx 0.5$, after which both metrics reflect a sharp deterioration in perceptual quality. The right plot illustrates the influence of the bit budget, showing that while lower bitrates (e.g., $B = 24$) incur a noticeable performance penalty, increasing $B$ yields steady visual improvements that eventually plateau, exhibiting diminishing returns beyond 36 bits.

### 4.2 Ablation Studies

**Pipeline Component Analysis.** Table 4a demonstrates the progressive effectiveness of our compression stages. Applying our multi-criteria pruning significantly reduces the raw model size, with an expected drop in rendering fidelity. However, the subsequent application of our training-free Voxel Mass Compensation (VMC) effectively recovers lost local coverage, improving reconstruction metrics without increasing the primitive count. Following refinement, degree-wise SH quantization compresses the view-dependent appearance attributes, yielding substantial storage savings while maintaining competitive visual quality. Finally, applying our two-stage spatial reordering strategy prior to entropy coding maximizes local redundancy in the serialized stream, further minimizing the overall storage footprint. The complete pipeline achieves an efficient compression ratio with minimal degradation in metrics compared to the uncompressed baseline.

In terms of runtime, pruning and VMC incur negligible overhead, whereas quantization, entropy coding and reordering dominate the processing time. Nev-

**Table 4:** (a) Ablation of our proposed compression pipeline stages on the Mip-NeRF 360 dataset. We report the cumulative impact on model size (MB), compression time (s) and reconstruction quality as each component is applied. All experiments use $p = 0.45$ and $B = 32$. (b) Ablation of importance score criteria and aggregation strategy on Mip-NeRF 360. All variants use fixed $p = 0.45$.

| **Stages** | PSNR↑ | SSIM↑ | LPIPS↓ | Size↓ | Time↓ |
|---|---|---|---|---|---|
| 3D-GS | 27.09 | 0.805 | 0.226 | 795.26 | - |
| + Pruning | 26.57 | 0.793 | 0.238 | 437.39 | 0.18 |
| + VMC | 26.74 | 0.795 | 0.237 | 437.39 | 0.21 |
| + SH Quantization | 26.40 | 0.788 | 0.249 | 119.93 | 3.31 |
| + Attribute Quantization | 26.24 | 0.778 | 0.255 | 32.66 | 5.13 |
| + Reordering | | | | | |
| + Hilbert Sort | 26.24 | 0.778 | 0.255 | 25.39 | 5.19 |
| + 2-Opt | 26.24 | 0.778 | 0.255 | 24.55 | 6.08 |
| Full Pipeline | 26.24 | 0.778 | 0.255 | 24.55 | 7.93 |

**(a)** Pipeline stages ablation.

| Variant | PSNR↑ | SSIM↑ | LPIPS↓ |
|---|---|---|---|
| only OS | 25.77 | 0.781 | 0.241 |
| only SAS | 25.93 | 0.782 | 0.242 |
| only SAE | 25.37 | 0.782 | 0.247 |
| w/o OS | 26.54 | 0.792 | 0.239 |
| w/o SAS | 26.53 | 0.792 | 0.239 |
| w/o SAE | 25.98 | 0.783 | 0.239 |
| SAE w/o clamp | 26.46 | 0.791 | 0.240 |
| Sum | 26.48 | 0.792 | 0.237 |
| **Max (Ours)** | **26.57** | **0.793** | **0.238** |

**(b)** Importance score criteria.

ertheless, the pipeline remains lightweight (7.93 s total), demonstrating that the compression gains are achieved with minimal burden.

**Effectiveness of Importance Criteria.** Table 4b presents an ablation of our multi-criteria importance score at a fixed pruning ratio ($p = 0.45$). Omitting any single proxy or using additive pooling instead of our proposed maximum pooling degrades performance, confirming that visibility, geometric specificity, and appearance complexity provide complementary signals that are best preserved jointly. Removing the geometric extent lower bound (SAE w/o clamp) leads to lower reconstruction quality, indicating that the proposed design may help preserve small, high-frequency primitives from premature culling.

## 5 Conclusion

In this work, we presented RaPTGS, a render-agnostic post-training compression pipeline for pre-trained 3DGS models under a strict model-only constraint. Our method prunes redundant Gaussians using a camera-independent importance score, restores local coverage through lightweight refinement, and further reduces storage via SH coefficient quantization, entropy coding, and spatial reordering. Across standard benchmarks, RaPTGS achieves an average compression ratio of 30× while maintaining competitive visual quality. For future work, we plan to address the heuristic nature of our importance score, particularly at high compression rates, by leveraging self-rendered synthetic views to derive lightweight visibility and occlusion cues for pruning and refinement while preserving the model-only setting.

## References


1. Bagdasarian, M.T., Knoll, P., Li, Y., Barthel, F., Hilsmann, A., Eisert, P., Morgenstern, W.: 3dgs.zip: A survey on 3d gaussian splatting compression methods. Computer Graphics Forum **44**(2), e70078 (2025)
2. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 5470–5479 (2022)
3. Chen, Y., Wu, Q., Li, M., Lin, W., Harandi, M., Cai, J.: Fast feedforward 3d gaussian splatting compression. In: The Thirteenth International Conference on Learning Representations (2025)
4. Chen, Y., Wu, Q., Lin, W., Harandi, M., Cai, J.: Hac: Hash-grid assisted context for 3d gaussian splatting compression. In: European Conference on Computer Vision. pp. 422–438. Springer (2024)
5. Duan, Y., Wei, F., Dai, Q., He, Y., Chen, W., Chen, B.: 4d-rotor gaussian splatting: towards efficient novel view synthesis for dynamic scenes. In: ACM SIGGRAPH 2024 Conference Papers. pp. 1–11 (2024)
6. Fan, Z., Wang, K., Wen, K., Zhu, Z., Xu, D., Wang, Z.: Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. Advances in neural information processing systems **37**, 140138–140158 (2024)
7. Girish, S., Gupta, K., Shrivastava, A.: Eagles: Efficient accelerated 3d gaussians with lightweight encodings. In: European Conference on Computer Vision. pp. 54–71. Springer (2024)
8. Hanson, A., Tu, A., Singla, V., Jayawardhana, M., Zwicker, M., Goldstein, T.: Pup 3d-gs: Principled uncertainty pruning for 3d gaussian splatting. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 5949–5958 (2025)
9. Hedman, P., Philip, J., Price, T., Frahm, J.M., Drettakis, G., Brostow, G.: Deep blending for free-viewpoint image-based rendering. ACM Transactions on Graphics (ToG) **37**(6), 1–15 (2018)
10. Hilbert, D.: Über die stetige abbildung einer linie auf ein flächenstück. Mathematische Annalen **38**(3), 459–460 (1891)
11. Huang, Y., Pang, J., Zhu, F., Tian, D.: Entropygs: an efficient entropy coding on 3d gaussian splatting. In: ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 11732–11736. IEEE (2026)
12. Kathariya, B., Lee, D.Y., Huang, T.W., Shao, T., Yin, P., Su, G.M.: Gaussian splatting: State-of-the-arts and future trends. In: 2025 International Conference on Computing, Networking and Communications (ICNC). pp. 876–881. IEEE (2025)
13. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G., et al.: 3d gaussian splatting for real-time radiance field rendering. ACM Trans. Graph. **42**(4), 139–1 (2023)
14. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics (ToG) **36**(4), 1–13 (2017)
15. Lee, J.C., Rho, D., Sun, X., Ko, J.H., Park, E.: Compact 3d gaussian representation for radiance field. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 21719–21728 (2024)
16. Liu, L., Chen, Z., Jiang, W., Wang, W., Xu, D.: Hemgs: A hybrid entropy model for 3d gaussian splatting data compression. arXiv preprint arXiv:2411.18473 (2024)

17. Liu, Y., Luo, C., Fan, L., Wang, N., Peng, J., Zhang, Z.: Citygaussian: Real-time high-quality large-scale scene rendering with gaussians. In: European Conference on Computer Vision. pp. 265–282. Springer (2024)
18. Lu, T., Yu, M., Xu, L., Xiangli, Y., Wang, L., Lin, D., Dai, B.: Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 20654–20664 (2024)
19. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
20. Morgenstern, W., Barthel, F., Hilsmann, A., Eisert, P.: Compact 3d scene representation via self-organizing gaussian grids. In: European conference on computer vision. pp. 18–34. Springer (2024)
21. Navaneet, K., Pourahmadi Meibodi, K., Abbasi Koohpayegani, S., Pirsiavash, H.: Compgs: Smaller and faster gaussian splatting with vector quantization. In: European Conference on Computer Vision. pp. 330–349. Springer (2024)
22. Niedermayr, S., Stumpfegger, J., Westermann, R.: Compressed 3d gaussian splatting for accelerated novel view synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10349–10358 (2024)
23. Papantonakis, P., Kopanas, G., Kerbl, B., Lanvin, A., Drettakis, G.: Reducing the memory footprint of 3d gaussian splatting. Proceedings of the ACM on Computer Graphics and Interactive Techniques **7**(1), 1–17 (2024)
24. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
25. Ramlot, B., Courteaux, M., Lambert, P., Van Wallendael, G.: Potr: Post-training 3dgs compression. arXiv preprint arXiv:2601.14821 (2026)
26. Skilling, J.: Programming the hilbert curve. AIP Conference Proceedings **707**(1), 381–387 (04 2004)
27. Tian, B., Gao, Q., Xianyu, S., Cui, X., Zhang, M.: Flexgaussian: Flexible and cost-effective training-free compression for 3d gaussian splatting. In: Proceedings of the 33rd ACM International Conference on Multimedia. pp. 7287–7296 (2025)
28. Wang, H., Zhu, H., He, T., Feng, R., Deng, J., Bian, J., Chen, Z.: End-to-end rate-distortion optimized 3d gaussian representation. In: European Conference on Computer Vision. pp. 76–92. Springer (2024)
29. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13**(4), 600–612 (2004)
30. Wu, T., Yuan, Y.J., Zhang, L.X., Yang, J., Cao, Y.P., Yan, L.Q., Gao, L.: Recent advances in 3d gaussian splatting. Computational Visual Media **10**(4), 613–642 (2024)
31. Xie, S., Liu, J., Zhang, W., Ge, S., Pan, S., Tang, C., Bai, Y., Zhang, C., Fan, X., Wang, Z.: Sizegs: Size-aware compression of 3d gaussian splatting via mixed integer programming. In: Proceedings of the 33rd ACM International Conference on Multimedia. pp. 8214–8223 (2025)
32. Xie, S., Zhang, W., Tang, C., Bai, Y., Lu, R., Ge, S., Wang, Z.: Mesongs: Post-training compression of 3d gaussians via efficient attribute transformation. In: European conference on computer vision. pp. 434–452. Springer (2024)
33. Yang, Q., Van Der Auwera, G., Li, Z.: Lightweight 3d gaussian splatting compression via video codec. In: 2026 Data Compression Conference (DCC). pp. 362–371. IEEE (2026)

34. Yu, Z., Chen, A., Huang, B., Sattler, T., Geiger, A.: Mip-splatting: Alias-free 3d gaussian splatting. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 19447–19456 (2024)
35. Zhan, Y.T., Ho, C.Y., Yang, H., Chen, Y.H., Chiang, J.C., Liu, Y.L., Peng, W.H.: CAT-3DGS: A context-adaptive triplane approach to rate-distortion-optimized 3DGS compression. In: Proceedings of the Thirteenth International Conference on Learning Representations (ICLR) (2025)
36. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018)
37. Zhang, Z., Song, T., Lee, Y., Yang, L., Peng, C., Chellappa, R., Fan, D.: Lp-3dgs: Learning to prune 3d gaussian splatting. Advances in Neural Information Processing Systems **37**, 122434–122457 (2024)