# ELMGS: Enhancing memory and computation scaLability through coMpression for 3D Gaussian Splatting

Muhammad Salman Ali<sup>1,2</sup>, Sung-Ho Bae\*<sup>2</sup>, Enzo Tartaglione\*<sup>1</sup>

LTCI, Télécom Paris, Institut Polytechnique de Paris, France

<sup>2</sup> Kyung Hee University, Republic of Korea

{salmanali, shbae}@khu.ac.kr,
enzo.tartaglione@telecom-paris.fr

## **Abstract**

3D models have recently been popularized by the potentiality of end-to-end training offered first by Neural Radiance Fields and most recently by 3D Gaussian Splatting models. The latter has the big advantage of naturally providing fast training convergence and high editability. However, as the research around these is still in its infancy, there is still a gap in the literature regarding the model's scalability. In this work, we propose an approach enabling both memory and computation scalability of such models. More specifically, we propose an iterative pruning strategy that removes redundant information encoded in the model. We also enhance compressibility for the model by including in the optimization strategy a differentiable quantization and entropy coding estimator. Our results on popular benchmarks showcase the effectiveness of the proposed approach and open the road to the broad deployability of such a solution even on resource-constrained devices.

# 1. Introduction

Recent advancements in novel view synthesis, driven by the emergence of Neural Radiance Fields (NeRFs) [26], have enabled the generation of new views of 3D scenes or objects by interpolating from a sparse collection of images with known camera angles. While NeRFs utilize an implicit neural representation to capture the volumetric radiance field for rendering novel views, they face significant limitations due to high memory requirements and computational complexity. This results in slow training and rendering times, prompting the exploration of alternative methods that often involve a trade-off between quality and complexity.

Using 3D voxel grids on the GPU combined with a multiresolution hash encoding of the input [27] significantly reduces the operations needed and permits real-time performance. However, NeRFs still require high computational complexity due to which they cannot be deployed on edge devices with limited computational resources. Recent research has focused on reducing the memory footprint of NeRF by compressing the learned parametrizations on regular grids. These include vector quantized feature encoding [23], learned tensor decomposition [5], and frequency domain transformations [35].

Recently, a novel technique known as differentiable 3D Gaussian Splatting (3DGS) has been introduced to create a sparse adaptive representation of scenes, enabling highspeed rendering on GPUs. This method represents the scene as a collection of 3D Gaussians characterized by shape (covariance) and appearance (opacity, color, spherical harmonics) parameters, refined through differentiable rendering to align with a given set of images. This method offers a significant advantage over NeRF techniques in terms of both training and rendering speed. The efficiency stems from the straightforward process of projecting 3D Gaussians onto the 2D image space and subsequently rendering the view by combining multiple projected Gaussians with their opacity through rasterization. As a result, scenes can be rendered in real-time on a single GPU. Another notable benefit is that, unlike NeRF, the 3D scene's structure is explicitly stored in the parameter space rather than implicitly encoded in NeRF models.

Nevertheless, the optimized scenes generated by this approach frequently consist of millions of Gaussians, leading to parameters that are orders of magnitude larger than those in NeRFs. Consequently, the substantial storage and memory requirements make rendering a challenging task on devices with limited video memory, such as handheld devices or head-mounted displays. Although the specialized compute pipeline utilized for Gaussian rendering achieves real-

<sup>\*</sup>Corresponding Authors

This article has been accepted for publication at the 2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV 2025).

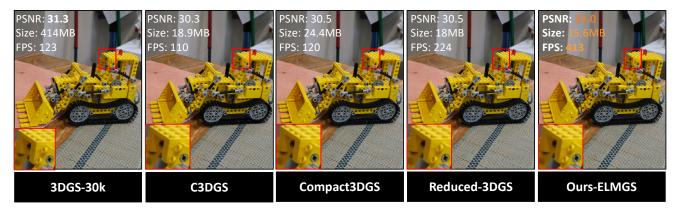


Figure 1. Qualitative comparison of ELMGS with 3DGS [19], C3DGS [28], Compact3DGS [21], and Reduced-3DGS [30]. With our proposed method we can achieve compression rates of about  $27 \times$  with indiscernible loss in visual quality and significantly better rendering speed as compared to other methods.

time performance on high-end GPUs, the seamless integration of this pipeline into VR/AR environments or games, working in tandem with hardware rasterization of polygon models, presents challenges.

This paper focuses on enhancing the efficiency of Gaussian Splatting representations without compromising their fidelity, aiming to accelerate rendering speed for their utilization across diverse applications, including low-storage or low-memory IoT devices and AR/VR headsets. Our key insight is that within the final optimized scene, many Gaussians are redundant, characterized by opacity levels close to zero, rendering them ineffective. We begin with a pretrained optimized Gaussian scene and prune it based on gradient and opacity levels, followed by fine-tuning to achieve superior performance compared to the baseline optimized scene. Subsequently, we employ quantization-aware training (QAT) to further compress the scene size. Upon completion of QAT, the scene undergoes entropy encoding, resulting in substantial compression gains along with performance improvements as shown in Fig. 1.

Our main contributions are listed here below.

- We leverage the optimized Gaussians as a 3D prior for pruning, facilitating the removal of redundant Gaussians while fine-tuning the remaining ones to more accurately represent scene features (Sec. 3.1).
- Our compression pipeline achieves an improved balance between scene fidelity and compression surpassing the baseline (Sec. 4.2).
- By integrating pruning with quantization-aware training, we enhance the compression of the scene representation and subsequently employ entropy coding for more efficient scene compression (Sec. 3.2 and Sec. 3.3).

## 2. Related Works

## 2.1. Novel View Synthesis

Early methods for novel view synthesis using Convolutional Neural Networks (CNNs) encountered difficulties with Multi-View Stereo (MVS) geometry and temporal flickering [12, 16]. The shift to volumetric representations began with Soft3D [1], followed by techniques that integrated deep learning with volumetric raymarching [32]. NeRFs, proposed by Mildenhall et al. [26], aimed to improve view synthesis quality but suffered from slow processing due to the use of a large Multi-Layer Perceptron (MLP) and dense sampling. Subsequent approaches like Mip-NeRF360 [2] sought a balance between quality and speed. Recent advancements focus on enhancing training and rendering speeds through optimized spatial data structures and MLP adjustments [7, 17, 27], with methods like InstantNGP [27] and Plenoxels [14] innovating for faster computations and eliminating neural networks, respectively. Despite progress, challenges persist in rendering speeds, image quality, and empty space representation. Conversely, 3DGS [19] offers superior quality and speed without implicit learning [2].

#### 2.2. Differentiable Gaussian Splatting

The Differentiable Gaussian splatting technique [19], extends the EWA volume splatting method [37] to accurately compute the projections of 3D Gaussian kernels onto the 2D image plane. Furthermore, it employs differentiable rendering to iteratively optimize the number and parameters of the Gaussian kernels utilized for scene representation.

The target final scene representation consists of a collection of 3D Gaussians, each characterized by a covariance matrix  $\Sigma \in \mathbb{R}^{3 \times 3}$  centered at location  $x \in \mathbb{R}^3$ . The covariance matrix can be parametrized by a rotation matrix R and a scaling matrix S. For separate optimization of R and S, Kerbl  $et\ al.\ [19]$  use a quaternion g to represent rotation

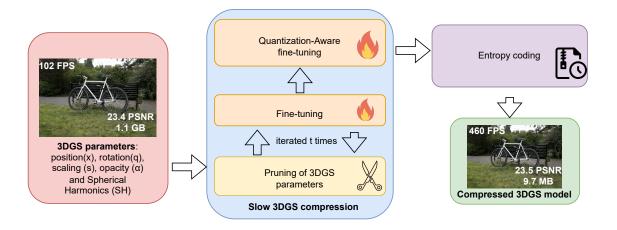


Figure 2. ELMGS begins with a pre-trained 3DGS scene and performs iterative pruning with finetuning to remove less significant Gaussians, followed by Quantization-Aware finetuning. The quantized model is then entropy-encoded to generate the final compressed scene.

and a vector s to represent scaling, both of which can be converted into their corresponding matrices. Additionally, each Gaussian possesses its opacity  $\alpha \in [0,1]$  and a set of spherical harmonics (SH) coefficients for reconstructing a view-dependent color.

To initialize the optimization process, 3DGS employs a point cloud obtained through a standard Structure from Motion (SfM) method [33]. During the training phase, 3DGS undertakes the rendering of training viewpoints and minimizes the loss between the ground truth and rendered images in the pixel space. The loss is a combination of  $\ell_1$ loss and SSIM loss in the pixel space. Subsequently, it iteratively prunes points with small opacity parameters and introduces new ones when the gradient is considered "substantial" (where this is mediated through a thresholding function). The optimization involves adjusting the position (x), rotation (q), scaling (s), opacity  $(\alpha)$ , and Spherical Harmonics (SH) coefficients of each 3D Gaussian to ensure that the rendered 2D Gaussians align with the training images. 3DGS showcases efficient training and real-time rendering performance, achieving or even exceeding the visualization quality benchmarks set by leading NeRF techniques [19].

## 2.3. Compression of 3D Gaussian Splatting

The unstructured nature of 3DGS presents challenges for compression compared to NeRFs [6, 11]. Recent research has introduced structural modifications to improve compressibility [6, 25]. For example, Scaffold-GS employs anchor points to distribute local 3D Gaussians and adjusts their attributes according to viewing direction and distance within the view frustum. The Hash-grid Assisted Context (HAC) [6] framework builds on Scaffold-GS by co-learning a compact hash grid that models the context of anchor at-

tributes. RadSplat [29] uses NeRFs as a prior for optimizing 3DGS and incorporates a pruning technique to achieve a compact scene representation. MiniSplatting [10] addresses the inefficient spatial distribution of Gaussians by introducing densification strategies, such as blur split and depth reinitialization, leading to a more uniform and efficient spatial arrangement.

Niedermayret al. [28] proposed a compression framework that maintains the original 3DGS parameters while compressing directional colors and Gaussian parameters. This framework uses sensitivity-aware vector clustering for pruning and QAT, achieving compression rates up to 30× with negligible performance loss relative to the original 3DGS. Likewise, Compact3DGS [21] employs a learnable mask to prune Gaussians and uses a grid-based neural field to compactly represent view-dependent colors, replacing spherical harmonics. Additionally, Compact3DGS leverages vector quantization to efficiently encode the geometric attributes of Gaussians into codebooks.

Previous research on 3DGS compression has explored vector quantization schemes [21, 28], which, while beneficial for compression, pose challenges for hardware implementation. Previous work in compression has shown that uniform quantization schemes are the most hardware-friendly [15, 24]. To the best of our knowledge, we are among the first works [6] to propose a QAT method based on a scalar uniform quantization scheme.

Our method aims to enhance or replace existing masking strategies for insignificant Gaussians (see Sec. 4.2), provide a more hardware-compatible quantization scheme, and surpass the performance of current techniques. However, this paper focuses exclusively on the vanilla 3DGS variant, given its broad applicability.

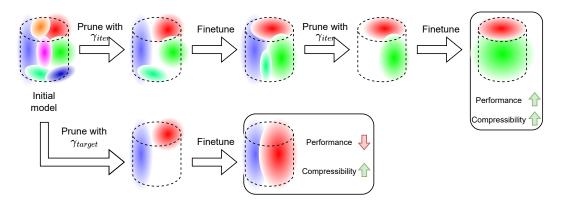


Figure 3. Effect of gradual pruning to the 3DGS model: a gradual removal allows the model to self-adjust and to better fit the scene/object.

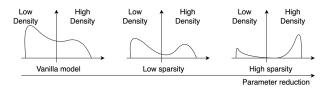


Figure 4. Effect of gradual pruning to the opacity values. We have two effects: (i) the number of parameters in our model reduces; (ii) the frequency around specific low and high density is increased.

# 3. Methodology

3DGS models, despite showcasing remarkable performances and enhanced editability, unfortunately, fall short in storage memory, as they typically require massive memory. In our approach, we propose a simple yet effective strategy to compress such a model, that can be decomposed in three distinct steps (Fig. 2). The first one employs gradient and opacity-aware pruning (discussed in Sec. 3.1), where we iteratively remove parameters from the 3DGS model and fine-tune the model. Implicitly, this provides a strong prior to the geometry of the scene. Later, we perform the quantization-aware fine-tuning step (discussed in Sec. 3.2), which prepares the field for the entropy coding step (detailed in Sec. 3.3).

#### 3.1. Gradient and Opacity Aware Pruning (GAP)

3DGS typically requires several million Gaussians to effectively model a standard scene. Each Gaussian entails 59 parameters, resulting in a storage size considerably larger than most NeRF methodologies, such as Mip-NeRF360 [2], K-planes [13] and instantNGP [27]. This renders it inefficient for certain applications, particularly those involving edge devices. Our focus lies in parameter reduction. In the original training process of 3DGS, the authors pruned and densified Gaussians up to a specified number of iterations. Pruning was based on a predetermined opacity threshold. However, upon examination of the final opti-

mized scene (see Fig.6c), a substantial number of Gaussians were found to fall below this opacity threshold, rendering them essentially redundant. Consequently, we adopt a pruning strategy based on the quantile functions of opacity and its corresponding gradient. The quantile function applies a threshold ( $\gamma_{iter}$ ) to identify a percentage of parameters that fall below the threshold value ( $\gamma_{iter}$ ). If both the opacity and gradient of a given Gaussian fall below the specified threshold,( $\gamma_{iter}$ ) the Gaussian can be pruned with minimal or no adverse impact on the rendered scene quality. We can define GAP as:

$$\Sigma' = \begin{cases} & \left[ |\Sigma_i^{\alpha}| \ge \mathcal{Q}_{|\Sigma^{\alpha}|}(\gamma_{\text{iter}}) \right] \\ \Sigma_i & \text{if} & \vee \\ & \left[ |\nabla \Sigma_i| \ge \mathcal{Q}_{|\nabla \Sigma|}(\gamma_{\text{iter}}) \right] \end{cases}$$
 (1)
$$0 \quad \text{otherwise,}$$

where  $\nabla \Sigma_i$  denotes the gradient for the *i*-th Gaussian,  $\Sigma_i^{\alpha}$  denotes the  $\alpha$  value of the *i*-th Gaussian, and  $\mathcal{Q}_{|\Sigma^{\alpha}|}(.)$  represents the quantile function for the opacity,  $\mathcal{Q}_{|\nabla\Sigma|}(.)$  is the quantile function for the gradients of the Gaussians, and  $\gamma \in [0,1]$  denotes the fraction of Gaussians to be removed.

This GAP pruning process, coupled with periodic fine-tuning, not only enhances performance but also yields significant compression gains. The pruning and fine-tuning strategy allows us to eliminate redundant Gaussians and refine the remaining ones to more accurately represent the scene compared to the baseline. We know from previous works on scene rendering that a low iterative pruning strategy enables access to much sparser models while still maintaining high fidelity [8]; however, it is still unclear what could it be the effect on 3DGS models. We argue that, given a target sparsity  $\gamma_{\text{target}}$ , employing a gradual pruning for t iterations, and for instance applying a sparsification process on the 3DGS model applying

$$\gamma_{\text{iter}} = 1 - (1 - \gamma_{\text{target}})^{\frac{1}{t}}, \tag{2}$$

would lead to enhanced results. This happens thanks to two effects. First, the fine-tuning process (that follows the pruning step) adjusts the covariance  $\Sigma$  (still minimizing the rendering loss) such that solid surfaces will have higher values, while semi-transparent artifacts remain at lower values and can be removed at the next iteration. Second, the slow iterative process prevents the optimization algorithm from falling into a sub-optimal local minimum. Specifically for this reason, although a pruning mechanism is already employed in 3DGS, it is important to begin with an overparametrized but well-performing model. Some parallels can be drawn also from traditional deep learning [4,34]. Fig. 3 pictures the effect of either performing a slow pruning or a one-shot one, where the target sparsity level is reached at once.

## 3.2. Quantization-Aware Training (QAT)

We also employ Learned Step Size Quantization (LSQ) [9] to optimize the quantization mapping for all parameters in 3DGS. LSQ provides a direct way to approximate the gradient for the quantizer step size by accounting for quantized state transitions. This method facilitates more precise optimization by treating the step size as a model parameter. Furthermore, LSQ employs a straightforward heuristic to effectively balance step size updates with weight updates, making it well-suited for QAT on 3DGS.

Given parameter to quantize  $\Sigma$ , quantizer step size  $\Delta$ , the number of positive and negative quantization levels  $Q_P$  and  $Q_N$ , respectively, we define a quantizer that computes  $\bar{\Sigma}$ , a quantized and integer-scaled representation of the data, and  $\hat{\Sigma}$ . This results in a quantized representation of the data at the same scale as  $\Sigma$ :

$$\bar{\Sigma} = \left[ \text{clip}\left(\frac{\Sigma}{\Delta}, -Q_N, Q_P\right) \right], \tag{3}$$

$$\hat{\Sigma} = \bar{\Sigma} \cdot \Delta,\tag{4}$$

where  $\operatorname{clip}(z,r_1,r_2)$  adjusts z by setting values below  $r_1$  to  $r_1$  and values above  $r_2$  to  $r_2$ . Additionally,  $\lfloor z \rfloor$  rounds z to the nearest integer. With b-bit encoding, for unsigned data, the feature values will range from  $Q_N=0$  to  $Q_P=2^b-1$ . For signed data,  $Q_N=-2^{(b-1)}$  and  $Q_P=2^{(b-1)}-1$ .

LSQ introduces a mechanism to learn the scale  $(\Delta)$  based on the training loss by incorporating the following gradient through the quantizer to the step size parameter:

$$\frac{\partial \hat{\Sigma}}{\partial \Delta} = \begin{cases}
-\frac{\Sigma}{\Delta} + \lfloor \frac{\Sigma}{\Delta} \rceil & \text{if } -Q_N < \frac{\Sigma}{\Delta} < Q_P \\
-Q_N & \text{if } \frac{\Sigma}{\Delta} \le -Q_N \\
Q_P & \text{if } \frac{\Sigma}{\Delta} \ge Q_P.
\end{cases}$$
(5)

This gradient is derived by employing the straight-through estimator [3] to approximate the gradient through the round

function as a pass-through operation (while retaining the round operation itself for the sake of differentiating downstream operations), and differentiating all other operations in (3) and (4) as usual.

# 3.3. Entropy Encoding

GAP significantly reduces the number of Gaussians, which are then subjected to quantization-aware training (QAT) to quantize all Gaussian parameters. The resulting quantized feature representation of the Gaussians enables high compression rates using entropy coding (EC). Finally, the quantized Gaussians undergo compression using the LZ77 [36] algorithm. By arranging the Gaussians according to their positions along a Z-order curve in Morton order (MO), we can further exploit coherence and enhance the effectiveness of entropy encoding. We can expect that our iterative pruning approach is also enhancing the model's compressibility, and for instance the entropy: this solves the pressing issue of including entropy encoding in the optimization loop (and bypassing practical obstacles including the differentiability of such an element). For instance, as suggested in Sec. 3.1, thanks to the pruning approach we expect high peaks around high-density values to rise naturally, showcasing a trend visualized in Fig. 4. Evidently, as the frequency around specific values increases, the first-order entropy of the encoded model (acting as an upper bound) is also minimized.

In the next section, we will show our empirical findings in the typical benchmarked setups.

#### 4. Experiments and Results

In this section, we are going to present our empirical findings on typical well-established benchmarks in the 3DGS community. First, we will provide the implementation details for our employed method alongside an overview of the benchmarked datasets and the evaluation metrics (Sec. 4.1); then we present the qualitative and quantitative results (Sec. 4.2); and finally, a detailed ablation study (Sec. 4.3). More results are also presented in the Supplementary Material, alongside the code, which will be opensourced upon acceptance of the article.

#### 4.1. Implementation Details

For all our experiments, we utilize the publicly available official code repository of 3DGS [19] provided by its authors, without altering the hyperparameters used for training compared to 3DGS. We initiate pruning with the optimized Gaussians trained for 30,000 iterations and apply pruning using (1) with  $\gamma_{\text{iter}}$  where  $\gamma_{\text{iter}} \in [0.375, 0.6]$ . The model undergoes sequential pruning with the same  $\gamma_{\text{iter}}$  value after every 500 iterations until 35,000 iterations, followed by further finetuning for 5,000 iterations. In most experiments, pruning and finetuning yield superior performance



Figure 5. Comparison of ground truth images from the test set of bicycle, drjohnson, and truck scenes between ELMGS "oursbig", "ours-medium" and "ours-small" compressed representation and 3DGS-30k.

Table 1. Comparison with SOTA methods for novel view synthesis. † Reported from [19]. Red indicating the best performance, followed by yellow and green.

	Mip-NeRF360				Tanks&Temples				Deep Blending						
Method	SSIM <sup>↑</sup>	$\mathbf{LPIPS}^{\downarrow}$	PSNR <sup>↑</sup>	$\mathbf{FPS}^{\uparrow}$	$\mathbf{Mem}^{\downarrow}$	SSIM <sup>↑</sup>	$\mathbf{LPIPS}^{\downarrow}$	PSNR <sup>↑</sup>	$\mathbf{FPS}^{\uparrow}$	$\mathbf{Mem}^{\downarrow}$	SSIM <sup>↑</sup>	$\mathbf{LPIPS}^{\downarrow}$	PSNR <sup>↑</sup>	FPS <sup>↑</sup>	Mem↓
3DGS-7k <sup>†</sup>	0.770	0.279	25.60	160	523MB	0.767	0.280	21.20	197	270MB	0.875	0.317	27.78	172	386MB
$3DGS-30k^{\dagger}$	0.815	0.214	27.21	134	734MB	0.841	0.183	23.14	154	411MB	0.903	0.243	29.41	137	676MB
C3DGS [28]	0.801	0.238	26.98	113	28.8MB	0.832	0.194	23.32	149	17.3MB	0.898	0.253	29.38	128	25.3MB
Compact3DGS [21]	0.798	0.247	27.08	128	48.8MB	0.831	0.201	23.32	185	39.4MB	0.901	0.258	29.79	181	43.2MB
Compact3DGS+PP [21]	0.797	0.247	27.03	-	29.1MB	0.831	0.202	23.32	-	20.9MB	0.900	0.258	29.73	-	23.8MB
Reduced-3DGS [30]	0.809	0.226	27.10	284	29.0MB	0.840	0.188	23.57	433	14.0MB	0.902	0.249	29.63	360	18.0MB
Ours-big	0.808	0.235	27.58	216	62.2MB	0.845	0.191	24.06	418	27.9MB	0.899	0.255	29.57	321	40.8MB
Ours-medium	0.792	0.264	27.31	290	38.6MB	0.838	0.191	24.08	600	18.8MB	0.897	0.261	29.48	440	23.5MB
Ours-small	0.779	0.286	27.00	361	25.8MB	0.825	0.233	23.90	624	11.6MB	0.894	0.273	29.24	568	12.3MB

compared to the baseline. Subsequently, we train the model using QAT with the LSQ method and further finetune it for 5,000 iterations on QAT. For QAT, we encode Spherical Harmonics (SH) features into 8 bits, while all other features such as opacity, scaling, rotation, and XYZ positions are quantized into 32 bits. Finally, the finetuned model after QAT undergoes entropy coding to complete our compression pipeline. Our method extends the training time of 3DGS by half, totaling 45,000 iterations.

**Datasets.** We evaluate the effectiveness of our compression and rendering method across various scenes, including the Mip-Nerf360 [2] indoor and outdoor scenes, as well as two scenes from the Tanks&Temples [20] and Deep Blending [16] datasets.

**Evaluation.** To ensure a fair comparison, we maintain consistency with the train-test split used in Mip-NeRF360 [2] and 3DGS [19], and directly present the metrics for other methods as reported in [19]. Our evaluation encompasses

Table 2. Proposed compression pipeline performance with various levels of pruning defined by  $\gamma_{\text{iter}}$ .

		Mip-l	NeRF360			Tanks	&Temples		Deep Blending			
$\gamma_{\mathrm{iter}}$	SSIM <sup>↑</sup>	<b>PSNR</b> <sup>↑</sup>	LPIPS $^↓$	$\mathbf{Mem}^{\downarrow}$	SSIM <sup>↑</sup>	<b>PSNR</b> <sup>↑</sup>	LPIPS↓	$\mathbf{Mem}^{\downarrow}$	SSIM <sup>↑</sup>	PSNR <sup>↑</sup>	LPIPS↓	Mem↓
0.375	0.808	27.579	0.235	62.23MB	0.845	24.059	0.191	27.90MB	0.898	29.558	0.250	68.29MB
0.450	0.792	27.311	0.264	38.64MB	0.838	24.084	0.209	18.82MB	0.899	29.571	0.255	40.83MB
0.500	0.779	26.999	0.286	25.76MB	0.825	23.903	0.233	11.65MB	0.897	29.482	0.261	23.53MB
0.550	0.741	26.214	0.337	12.78MB	0.802	23.451	0.268	6.43MB	0.894	29.241	0.273	12.32MB
0.600	0.692	25.142	0.394	5.73MB	0.765	22.694	0.318	2.99MB	0.887	28.646	0.294	5.40MB

standard metrics such as SSIM, PSNR, and LPIPS, along with the FPS and average memory consumption across all datasets.

#### 4.2. Results

Quantitative Comparison. Table 1 compares our method with existing state-of-the-art approaches. C3DGS [28], Compact3DGS [21], and Reduced-3DGS [30] exhibit performance similar to the 3DGS-30k baseline, with a minor improvement in rendering speed. Only Reduced-3DGS [30] shows a notable FPS increase, improving by 2.5×. In contrast, our "Ours-big" variant surpasses the 3DGS baseline, achieving a 14× compression ratio and a 2.2× increase in FPS. The "Ours-medium" variant maintains baseline performance while offering a  $22\times$  compression ratio and a  $3\times$ increase in FPS. Notably, the "Ours-small" variant achieves a remarkable 38× compression ratio and a 4× increase in rendering speed, averaging around 520 FPS across all datasets. Our method significantly reduces the model's memory footprint, making it competitive with NeRF approaches and addressing a key limitation of 3DGS models. The substantial compression is especially noteworthy given that many Gaussians in the original 3DGS are non-essential. Qualitative Comparison. Fig. 5 compares the ELMGS "Ours-big", "Ours-medium", and "Ours-small" models with the 3DGS-30k baseline. For the bicycle scene, "Ours-big" not only delivered better visual quality but also achieved a compression ratio of 14× with about 2.5× improvement in rendering speed. Similarly, "Ours-medium" provided a better PSNR with a compression ratio of approximately 28× and an increase in rendering speed by about 4× on the drjohnson scene. "Ours-small" achieved similar visual quality for the truck scene while achieving a compression ratio of about 32× and an FPS gain of about 4x. Please refer to the Supplementary Material for additional images and further qualitative comparisons.

#### 4.3. Ablation Studies

Effect of different Pruning levels. Table 2 illustrates the impact of different levels of pruning on the benchmark datasets. Across all datasets, it is evident that even with high pruning levels, our proposed method can still deliver reasonable performance. For the smallest Gaussian splats,

Table 3. The impact of each ELMGS module on the garden scene from the Mip-NeRF360 dataset, including Gradient and Opacity Aware Pruning (GAP), Learned Step size Quantization (LSQ), Morton Ordering (MO), and Entropy Coding (EC).

GAP	LSQ	EC	MO	PSNR	Mem	Comp
X	Х	Х	X	27.296	1.4GB	1×
$\checkmark$	X	X	X		306MB	
$\checkmark$	X	$\checkmark$	X	27.026	257MB	$5.5 \times$
$\checkmark$	X	$\checkmark$	$\checkmark$	27.026	252MB	5.5×
$\checkmark$	$\checkmark$	X	X	27.023	306MB	$5 \times$
$\checkmark$	$\checkmark$	$\checkmark$	X	27.023	70MB	$19 \times$
$\checkmark$	$\checkmark$	$\checkmark$	$\checkmark$	27.023	64MB	$22\times$

the average PSNR is 25.7 with an average size of 6.4 MBs, while for the least compressed Gaussian splats, the average PSNR is 27.04 with an average size of 59 MBs. By varying the pruning levels, we are able to achieve significantly smaller Gaussian splats with good performance, paving the way for them to be deployed in smaller end-to-end devices where memory footprint is crucial.

**Quantization Bits.** Gaussian splats exhibit sensitivity to quantization bits, with only spherical harmonics (SH) features akin to those in deep neural networks that can be quantized to lower bit depths without loss in information. However, other attributes like opacity, scaling, rotation, and XYZ positions pose challenges for quantization due to their intricate nature. To explore the impact of varying quantization bit levels on the Tanks&Temples dataset, we conducted experiments across different levels of quantization bits for various Gaussian parameters. The results are presented in Fig. 6a. For all the experiments, the spherical harmonics (SH) features were fixed at 8 bits. Fig. 6a illustrates a notable decline in performance as quantization bits decrease, while the compression ratio remains the same. Therefore, to minimize quantization errors, we adopted 32 bits as the standard setting for all experiments.

FPS Gain with ELMGS. ELMGS significantly boosts the FPS rate of 3DGS. On the Tanks&Temples dataset, our method achieves an FPS of over 600 while maintaining state-of-the-art performance, as shown in Fig. 6b. The renderings were performed using an NVIDIA RTX-3090 using Kerb *et al.* [19] rasterization approach, with the final

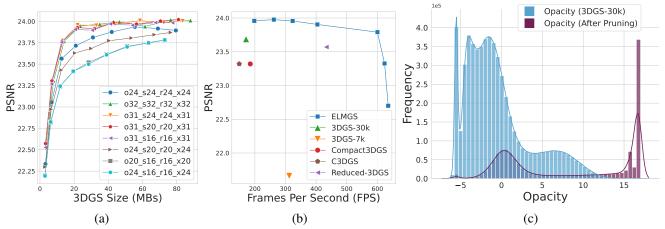


Figure 6. Tradeoff of performance and size at different bit-depths where opacity (o), scale (s), rotation (r), XYZ position (x) (a) and in terms of FPS on the Tanks&Temples dataset (b), and opacity distribution before and after GAP for the garden scene from the Mip-NeRF360 dataset (c).

FPS averaged over three separate runs. Compared to Compact3DGS [21], C3DGS [28], and the 3DGS-30k baseline, we observe an approximate  $4\times$  and about  $2\times$  improvement compared to Reduced-3DGS [30] in FPS while maintaining better/ similar performance.

Impact of each module on compression. Table 3 shows the performance and compression impact of each module within our proposed method. With the integration of GAP, our approach achieves a compression gain of  $5\times$ , while the introduction of LSQ does not directly affect compression. However, employing Morton ordering (MO) and entropy encoding (EC) on quantized Gaussians yields substantial gains  $(22\times)$  compared to solely applying EC on pruned Gaussians  $(5.5\times)$  signifying the importance of each module in our proposed method.

Impact of GAP on Opacity. GAP significantly reduces the number of Gaussians and enforces a strong prior on the scene. As depicted in Fig. 6c, the opacity distribution for the garden scene changes significantly before and after pruning. In the baseline 3DGS, most opacity values are very low, indicating minimal contribution to scene reconstruction. However, after GAP, a substantial proportion of opacity values increase significantly, indicating that the model learns a more solid geometry.

#### 5. Discussion

ELMGS is a flexible, plug-and-play method that can be applied to any 3DGS-based approach. With the widespread use of 3DGS in applications like inverse rendering [18, 31] and animatable avatars [22], various adaptations have been developed [11]. Some methods, such as Gaussian Shader [18] and GIR [31], introduce additional attributes or enhance the baseline algorithm, as demonstrated by Mini-Splatting [10]. Other approaches, including Radsplat [29]

and GART [22], make structural changes to the baseline 3DGS. ELMGS can be seamlessly integrated with any of these 3DGS variants to improve rendering speed and achieve higher compression ratios. Given the widespread use and versatility of 3DGS-based methods, our approach is effectively applicable across various implementations, owing to its plug-and-play nature and high performance.

## 6. Limitations

ELMGS offers significant improvements in rendering speeds and achieves high compression gains while maintaining performance comparable to the baseline. However, one notable limitation is the extreme pruning, which allows us to significantly reduce the number of Gaussians only up to a certain threshold before performance starts to deteriorate. Additionally, our method requires training on top of a pre-trained 3DGS model, which increases the overall training cost. Future work will aim to enhance pruning ratios and reduce training times to address these limitations.

#### 7. Conclusion

We present ELMGS, a novel plug-and-play compression pipeline for 3DGS, incorporating Gradient and Opacity Aware Pruning (GAP), Learned Step size Quantization (LSQ), and entropy encoding with Morton ordering. Our method achieves compression rates of up to  $38 \times$  and enhances rendering speed to over 600 FPS without compromising visual quality on standard benchmark datasets. By substantially reducing the memory and computational footprint of 3DGS and adopting a hardware-friendly quantization approach, ELMGS paves the way for future research to utilize 3DGS on low-power AR/VR and edge devices.

## References

- [1] Adams, F., Qiu, T., Mark, A., Fritz, B., Kramer, L., Schlager, D., Wetterauer, U., Miernik, A., Fischer, P.: Soft 3d-printed phantom of the human kidney with collecting system. Annals of biomedical engineering **45**, 963–972 (2017) 2
- [2] Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded antialiased neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5470–5479 (2022) 2, 4, 6
- [3] Bengio, Y., Léonard, N., Courville, A.: Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432 (2013) 5
- [4] Blalock, D., Gonzalez Ortiz, J.J., Frankle, J., Guttag, J.: What is the state of neural network pruning? Proceedings of machine learning and systems **2**, 129–146 (2020) 5
- [5] Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields. In: European Conference on Computer Vision. pp. 333–350. Springer (2022) 1
- [6] Chen, Y., Wu, Q., Cai, J., Harandi, M., Lin, W.: Hac: Hash-grid assisted context for 3d gaussian splatting compression. arXiv preprint arXiv:2403.14530 (2024)
- [7] Chen, Z., Funkhouser, T., Hedman, P., Tagliasacchi, A.: Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16569–16578 (2023) 2
- [8] Deng, C.L., Tartaglione, E.: Compressing explicit voxel grid representations: fast nerfs become also small. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 1236–1245 (2023) 4
- [9] Esser, S.K., McKinstry, J.L., Bablani, D., Appuswamy, R., Modha, D.S.: Learned step size quantization. In: International Conference on Learning Representations (2020) 5
- [10] Fang, G., Wang, B.: Mini-splatting: Representing scenes with a constrained number of gaussians. arXiv preprint arXiv:2403.14166 (2024) 3, 8
- [11] Fei, B., Xu, J., Zhang, R., Zhou, Q., Yang, W., He, Y.: 3d gaussian as a new vision era: A survey. arXiv preprint arXiv:2402.07181 (2024) 3, 8

- [12] Flynn, J., Neulander, I., Philbin, J., Snavely, N.: Deep-stereo: Learning to predict new views from the world's imagery. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5515–5524 (2016) 2
- [13] Fridovich-Keil, S., Meanti, G., Warburg, F.R., Recht, B., Kanazawa, A.: K-planes: Explicit radiance fields in space, time, and appearance. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12479–12488 (2023) 4
- [14] Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5501–5510 (2022) 2
- [15] Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M.W., Keutzer, K.: A survey of quantization methods for efficient neural network inference. In: Low-Power Computer Vision, pp. 291–326. Chapman and Hall/CRC (2022) 3
- [16] 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) **2**, 6
- [17] Hedman, P., Srinivasan, P.P., Mildenhall, B., Barron, J.T., Debevec, P.: Baking neural radiance fields for real-time view synthesis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5875–5884 (2021) 2
- [18] Jiang, Y., Tu, J., Liu, Y., Gao, X., Long, X., Wang, W., Ma, Y.: Gaussianshader: 3d gaussian splatting with shading functions for reflective surfaces. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5322–5332 (2024) 8
- [19] Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (2023) 2, 3, 5, 6, 7
- [20] 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) 6
- [21] 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) 2, 3, 6, 7, 8

- [22] Lei, J., Wang, Y., Pavlakos, G., Liu, L., Daniilidis, K.: Gart: Gaussian articulated template models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 19876–19887 (2024) 8
- [23] Li, L., Shen, Z., Wang, Z., Shen, L., Bo, L.: Compressing volumetric radiance fields to 1 mb. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4222–4231 (2023) 1
- [24] Liu, Z., Cheng, K.T., Huang, D., Xing, E.P., Shen, Z.: Nonuniform-to-uniform quantization: Towards accurate quantization via generalized straight-through estimation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4942–4952 (2022) 3
- [25] 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) 3
- [26] Mildenhall, B., Srfinivasan, 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) 1, 2
- [27] Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics (ToG) **41**(4), 1–15 (2022) 1, 2, 4
- [28] 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) 2, 3, 6, 7, 8
- [29] Niemeyer, M., Manhardt, F., Rakotosaona, M.J., Oechsle, M., Duckworth, D., Gosula, R., Tateno, K., Bates, J., Kaeser, D., Tombari, F.: Radsplat: Radiance field-informed gaussian splatting for robust real-time rendering with 900+ fps. arXiv preprint arXiv:2403.13806 (2024) 3, 8
- [30] 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) 2, 6, 7, 8

- [31] Shi, Y., Wu, Y., Wu, C., Liu, X., Zhao, C., Feng, H., Liu, J., Zhang, L., Zhang, J., Zhou, B., et al.: Gir: 3d gaussian inverse rendering for relightable scene factorization. arXiv preprint arXiv:2312.05133 (2023) 8
- [32] Sitzmann, V., Thies, J., Heide, F., Nießner, M., Wetzstein, G., Zollhofer, M.: Deepvoxels: Learning persistent 3d feature embeddings. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2437–2446 (2019) 2
- [33] Ullman, S.: The interpretation of structure from motion. Proceedings of the Royal Society of London. Series B. Biological Sciences **203**(1153), 405–426 (1979) 3
- [34] Woodworth, B., Gunasekar, S., Lee, J.D., Moroshko, E., Savarese, P., Golan, I., Soudry, D., Srebro, N.: Kernel and rich regimes in overparametrized models. In: Conference on Learning Theory. pp. 3635–3673. PMLR (2020) 5
- [35] Zhao, T., Chen, J., Leng, C., Cheng, J.: Tinynerf: To-wards 100 x compression of voxel radiance fields. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 3588–3596 (2023) 1
- [36] Ziv, J., Lempel, A.: A universal algorithm for sequential data compression. IEEE Transactions on information theory **23**(3), 337–343 (1977) **5**
- [37] Zwicker, M., Pfister, H., Van Baar, J., Gross, M.: Ewa volume splatting. In: Proceedings Visualization, 2001. VIS'01. pp. 29–538. IEEE (2001) 2