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

Muhammad Salman Ali[1,2], **Sung-Ho Bae**[*2], **Enzo Tartaglione**[*1]
[1]LTCI, Télécom Paris, Institut Polytechnique de Paris, France
[2] Kyung Hee University, Republic of Korea
{salmanali, shbae}@khu.ac.kr,
enzo.tartaglione@telecom-paris.fr

A. Additional Ablation Studies

A.1. Performance comparison on Shiny Blender Dataset

We also conducted a comparative analysis of our approach against the baseline 3DGS using five scenes from the Shiny Blender dataset [38]. As shown in the table, our method achieves a substantial reduction in memory footprint relative to the baseline, while preserving comparable performance, with an average compression gain of approximately 22×.

A.2. Rendering Speed Comparison

We also evaluated the rendering speed of our method using both the Kerb *et al*. rasterization approach and the C3DGS custom renderer. C3DGS achieves fast rendering speeds with its custom renderer, whereas, with the Kerb *et al*. rasterization approach, its rendering speed is comparable to the baseline 3DGS. In contrast, our method significantly enhances rendering speed on both the Kerb *et al*. rasterization approach and the C3DGS renderer. Detailed comparisons are shown in Table 5. For the RTX 3090, we used the Kerb *et al*. rasterization approach, while the AMD and RTX 3080 Laptop GPU evaluations utilized the C3DGS renderer.

A.3. GAP vs Opacity Based Pruning

We also compare our Gradient and Opacity-Aware Pruning (GAP) method with an opacity-based pruning approach that excludes gradient information from (eq. 3). At small pruning thresholds, opacity-based pruning demonstrates similar performance to GAP. However, as the compression rate increases, the performance difference becomes more significant, as shown in Fig. 7. The inclusion of gradient information enhances pruning performance because certain scene features (e.g., sky, glass) may have low opacity but are still essential for accurate scene rendering.

A.4. One Shot Pruning vs Iterative Pruning

We also investigated the impact of one-shot pruning compared to iterative pruning methods. Both GAP and opacity-based pruning (Sec A.3) are iterative pruning methods. Our findings reveal that gradual pruning allows the model to adapt more effectively to the scene compared to one-shot pruning. For one-shot pruning, we utilized the pre-trained 3DGS-30k model, conducted one-shot pruning, and subsequently fine-tuned the model for 30k iterations. The Table 6 provides a detailed comparison of one-shot pruning versus iterative pruning on the Tanks&Temples dataset, while the Fig. 7 illustrates the compression performance tradeoff. It is important to note that these results do not include QAT and entropy encoding.

A.5. Effect of Quantization Aware Training (QAT) and Pruning Order on 3DGS Compression

We examined the efficacy of our proposed method by reversing the order of Quantization Aware Training (QAT) and GAP. Contrary to our standard approach, we applied QAT first to a pre-trained Gaussian model followed by GAP. Notably, we observed a decline in performance compared to our standard approach. In our standard pipeline, the performance of 3DGS improves due to pruning but deteriorates slightly with QAT. Applying QAT first resulted in a performance drop due to quantization, as evidenced by the final results. Detailed outcomes are presented in Table 7.

A.6. Reducing memory footprint of training 3DGS-30k

To reduce the memory footprint of baseline 3DGS, we can also introduce GAP during the training of baseline 3DGS. We present the results in Fig. 8. Our GAP technique yields substantial reductions in the memory footprint if in-

*Corresponding Authors

tegrated during the training of 3DGS. On Tanks&Temples, we observe reductions of up to 24× and 39× in memory footprint, with a respective PSNR drop of 0.1 and 0.3.

B. Why Integer Quantization is Preferred

Vector quantization (VQ) schemes tend to be inefficient on conventional hardware platforms, such as CPUs and GPUs, and even more so on resource-constrained devices like mobile CPUs and IoT devices. The primary issue arises from the high encoding complexity of VQ. Specifically, VQ exhibits exponential encoding complexity due to the need for an extensive nearest-neighbor search. The codebook size used in VQ is typically 2^b, where b represents the quantization precision. Consequently, encoding a matrix using VQ involves 2^b sequential comparisons, resulting in an encoding complexity of $O(2^b)$. In contrast, integer quantization offers significantly lower encoding complexity. Encoding in this scheme can be achieved using only b bitshift operations and a truncation operation, leading to a linear encoding complexity of $O(b+1)$. This makes integer quantization much more favorable for hardware implementations, especially on resource-constrained devices, as it avoids the overhead of VQ's exhaustive search process.

C. 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.

D. Additional Qualitative Comparison

D.1. Visualization at Different Pruning Levels

We provide visualizations of the `bicycle` scene image, which demands significant memory resources: 1.4GB for 3DGS-30K and 778MB for 3DGS-7K. In Fig. 9, we compare the visualizations of the test set image at different pruning levels denoted by γ_{iter}. Our end-to-end compression pipeline demonstrates substantial compression rates while preserving comparable visual quality. For instance, with $\gamma_{iter} = 0.375$, our model achieves a compression ratio of approximately 14× and increases rendering speed by 2.5× while maintaining similar visual quality to 3DGS-30K. Furthermore, with $\gamma_{iter} = 0.6$, our method attains a compression ratio of about 80×, preserving visual quality akin to 3DGS-7K and increasing rendering speed by 4.5×.

D.2. Additional Visualizations

We provide additional visualizations on benchmark datasets Tanks&Temples, Deep Blending, and selected scenes from the Mip-NeRF360 data. We compare our method variants, namely ours-big, ours-medium, and ours-small, with 3DGS-30k and 3DGS-7k baselines, respectively. Our proposed method, ELMGS, demonstrates performance comparable to the original Gaussian counterparts while achieving significant compression rates. Please refer to this link for additional visualizations.

Table 4. Comparison of performance between 3DGS-30k and ELMGS on five scenes from the Shiny Blender dataset.

Scene	3DGS-30k				ELMGS			
	SSIM$^{\uparrow}$	**PSNR**$^{\uparrow}$	**LPIPS**$^{\downarrow}$	**Mem**$^{\downarrow}$	**SSIM**$^{\uparrow}$	**PSNR**$^{\uparrow}$	**LPIPS**$^{\downarrow}$	**Mem**$^{\downarrow}$
car	0.925	27.176	0.050	63.0MB	0.925	27.216	0.052	4.4MB
coffee	0.967	31.982	0.083	35.0MB	0.968	31.883	0.085	2.7MB
helmet	0.945	27.942	0.086	35.0MB	0.946	28.039	0.084	1.6MB
teapot	0.996	43.017	0.009	17.0MB	0.996	43.621	0.009	0.5MB
toaster	0.884	20.978	0.133	83.0MB	0.871	21.060	0.151	3.6MB
Average	0.944	30.219	0.072	46.6MB	0.941	30.364	0.076	2.6MB

Table 5. FPS Comparison between C3DGS and Our method.

	Method	**RTX 3090**	**AMD Ryzen S 2440**	**RTX 3080 Mobile**
Bicycle	C3DGS-Compressed	80	194	210
	Ours	221	202	463
Bonsai	C3DGS-Compressed	153	192	265
	Ours	378	231	417

Table 6. Performance comparison between one-shot pruning, opacity based pruning and gradient and opacity aware (GAP) iterative pruning on Tanks&Temples dataset.

One Shot Pruning				Opacity based Iterative Pruning				Gradient and Opacity Aware Pruning (GAP)			
SSIM$^{\uparrow}$	**PSNR**$^{\uparrow}$	**LPIPS**$^{\downarrow}$	**Mem**$^{\downarrow}$	**SSIM**$^{\uparrow}$	**PSNR**$^{\uparrow}$	**LPIPS**$^{\downarrow}$	**Mem**$^{\downarrow}$	**SSIM**$^{\uparrow}$	**PSNR**$^{\uparrow}$	**LPIPS**$^{\downarrow}$	**Mem**$^{\downarrow}$
0.848	23.920	0.171	326.8MB	0.849	23.981	0.169	338.5MB	0.849	23.989	0.170	335.5MB
0.848	23.901	0.172	261.5MB	0.849	23.939	0.171	261.0MB	0.849	23.971	0.171	280.8MB
0.844	23.813	0.180	196.3MB	0.846	23.961	0.178	200.0MB	0.848	23.966	0.175	219.3MB
0.839	23.698	0.191	152.8MB	0.839	23.799	0.192	152.0MB	0.844	23.936	0.187	153.0MB
0.819	23.376	0.228	87.5MB	0.818	23.404	0.232	86.0MB	0.829	23.848	0.220	76.8MB
0.793	22.927	0.272	44.0MB	0.794	23.069	0.574	47.0MB	0.810	23.559	0.251	45.3MB
0.756	22.234	0.322	22.0MB	0.767	22.655	0.310	25.0MB	0.780	22.919	0.294	23.4MB

Table 7. Comparison of performance between Quantization Aware Training (QAT) before and after Gradient and Opacity aware Pruning (GAP) on Tanks&Temples dataset.

QAT Before GAP				QAT After GAP			
SSIM$^{\uparrow}$	**PSNR**$^{\uparrow}$	**LPIPS**$^{\downarrow}$	**Mem**$^{\downarrow}$	**SSIM**$^{\uparrow}$	**PSNR**$^{\uparrow}$	**LPIPS**$^{\downarrow}$	**Mem**$^{\downarrow}$
0.837	23.629	0.191	51.0MB	0.846	24.000	0.181	46.5MB
0.828	23.391	0.205	39.0MB	0.842	23.967	0.192	35.0MB
0.798	22.907	0.253	23.5MB	0.834	23.946	0.210	23.5MB
0.750	21.994	0.316	13.4MB	0.820	23.769	0.236	14.2MB
0.707	21.163	0.367	7.9MB	0.796	23.309	0.271	8.0MB
0.696	20.902	0.380	5.8MB	0.760	22.549	0.321	3.6MB

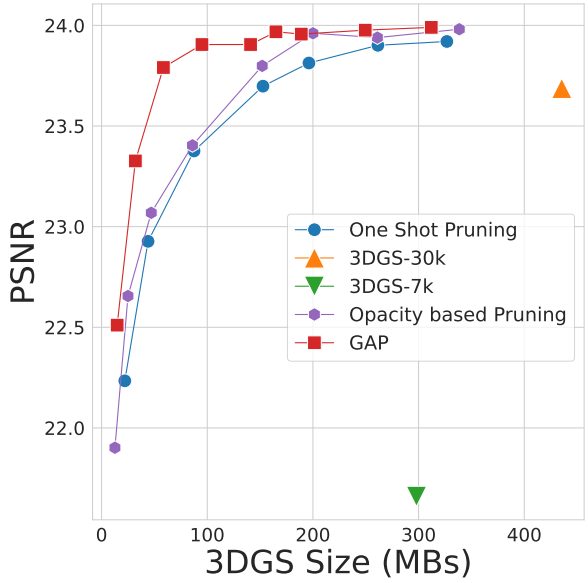

Figure 7. The graph illustrates the trade-off between performance and size when employing GAP, opacity based pruning and one-shot pruning techniques on Tanks&Temple dataset.

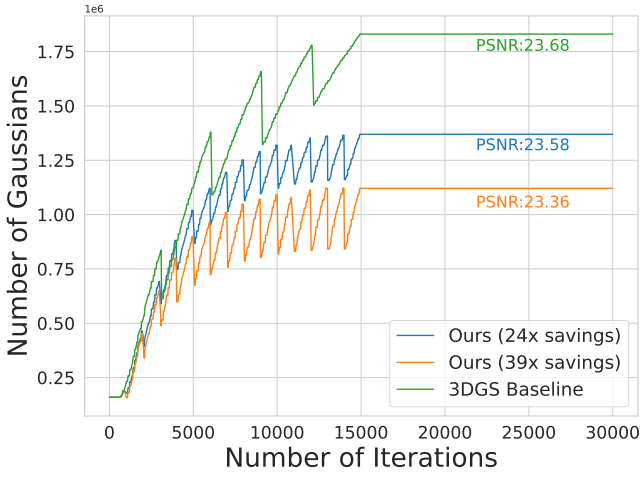

Figure 8. Memory savings while training using Gradient and Opacity based Pruning (GAP) on Tanks&Temples dataset.

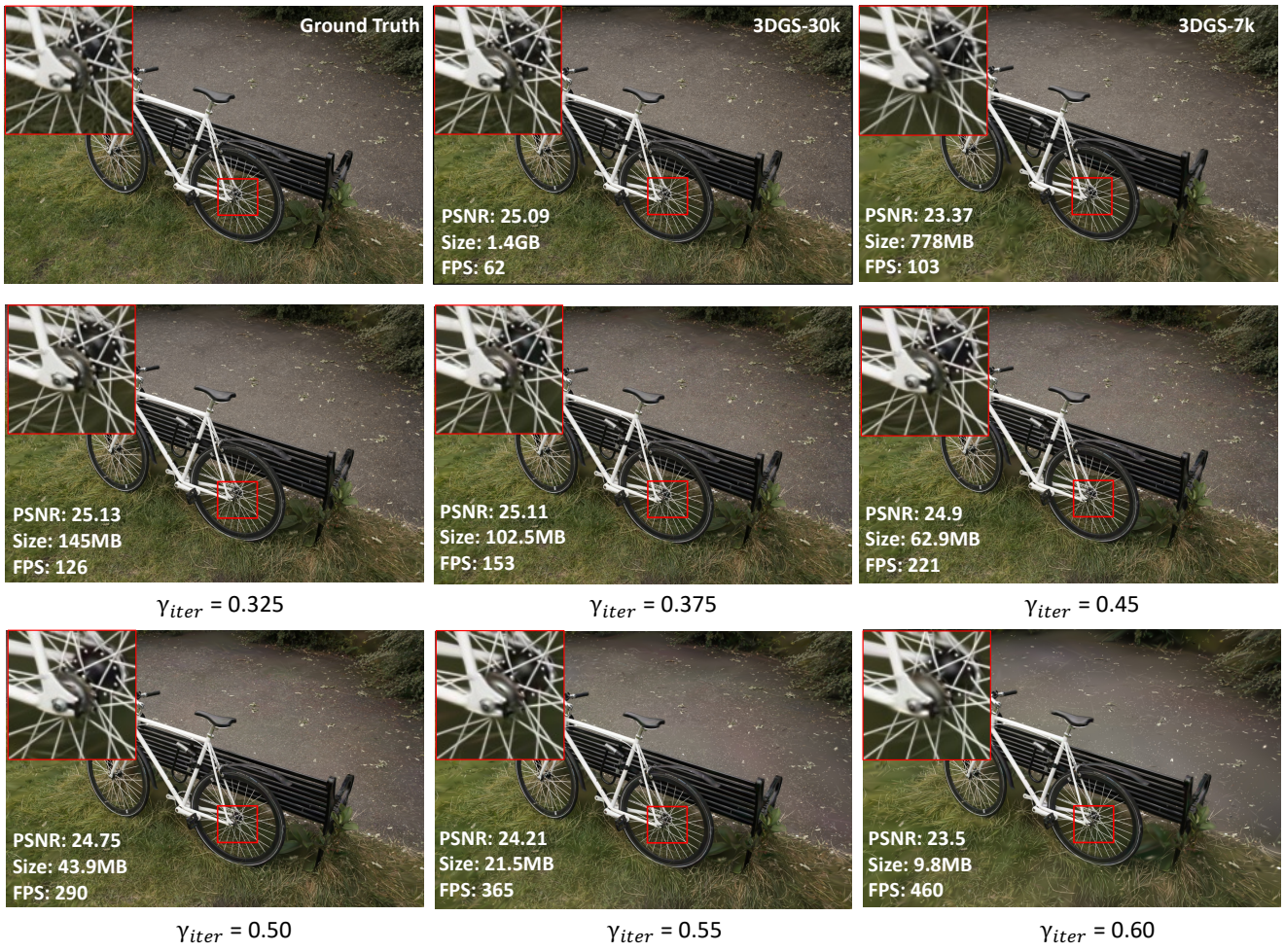

Figure 9. Qualitative comparison of the bicycle image at various pruning levels, defined by γ_{iter}. ELMGS demonstrates substantially higher compression rates compared to both baselines while maintaining similar visual quality.

References

[38] Verbin, D., Hedman, P., Mildenhall, B., Zickler, T., Barron, J.T., Srinivasan, P.P.: Ref-nerf: Structured view-dependent appearance for neural radiance fields. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5481–5490. IEEE (2022) 1