PUP 3D-GS: Principled Uncertainty Pruning for 3D Gaussian Splatting

Alex Hanson* Allen Tu* Vasu Singla Mayuka Jayawardhana
Matthias Zwicker Tom Goldstein

University of Maryland, College Park

Figure 1. We prune the 3D Gaussian Splatting (3D-GS) reconstruction of the Deep Blending *playroom* scene from 2.65M Gaussians to 0.265M Gaussians using our PUP 3D-GS pipeline, accelerating rendering speed from 76.65 FPS to 318.06 FPS – a $\mathbf{4.15\times}$ **speed-up** on this scene– while preserving fine details. In comparison, LightGaussian, a recent high-performing post-hoc pruning pipeline for pretrained 3D-GS models, loses substantially more fine details than PUP 3D-GS and achieves a slower rendering speed of 261.27 FPS.

Abstract

Recent advances in novel view synthesis have enabled real-time rendering speeds with high reconstruction accuracy. 3D Gaussian Splatting (3D-GS), a foundational point-based parametric 3D scene representation, models scenes as large sets of 3D Gaussians. However, complex scenes can consist of millions of Gaussians, resulting in high storage and memory requirements that limit the viability of 3D-GS on devices with limited resources. Current techniques for compressing these pretrained models by pruning Gaussians rely on combining heuristics to determine which Gaussians to remove. At high compression ratios, these pruned scenes suffer from heavy degradation of visual fidelity and loss of foreground details. In this paper, we propose a principled sensitivity pruning score that preserves visual fidelity and foreground details at significantly higher compression ratios than existing approaches. It is computed as a second-order approximation of the reconstruction error on the training views with respect to the spatial parameters of each Gaussian. Additionally, we propose a multi-round prune-refine pipeline that can be applied to any pretrained 3D-GS model without changing its training pipeline. After pruning 90% of Gaussians, a substantially higher percentage than previous methods, our PUP 3D-GS pipeline increases average rendering speed by $3.56\times$ while retaining more salient foreground information and achieving higher image quality metrics than existing techniques on scenes from the Mip-NeRF 360, Tanks & Temples, and Deep Blending datasets.

1. Introduction

Novel view synthesis aims to render views from new viewpoints given a set of 2D images. Neural Radiance Fields (NeRFs) [17] use volume rendering to represent the 3D scene using a multilayer perceptron that can be used to render novel views. Although NeRF and its variants achieve high-quality reconstructions, they suffer from slow inference and require several seconds to render a single image.

*Authors contributed equally. Correspondence to hanson@cs.umd.edu and atu1@umd.edu.

3D Gaussian Splatting (3D-GS) [9] has recently emerged as a faster alternative to NeRF, achieving real-time rendering on modern GPUs and comparable image quality. It represents scenes using a large set of 3D Gaussians with independent location, shape, and appearance parameters. Since they typically consist of millions of Gaussians, 3D-GS scenes often have high storage and memory requirements that limit their viability on devices with limited resources.

Several recent works propose techniques that reduce the storage requirements of 3D-GS models, including heuristics for pruning Gaussians that have an insignificant contribution to the rendered images [3, 4, 12, 14, 20]. In this work, we propose a more mathematically principled approach for deciding which Gaussians to prune from a 3D-GS scene. We introduce a computationally feasible sensitivity score that is derived from the Hessian of the reconstructed error on the training images in a converged scene. Following the methodology in LightGaussian [3], we prune the scene using our sensitivity score and then fine-tune the remaining Gaussians. Our experiments demonstrate that this process can remove 80% of the Gaussians in the base 3D-GS scene while preserving image quality metrics.

Moreover, our approach enables a second consecutive round of pruning and fine-tuning. As shown in Figure 1, our multi-round pruning approach can remove 90% of the Gaussians in the base 3D-GS scene while surpassing previous heuristics-based methods like LightGaussian in both rendering speed and image quality. Our work is orthogonal to several other methods that modify the training framework of 3D-GS or apply quantization-based techniques to reduce its disk storage. We show that our PUP 3D-GS pipeline can be used in conjunction with these techniques to further improve performance and compress the model.

In summary, we propose the following contributions:

1. A post-hoc pruning technique, PUP 3D-GS, that can be applied to any pretrained 3D-GS model without changing its training pipeline.
2. A novel spatial sensitivity score that is more effective at pruning Gaussians than heuristics-based approaches.
3. A multi-round prune-refine approach that produces higher fidelity reconstructions than an equivalent single round of prune-refining.

2. Related work

Neural Radiance Field (NeRF) based [17] approaches use neural networks to represent scenes and perform novel-view synthesis. They produce remarkable visual fidelity but suffer from slow training and inference. Recently, 3D Gaussian Splatting (3D-GS) [9] has emerged as an effective alternative for novel view synthesis, achieving faster inference speed and comparable visual fidelity to state-of-the-art NeRF-based approaches [1].

2.1. Uncertainty Estimation

Several works estimate uncertainty in NeRF-based approaches that arise from sources like transient objects, camera model differences, and lightning changes [8, 21, 22]. Other works focus on uncertainty caused by occlusion or sparsity of training views [16]. These approaches rely on an ensemble of models [26] or variational inference and KL Divergence [24, 25], requiring intricate changes to the training pipeline to model uncertainty.

BayesRays [5] uses Fisher information for post-hoc uncertainty quantification in NeRF-based approaches. Since NeRFs [16] represent the scene as a continuous 3D function, they compute the Hessian over a hypothetical perturbation field to estimate uncertainty. In contrast, our approach focuses on 3D Gaussian Splats [9] instead, and we directly compute the Fisher information using gradient information without relying on a hypothetical perturbation field.

FisherRF [7] also computes Fisher information for 3D Gaussian Splats; however, it only approximates the *diagonal* of the Fisher matrix and uses the color parameters (DC color and spherical harmonic coefficients) of the Gaussians for post-hoc uncertainty estimation. Our approach uses the spatial mean and scaling parameters to compute a more accurate *block-wise* approximation of the Fisher instead (see Section 4.1). Lastly, our work uses uncertainty estimates to prune Gaussians from the model, whereas FisherRF applies their method to perform active-view selection.

2.2. Pruning Gaussian Splat Models

While 3D-GS [9] demonstrates remarkable performance, it also entails substantial storage requirements. Several recent works use codebooks to quantize and reduce storage for various Gaussian parameters [12, 19, 20]. Others use the spatial relationships between neighboring Gaussians to reduce the number of parameters [2, 14, 15, 18]. Although these methods tout high compression rates, they modify the underlying primitives and training framework of 3D-GS. They also do not necessarily reduce the number of primitives and are, therefore, orthogonal to our work. We apply one such technique, Vectree Quantization [3], to further compress our pruned scenes in Section 5.3.3.

A recent pruning-based method, Compact-3DGS [12], proposes a learnable masking strategy to prune small, transparent Gaussians during training. EAGLES [4], which we apply our pipeline to in Appendix A.5, prunes Gaussians based on the least total transmittance per Gaussian. They also begin with low-resolution images, progressively increasing image resolution to reduce Gaussian densification during training, then quantize several attributes to reduce disk storage. LightGaussian [3] computes a global significance score for each Gaussian with heuristics, uses that score to prune the least significant Gaussians, and finally uses quantization to further reduce storage requirements.

3. Background: 3D Gaussian Splatting

3D Gaussian Splatting (3D-GS) is a point-based novel view synthesis technique that uses 3D Gaussians to model the scene. The attributes of the Gaussians are optimized over input training views given by a set of camera poses $\mathcal{P}_{gt} = \{\phi_i \in \mathbb{R}^{3\times4}\}_{i=1}^{K}$ and corresponding ground truth training images $\mathcal{I}_{gt} = \{\boldsymbol{I}_i \in \mathbb{R}^{H\times W}\}_{i=1}^{K}$. A sparse point cloud of the scene generated by Structure from Motion (SfM) over the training views is used to initialize the Gaussians. At fixed intervals during training, a Gaussian densification step is applied to increase the number of Gaussians in areas of the model where small-scale geometry is insufficiently reconstructed.

Each 3D Gaussian $\mathcal{G}_i$ is independently parameterized by position $x_i \in \mathbb{R}^3$, scaling $s_i \in \mathbb{R}^3$, rotation $r_i \in \mathbb{R}^4$, base color $c_i \in \mathbb{R}^3$, view-dependent spherical harmonics $h_i \in \mathbb{R}^{15\times3}$, and opacity $\alpha_i \in \mathbb{R}$. From these, we define the set of all Gaussian parameters as:

$$\mathcal{G} = \{\mathcal{G}_i = \{x_i, s_i, r_i, c_i, h_i, \alpha_i\}\}_{i=1}^{N}, \tag{1}$$

where N is the total number of Gaussians in the model.

During view synthesis, the scaling parameters s_i and rotation parameters r_i are converted into the scaling and rotation matrices $\boldsymbol{S}_i$ and $\boldsymbol{R}_i$. The Gaussian $\mathcal{G}_i$ is spatially characterized in the 3D scene by its center point, or mean position, x_i and a decomposable covariance matrix $\boldsymbol{\Sigma}_i$:

$$\mathcal{G}_i(x_i) = e^{-\frac{1}{2}x_i^T\boldsymbol{\Sigma}_i^{-1}x_i}, \boldsymbol{\Sigma}_i = \boldsymbol{R}_i\boldsymbol{S}_i\boldsymbol{S}_i^T\boldsymbol{R}_i^T. \tag{2}$$

For a given camera pose ϕ, a differentiable rasterizer renders 2D image $I_{\mathcal{G}}(\phi)$ by projecting all Gaussians observed from ϕ onto the image plane. The color of each pixel p in $I_{\mathcal{G}}(\phi)$ is given by the blending of the $\mathcal{N}$ ordered Gaussians that overlap it:

$$C(p) = \sum_{i\in\mathcal{N}} \tilde{c}_i\tilde{\alpha}_i(p)\prod_{j=1}^{i-1}(1-\tilde{\alpha}_j(p)), \tag{3}$$

where $\tilde{c}_i$ represents the view-dependent color calculated from the camera pose ϕ and the optimizable per-Gaussian color c_i and spherical harmonics h_i, and $\tilde{\alpha}_i(p)$ represents the projected Gaussian value at p times the Gaussian's opacity α_i.

The 3D-GS model is trained by optimizing the loss function L via stochastic gradient descent:

$$L(\mathcal{G}|\phi, I_{gt}) = ||I_{\mathcal{G}}(\phi) - I_{gt}||_1 + L_{SSIM}(I_{\mathcal{G}}(\phi), I_{gt}), \tag{4}$$

where the first term is a L1 residual loss and the second term is a SSIM loss.

4. Method

3D scene reconstruction is an inherently underconstrained problem. Capturing a scene as a set of posed images $(\mathcal{P}_{gt}, \mathcal{I}_{gt})$ involves projecting it onto the 2D image plane of each view. As illustrated by Figure 2, this introduces uncertainty in the locations and sizes of the Gaussians reconstructing the scene: a large Gaussian far from the camera can be equivalently modeled in pixel space by a small Gaussian close to the camera. In other words, Gaussians that are not perceived by a sufficient number of cameras may be able to reconstruct the input view image from a range of locations and scales.

We define **uncertainty** in 3D-GS as the amount that a Gaussian's parameters, such as its location and scale, can be perturbed without affecting the reconstruction loss over the input views. Concretely, this is the **sensitivity** of the error over the input views to that particular Gaussian. Given a loss function $L : \mathbb{R}^{\mathcal{G}} \to \mathbb{R}$ that takes the set of Gaussians $\mathcal{G}$ as inputs and outputs an error value over the set of training views, this sensitivity can be captured by the Hessian $\nabla^2_{\mathcal{G}}L$.

In the following subsections, we demonstrate how to compute this sensitivity and use it to decide which Gaussians to prune from the model. Directly computing the full Hessian matrix is intractable due to memory constraints. We demonstrate how to obtain a close estimate of the Hessian via a Fisher approximation in Section 4.1. We also find that only a block-wise approximation of the Hessian parameters is needed to quantify Gaussian sensitivity in Section 4.2, and that computing this over image patches is sufficient for pruning in Section 4.3. Finally, we find that multiple rounds of pruning and fine-tuning improves our performance over one-shot pruning and fine-tuning, giving our full PUP 3D-GS pipeline in Section 4.4.

4.1. Fisher Information Matrix

To obtain a per-Gaussian sensitivity, we begin by taking the L_2 error over the input reconstruction images $I_{\mathcal{G}}$:

$$L_2 = \frac{1}{2}\sum_{\phi\in\mathcal{P}_{gt}} ||I_{\mathcal{G}}(\phi) - I_{gt}||_2^2. \tag{5}$$

Differentiating this twice gives us the Hessian:

$$\nabla^2_{\mathcal{G}}L_2 = \sum_{\phi\in\mathcal{P}_{gt}} \nabla_{\mathcal{G}}I_{\mathcal{G}}(\phi)\nabla_{\mathcal{G}}I_{\mathcal{G}}(\phi)^T + (I_{\mathcal{G}}(\phi) - I_{gt})\nabla^2_{\mathcal{G}}I_{\mathcal{G}}(\phi). \tag{6}$$

On a converged 3D-GS model, the $||I_{\mathcal{G}} - I_{gt}||_1$ residual term of Equation 4 approaches zero, causing the second order term $(I_{\mathcal{G}}(\phi) - I_{gt})\nabla^2_{\mathcal{G}}I_{\mathcal{G}}$ in Equation 6 to vanish:

$$\nabla^2_{\mathcal{G}}L_2 = \sum_{\phi\in\mathcal{P}_{gt}} \nabla_{\mathcal{G}}I_{\mathcal{G}}(\phi)\nabla_{\mathcal{G}}I_{\mathcal{G}}(\phi)^T. \tag{7}$$

This is identified as the Fisher Information matrix in related literature [5, 7]. We provide an explicit derivation of the Fisher Information matrix from this L_2 loss along with a Bayesian interpretation of our method in Appendix A.1.

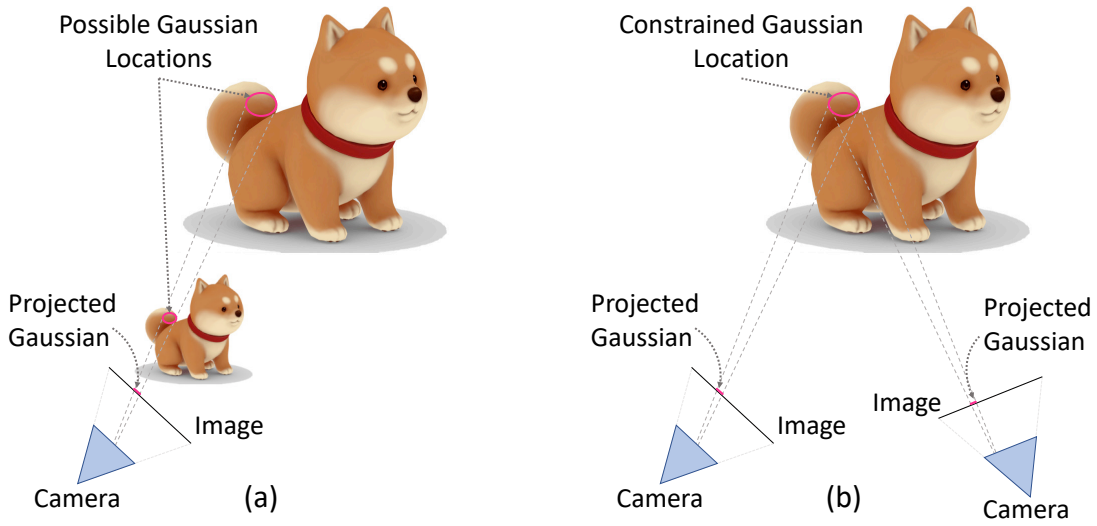

Figure 2. Spatial uncertainty arises from limited views because there are multiple possible 3D Gaussian locations that map to the same projected Gaussian in pixel space (a). This is reduced when multiple cameras observe previously unconstrained Gaussians (b).

Note that $\nabla_{\mathcal{G}} I_{\mathcal{G}}$ is the gradient over only the reconstructed images, so our approximation only depends on the input poses $\mathcal{P}_{gt}$ and not the input images $\mathcal{I}_{gt}$.

4.2. Sensitivity Pruning Score

The full Hessian over the model parameters $\nabla^2_{\mathcal{G}} L \in \mathbb{R}^{\mathcal{G}\times\mathcal{G}}$ is quadratically large. However, we find that using only a subset of these parameters is sufficient for an effective sensitivity pruning score. For brevity, we remove the $\mathcal{P}_{gt}$ terms.

To obtain sensitivity scores for each Gaussian $\mathcal{G}_i$, we restrict ourselves to the block diagonal of the Hessian that only captures inter-Gaussian parameter relationships. This allows us to use each block as a per-Gaussian Hessian from which we can obtain an independent sensitivity score:

$$\boldsymbol{H}_i = \nabla_{\mathcal{G}_i} I_{\mathcal{G}} \nabla_{\mathcal{G}_i} I_{\mathcal{G}}^T, \tag{8}$$

where $\nabla_{\mathcal{G}_i}$ is the gradient with respect to the parameters of Gaussian $\mathcal{G}_i$. Intuitively, each block Hessian $\boldsymbol{H}_i$ measures the isolated impact that perturbing only $\mathcal{G}_i$'s parameters would have on the reconstruction error. To turn $\boldsymbol{H}_i$ into a sensitivity score $\tilde{U}_i \in \mathbb{R}$, we take its log determinant:

$$\tilde{U}_i = \log|\boldsymbol{H}_i| = \log|\nabla_{\mathcal{G}_i} I_{\mathcal{G}} \nabla_{\mathcal{G}_i} I_{\mathcal{G}}^T|. \tag{9}$$

This captures the relative impact of all the parameters of Gaussian $\mathcal{G}_i$ on the reconstruction error. Specifically, it is a relative volume measure of the basin around the Gaussian parameters $\mathcal{G}_i$ in the second-order approximation of the function describing their impact on the reconstruction error.

We find that we can restrict the per Gaussian parameters even further and consider only the spatial mean x_i and scaling s_i parameters for an effective sensitivity score. Our final sensitivity pruning score $U_i \in \mathbb{R}$ is:

$$U_i = \log|\nabla_{x_i,s_i} I_{\mathcal{G}} \nabla_{x_i,s_i} I_{\mathcal{G}}^T|. \tag{10}$$

We speculate that only the mean and scaling parameters are needed because they capture the projective geometric invariances that exist between the 3D scene and the input views as shown in Figure 2. Rotations – the remaining geometric parameters – are excluded because they do not induce a change of 3D geometry when invariances are present. We ablate the choice of parameters in Section 6.3.

4.3. Patch-wise Uncertainty

The computation of the Hessian over the entire scene requires a sum over all per-pixel Fisher approximations of the reconstructed input views. However, we empirically observe that sensitivity scores computed over image patches are highly correlated with those computed over individual pixels. Specifically, we compute the Fisher information approximation on image patches by rendering downsampled images and computing the Fisher approximation on each of their pixels. Then, we obtain the scene-level Hessian by summing the patch-wise Fisher approximations over all views. We use 4×4 image patches in our experiments and ablate our choice of patch size in Appendix A.2.

4.4. Multi-Round Pipeline

Similar to LightGaussian [3], we prune the model and then fine-tune it without further Gaussian densification. For brevity, we will refer to this process as **prune-refine**. We find that, in many circumstances, the model is able to recover the small $||I_{\mathcal{G}} - I_{gt}||_1$ residual after fine-tuning, allowing us to repeat prune-refine over multiple rounds. We empirically notice that multiple rounds of prune-refine outperforms an equivalent single round of prune-refine. Details of this evaluation are in Section 6.1.

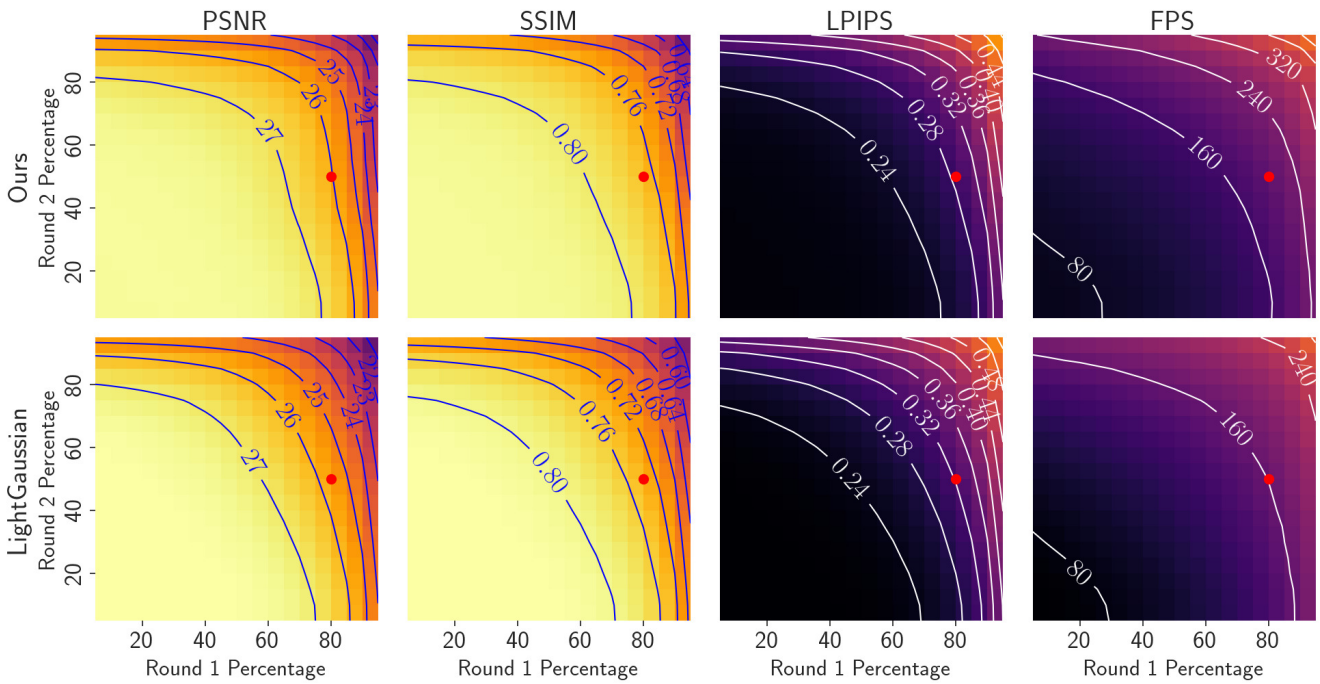

Figure 3. The average metrics over all scenes in the Mip-NeRF 360 dataset after pruning with different percentages in our two-round pipeline. PSNR and SSIM decrease and LPIPS and FPS increase with more pruning. Per-round percentages are selected in 5% intervals and the model is fine-tuned for $5,000$ iterations in each round. The red dots at $(80\%, 50\%)$ represent our percentages for 90% compression.

5. Experiments

5.1. Datasets

We evaluate our approach on the same challenging, real world scenes as 3D-GS [9]. All nine scenes from the Mip-NeRF 360 dataset [1], which consists of five outdoor and four indoor scenes that each contain a complex central object or viewing area and a detailed background, are used. Two outdoor scenes, *truck* and *train*, are taken from the Tanks & Temples dataset [11], and two indoor scenes, *dr-johnson* and *playroom*, are taken from the Deep Blending dataset [6]. For consistency, we use the COLMAP [23] camera pose estimates that were provided by the creators of the datasets.

5.2. Implementation Details

Our method, PUP 3D-GS, can be applied to any 3D-GS model. We implement the Hessian computation as a CUDA kernel in the original 3D-GS codebase [9], then adapt the pruning and refining framework from LightGaussian [3] to accommodate our uncertainty estimate and multi-round pruning method. For a fair comparison, we run LightGaussian and our pipeline on the same pretrained 3D-GS scenes. Rendering speeds are collected using a Nvidia RTXA4000 GPU in frames per second (FPS).

5.3. Results

5.3.1. Analysis of PUP 3D-GS Pipeline

The efficacy of our PUP 3D-GS pipeline is demonstrated by Table 1, where we record the mean of each metric over the Mip-NeRF 360 dataset. Pruning 80% of Gaussians from the original 3D-GS model more than doubles its FPS while degrading PSNR, SSIM, and LPIPS. Fine-tuning the pruned model for 5,000 iterations recovers the image quality metrics with only a slight decrease in rendering speed. Pruning 50% of the remaining Gaussians, such that a total of 90% of Gaussians are pruned from the base 3D-GS scene, causes another drop in the image quality metrics. However, we recover most of this degradation by fine-tuning this significantly smaller model for another 5,000 iterations. Rendering speed is more than tripled over the 3D-GS model.

Table 1. Mean PSNR, SSIM, LPIPS, FPS, and point cloud size for the Mip-NeRF 360 dataset at each stage of our PUP 3D-GS pipeline. The operation in each row is applied **cumulatively** to all of the following rows.

Methods	PSNR↑	SSIM↑	LPIPS↓	FPS↑	Size (MB)↓
3D-GS	27.47	0.8123	0.2216	63.88	746.46
+ Prune 80%	21.00	0.7075	0.3075	167.93	149.29
+ Refine	26.97	0.7991	0.2444	148.28	149.29
+ Prune 50%	22.63	0.7021	0.3316	241.97	74.65
+ Refine	26.67	0.7862	0.2719	204.81	74.65

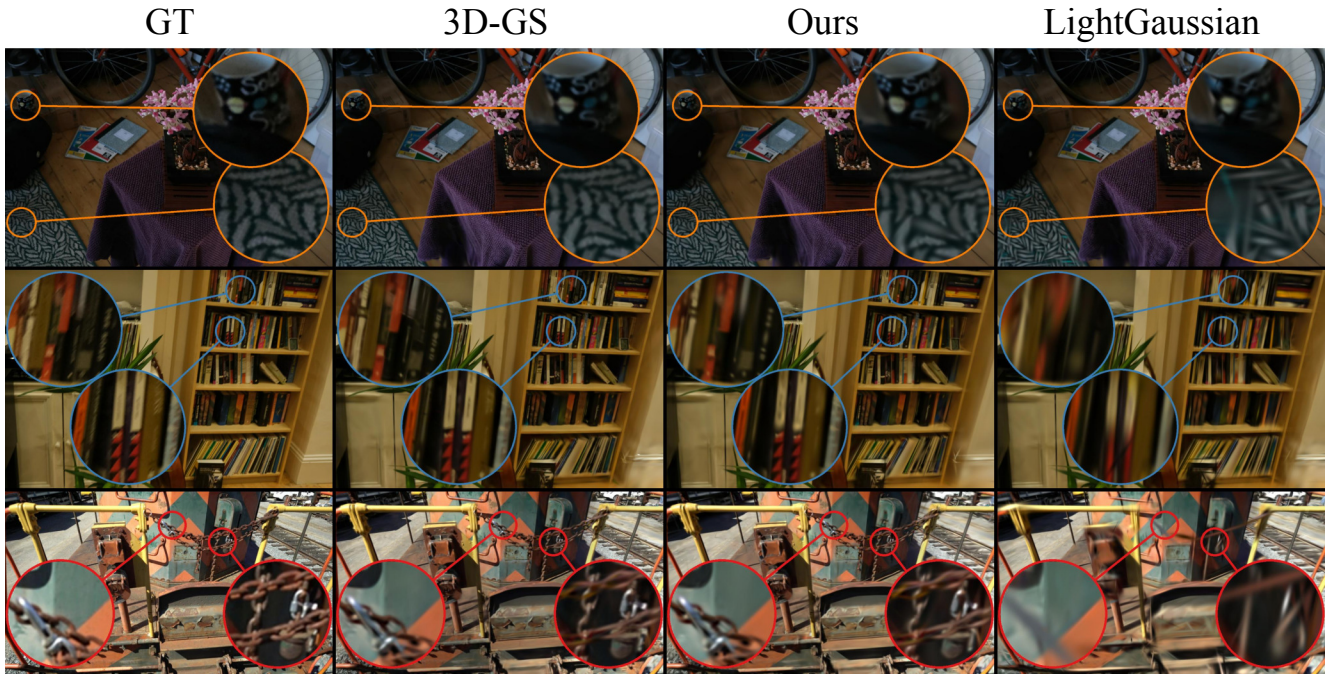

Figure 4. Visual comparison after two rounds of prune-refine using our and LightGaussian's methods. Top: *bonsai* from Mip-NeRF 360. Middle: *room* from Mip-NeRF 360. Bottom: *train* from Tanks & Temples. Additional visualizations are presented in Appendix A.6.

5.3.2. Comparison with LightGaussian

Figure 3 compares our spatial uncertainty estimate against LightGaussian's global significance score at different per-round pruning percentages in our two-round prune refine pipeline. After each round of pruning, the models are fine-tuned for $5,000$ iterations for a total of $10,000$ iterations. The per-dataset means of each metric are illustrated as heatmaps. As illustrated by the contour lines, our method consistently outperforms LightGaussian across all metrics and pruning percentage permutations.

We choose to prune 80% then 50% of the Gaussians in the first and second rounds of pruning to optimize image quality and rendering speed at 90% pruning, a significantly higher compression ratio than previous methods. Table 2 compares our 90% pruning results against LightGaussian's using the per-dataset mean of each metric. We show that our spatial uncertainty estimate outperforms LightGaussian across nearly every metric – the lone exception is PSNR in the Tanks & Temples dataset, where we are outperformed by LightGaussian by 0.36 points. Our choice of pruning percentages is further ablated in Section 6.2.

In Figure 4, we report qualitative results on one scene taken from each dataset. The magnified regions demonstrate that our method retains many fine foreground details that are not preserved by LightGaussian. Error residual visualizations with respect to the ground truth images and original 3D-GS renders are available in Appendix A.7; visualizations of other scenes are provided in Appendix A.6.

Table 2. Comparison of our PUP 3D-GS pipeline against LightGaussian [3]. In both methods, we prune-refine 80% of the Gaussians from the 3D-GS model and then 50% of the remaining Gaussians for a total of 90%. The final sizes are identical. PUP 3D-GS increases the rendering speed of 3D-GS by $3.56\times$ on average. Per-scene metrics for each dataset are recorded in Appendix A.4.

Datasets	Methods	PSNR↑	SSIM↑	LPIPS↓	FPS↑	Size (MB)↓
MipNeRF-360	3D-GS	27.47	0.8123	0.2216	64.07	746.46
	LightGaussian	26.28	0.7622	0.3054	162.12	74.65
	Ours	**26.67**	**0.7862**	**0.2719**	**204.81**	74.65
Tanks & Temples	3D-GS	23.77	0.8458	0.1777	97.86	433.24
	LightGaussian	**23.08**	0.7950	0.2634	329.03	43.33
	Ours	22.72	**0.8013**	**0.2441**	**391.10**	43.33
Deep Blending	3D-GS	28.98	0.8816	0.2859	66.79	699.19
	LightGaussian	28.51	0.8675	0.3292	234.10	69.92
	Ours	**28.85**	**0.8810**	**0.3015**	**301.43**	69.92

We speculate that the visibility score in LightGaussian's importance sampling heuristic is biased towards retaining larger Gaussian because they have a higher probability of being visible in more pixels across the training views. In Figure 5, we plot the distributions of the log determinants of the Gaussian covariances over the scenes visualized in Figure 4. The distributions skew more heavily towards larger Gaussians in the scenes pruned with LightGaussian's pipeline than ours.

5.3.3. Vectree Quantization

We apply Vectree Quantization – a compression method that is orthogonal to pruning – from the LightGaussian framework [3] to further compress our pruned models. Our final

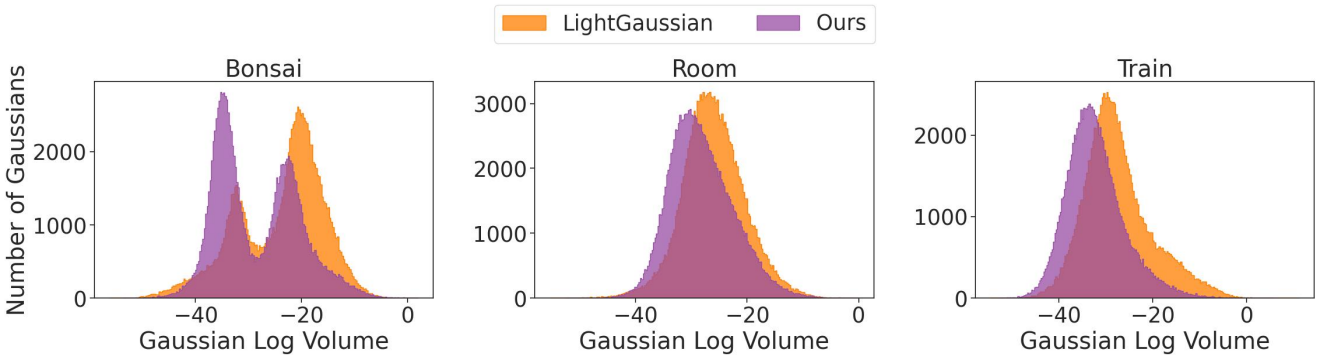

Figure 5. Histograms of the distribution of Gaussians over the log of their volumes for the *bonsai*, *room*, and *train* scenes after two rounds of prune-refine. PUP 3D-GS retains smaller Gaussians than LightGaussian, consistent with its higher rendering speed and visual fidelity.

models are smaller than the unpruned 3D-GS scenes by an average of $51.70\times$, $52.56\times$, and $51.06\times$ for the Mip-NeRF 360, Tanks & Temples, and Deep Blending datasets. Detailed results are in Table 3. We also apply our method to EAGLES [4], another orthogonal method, in Appendix A.5.

Table 3. Vectree Quantization comparison with LightGaussian after pruning to 90%. PUP 3D-GS outperforms LightGaussian in terms of image quality, rendering speed, and now size.

Datasets	Methods	PSNR↑	SSIM↑	LPIPS↓	FPS↑	Size (MB)↓
Mip-NeRF 360	3D-GS	27.47	0.8123	0.2216	64.07	746.46
	LightGaussian	24.65	0.7302	0.3341	162.34	14.44
	Ours	**24.93**	**0.7584**	**0.2988**	**205.97**	**14.44**
Tanks & Temples	3D-GS	23.77	0.8458	0.1777	97.86	433.24
	LightGaussian	**21.88**	0.7679	0.2886	331.47	8.53
	Ours	21.61	**0.7787**	**0.2670**	**389.18**	**8.49**
Deep Blending	3D-GS	28.98	0.8816	0.2859	66.79	699.19
	LightGaussian	27.90	0.8586	0.3403	233.28	13.38
	Ours	**28.24**	**0.8735**	**0.3108**	**300.15**	**13.30**

6. Ablations

6.1. Multi-Round Pruning

A core component of our pipeline is multiple rounds of prune-refine. In Table 4, we compare the mean image quality metrics and FPS over the Mip-NeRF 360 scenes when we prune using our two step prune-refine method against pruning 90% of Gaussians in a single round with an equivalent number of fine-tuning steps. Two-round pruning produces better measurements across all metrics.

The number of pruning percentage permutations increases exponentially as the number of rounds increases, so we use interpolation to compare two against three rounds of pruning on the Mip-NeRF 360 *bicycle* scene in Figure 6. Three-round pruning is substantially worse at lower percentages and only slightly better at higher percentages. We choose the simpler two-round approach for our pipeline because both configurations outperform LightGaussian by a much larger margin at 90%.

Table 4. One step of pruning 90% of Gaussians from 3D-GS and then fine-tuning for 10,000 iterations, versus pruning 80% then 50% of Gaussians and fine-tuning for 5,000 iterations in each step. In both methods, 90% of Gaussians are pruned cumulatively and the model is fine-tuned for 10,000 total steps. Our multi-round approach outperforms the one-round approach across all metrics.

Methods	Mip-NeRF 360				
	PSNR↑	SSIM↑	LPIPS↓	FPS↑	Size (MB)↓
3D-GS	27.47	0.8123	0.2216	95.59	746.46
Prune 90% + 10K Fine-tune	26.12	0.7761	0.2807	189.95	74.65
Prune 80% + 50% (5K Each)	**26.67**	**0.7862**	**0.2719**	**204.81**	74.65

6.2. Per-Round Pruning Percentages

We choose to prune 90% of Gaussians, substantially more than previous methods, because preserving visual fidelity at extreme compression ratios is challenging. Figure 7 plots the metrics for each permutation of per-round pruning percentages that results in approximately 90% total pruning. Pruning 80% then 50% of the Gaussians in the first and second rounds optimizes image quality and rendering speed at exactly 90% total pruning. Our method outperforms LightGaussian across all metrics and percentage permutations.

6.3. Spatial vs. Color Parameters

Another component of our method is restricting our sensitivity pruning score to only the mean and scale parameters of each Gaussian. We choose to exclude rotations, the remaining geometric parameters, because they do not induce a change in 3D geometry when invariances are present. Furthermore, including rotation parameters produces 10×10 Fisher Information matrices that are $2.78\times$ larger than the 6×6 mean and scale parameter-only matrices used in our method, inducing significant additional computational overhead. It is no longer possible to run the pruning pipeline on the Nvidia RTXA4000 GPU used for our other experiments due to memory constraints, necessitating more expensive hardware like the Nvidia RTXA5000.

Table 5 shows that image quality decreases if we include

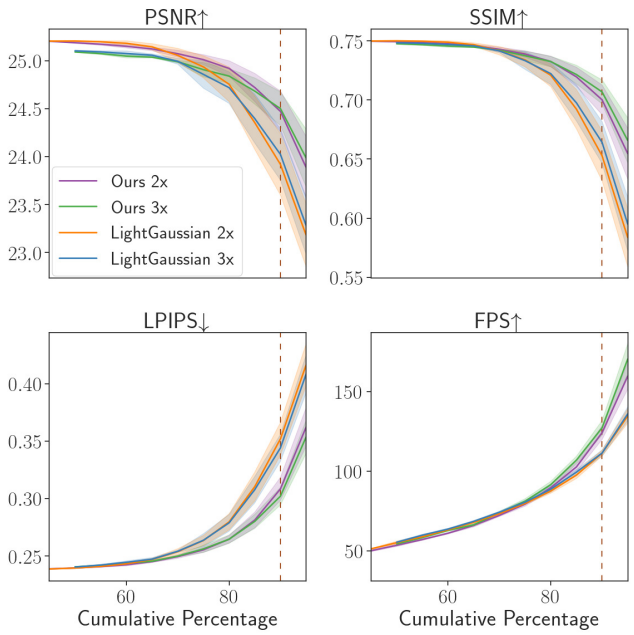

Figure 6. Summary results on the MipNeRF-360 *bicycle* scene for $2\times$ and $3\times$ prune-refine rounds. We plot the mean and standard deviation of $2\times$ for all cumulative pruning percents in Figure 3, computed via cubic interpolation, then do the same for $3\times$ with 10% pruning intervals. The dotted red line denotes our target 90% cumulative percentage; per-round results are ablated in Figure 7.

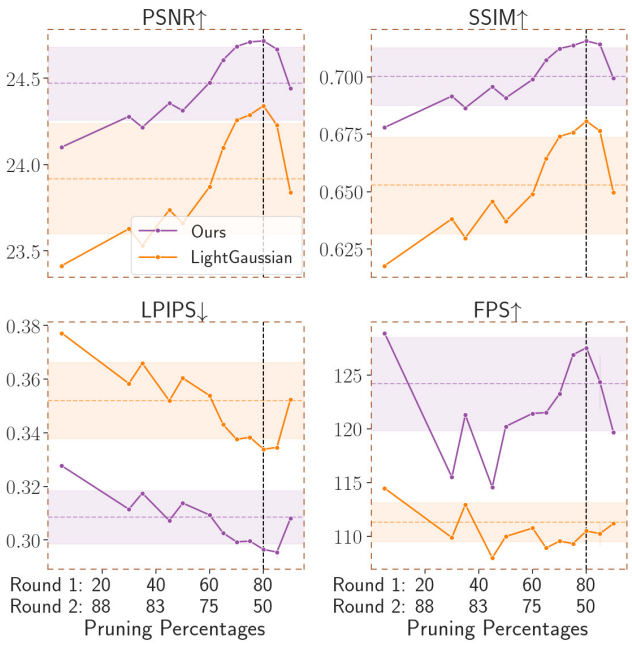

Figure 7. Results for the MipNeRF-360 *bicycle* scene when pruning approximately 90% of Gaussians with varying per-round percentages in our two-round pipeline. The dotted black line denotes our chosen pruning percentages of 80% then 50%; the 90% means from Figure 6 are also shown as dotted lines. Appendix A.3 reports the average metrics across all scenes in the MipNeRF-360 dataset.

the rotation parameters in our mean and scale parameter sensitivity score. Image quality also decreases if we use the RGB color parameters of the Gaussians to compute our sensitivity score instead. We speculate that the lower performance of the RGB sensitivity score is due to its similarity with LightGaussian's visibility score – the change in the L2 error over the training images with respect to perturbations in the RGB value of a Gaussian correlates with its visibility across the training views.

Table 5. Results from our two step prune-refine pipeline when computing sensitivity via our score, our score with rotation parameters, and RGB color parameters. Our spatial score outperforms the spatial with rotation and RGB color scores.

Methods	Mip-NeRF 360				
	PSNR↑	SSIM↑	LPIPS↓	FPS↑	Size (MB)↓
3D-GS	27.47	0.8123	0.2216	95.59	746.46
Ours	26.67	**0.7862**	**0.2719**	**204.81**	74.65
Ours + Rotation	**26.79**	0.7861	0.2726	177.30	74.65
RGB Sensitivity	26.40	0.7798	0.2769	195.76	74.65

7. Limitations

A limitation of our approach is that it assumes that the scene is converged because the Fisher approximation requires a small L1 residual to be accurate. We find that a small residual is preserved even after removing 80% of the Gaussians with prune-refine, allowing for a second round of pruning, as shown in Section 4.4. Another limitation is that the memory requirement for computing the Fisher matrices is proportional to $N \times 36$, where N is the total number of Gaussians. We do not find this size to be prohibitive on Nvidia RTXA4000 or larger GPUs.

8. Conclusion

In this work, we propose PUP 3D-GS: a new post-hoc pipeline for pruning pretrained 3D-GS models without changing their training pipelines. It uses a novel, mathematically principled approach for choosing which Gaussians to prune by assessing their sensitivity to the reconstruction error over the training views. This sensitivity pruning score is derived from a computationally tractable Fisher approximation of the Hessian over that reconstruction error. We use our sensitivity score to prune Gaussians from the reconstructed scene, then fine-tune the remaining Gaussians in the model over multiple rounds. After pruning 90% of Gaussians, PUP 3D-GS increases average rendering speed by $3.56\times$, while retaining more salient foreground information and achieving higher image quality metrics than existing techniques on scenes from the Mip-NeRF 360, Tanks & Temples, and Deep Blending datasets.

9. Acknowledgements

This work was made possible by the IARPA WRIVA Program, the ONR MURI program, and DAPRA TIAMAT. Commercial support was provided by Capital One Bank, the Amazon Research Award program, and Open Philanthropy. Further support was provided by the National Science Foundation (IIS-2212182), and by the NSF TRAILS Institute (2229885). Zwicker was additionally supported by the National Science Foundation (IIS-2126407).

References

[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5470–5479, 2022. 2, 5

[2] Yihang Chen, Qianyi Wu, Jianfei Cai, Mehrtash Harandi, and Weiyao Lin. Hac: Hash-grid assisted context for 3d gaussian splatting compression. *arXiv preprint arXiv:2403.14530*, 2024. 2

[3] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *arXiv preprint arXiv:2311.17245*, 2023. 2, 4, 5, 6

[4] Sharath Girish, Kamal Gupta, and Abhinav Shrivastava. Eagles: Efficient accelerated 3d gaussians with lightweight encodings. *arXiv preprint arXiv:2312.04564*, 2023. 2, 7, 12

[5] Lily Goli, Cody Reading, Silvia Selllán, Alec Jacobson, and Andrea Tagliasacchi. Bayes' rays: Uncertainty quantification for neural radiance fields. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. 2, 3

[6] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics*, 37(6):1–15, 2018. 5

[7] Wen Jiang, Boshu Lei, and Kostas Daniilidis. Fisherrf: Active view selection and uncertainty quantification for radiance fields using fisher information. *arXiv preprint arXiv:2311.17874*, 2023. 2, 3

[8] Liren Jin, Xieyuanli Chen, Julius Rückin, and Marija Popović. Neu-nbv: Next best view planning using uncertainty estimation in image-based neural rendering. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 11305–11312. IEEE, 2023. 2

[9] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4):1–14, 2023. 2, 5, 11

[10] Andreas Kirsch and Yarin Gal. Unifying approaches in active learning and active sampling via fisher information and information-theoretic quantities. *Transactions on Machine Learning Research*, 2022. Expert Certification. 11

[11] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4):1–13, 2017. 5

[12] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian representation for radiance field. *arXiv preprint arXiv:2311.13681*, 2023. 2

[13] Erich L Lehmann and George Casella. *Theory of point estimation*. Springer Science & Business Media, 1998. 11

[14] Xiangrui Liu, Xinju Wu, Pingping Zhang, Shiqi Wang, Zhu Li, and Sam Kwong. Compgs: Efficient 3d scene representation via compressed gaussian splatting. *arXiv preprint arXiv:2404.09458*, 2024. 2

[15] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. *arXiv preprint arXiv:2312.00109*, 2023. 2

[16] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7210–7219, 2021. 2

[17] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 1, 2

[18] Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. Compact 3d scene representation via self-organizing gaussian grids. *arXiv preprint arXiv:2312.13299*, 2023. 2

[19] KL Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. Compact3d: Compressing gaussian splat radiance field models with vector quantization. *arXiv preprint arXiv:2311.18159*, 2023. 2

[20] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3d gaussian splatting for accelerated novel view synthesis. *arXiv preprint arXiv:2401.02436*, 2023. 2

[21] Xuran Pan, Zihang Lai, Shiji Song, and Gao Huang. Activenerf: Learning where to see with uncertainty estimation. In *European Conference on Computer Vision*, pages 230–246. Springer, 2022. 2

[22] Sara Sabour, Suhani Vora, Daniel Duckworth, Ivan Krasin, David J Fleet, and Andrea Tagliasacchi. Robustnerf: Ignoring distractors with robust losses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20626–20636, 2023. 2

[23] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016. 5

[24] Jianxiong Shen, Adria Ruiz, Antonio Agudo, and Francesc Moreno-Noguer. Stochastic neural radiance fields: Quantifying uncertainty in implicit 3d representations. In *2021 International Conference on 3D Vision*, pages 972–981. IEEE, 2021. 2

[25] Jianxiong Shen, Antonio Agudo, Francesc Moreno-Noguer, and Adria Ruiz. Conditional-flow nerf: Accurate 3d modelling with reliable uncertainty quantification. In *European Conference on Computer Vision*, pages 540–557. Springer, 2022. 2

[26] Niko Sünderhauf, Jad Abou-Chakra, and Dimity Miller. Density-aware nerf ensembles: Quantifying predictive uncertainty in neural radiance fields. In *2023 IEEE International Conference on Robotics and Automation*, pages 9370–9376. IEEE, 2023. 2

A. Appendix

A.1. Fisher Information Derivation: A Bayesian Interpretation of PUP 3D-GS

The Fisher Information is the variance of the score:

$$I(\theta) = E_\theta[(\nabla_\theta \log p(x|\theta))^2]. \tag{11}$$

Lemma 5.3 from [13] gives that this is equivalent to:

$$I(\theta) = E_\theta[-\nabla_\theta \nabla_\theta \log p(x|\theta)], \tag{12}$$

assuming $\log p(x|\theta)$ is twice differentiable and with certain regularity conditions.

To start, we reformulate our L_2 objective as a log-likelihood:

$$-\log p(\mathcal{I}|\Phi, \mathcal{G}) = E_{(I,\phi)\sim(\mathcal{I},\Phi)}[(I - I_\mathcal{G}(\phi))^T (I - I_\mathcal{G}(\phi))], \tag{13}$$

where $\mathcal{I}$ is the set of ground truth image, Φ is the set of their corresponding poses, $\mathcal{G}$ are the model Gaussian parameters, $I_\mathcal{G}(\phi)$ is the rendering function for pose ϕ, and we assume that the error follows a Gaussian distribution.

We can take the Laplace approximation of the log of the posterior distribution over the model parameters $\mathcal{G}$ on the converged scene parameters $\hat{\mathcal{G}}$ as:

$$-\log p(\mathcal{G}|\mathcal{I}, \Phi) \approx -\log p(\hat{\mathcal{G}}|\mathcal{I}, \Phi) + \frac{1}{2}(\mathcal{G} - \hat{\mathcal{G}}) H(\hat{\mathcal{G}})(\mathcal{G} - \hat{\mathcal{G}}), \tag{14}$$

where:

$$H(\hat{\mathcal{G}}) = -\nabla_\mathcal{G} \nabla_\mathcal{G} \log p(\hat{\mathcal{G}}|\mathcal{I}, \Phi). \tag{15}$$

If we assume a uniform prior, then our claimed Fisher Information matrix from Section 4.1 is precisely the Hessian $H(\hat{\mathcal{G}})$ of this posterior.

From this formulation of our Fisher Information matrix, Proposition 3.5 from [10] gives that the log determinant of the Fisher as the entropy of the second order approximation of $p(\mathcal{G}|\mathcal{I}, \Phi)$ around $\hat{\mathcal{G}}$. If we restrict the posterior to a particular Gaussian's parameters $\mathcal{G}_i$, giving $p(\mathcal{G}_i|\mathcal{I}, \Phi)$, the log determinant of the block diagonal element corresponding to this Gaussian is a measure of entropy for that particular Gaussian. This interpretation gives our pruning scores as a ranking of the Gaussians by their entropy on this posterior.

A.2. Ablation on Patch Size

We ablate our choice of patch size in Table 6. Notice that, although the 4×4 patches that we use in our experiments produce slightly better image quality metrics, the 2×2 and 64×64 patches also produce similar results.

A.3. Ablation on Per-Round Pruning Percentages

Figure 8 plots the average metrics for each permutation of per-round pruning percentages that results in approximately 90% total pruning across all scenes in the Mip-NeRF 360 dataset. Similar to the *bicycle* scene evaluated in Figure 7, pruning 80% of Gaussians in the first round and then 50% in the second optimizes image quality and rendering speed at exactly 90% total pruning. Our method outperforms LightGaussian across all metrics and per-round pruning percentage permutations.

Table 6. Mean PSNR, SSIM, LPIPS, FPS, and point cloud size for the Mip-NeRF 360 dataset using our sensitivity score computed with 2×2, 4×4 and 8×8 patches.

Methods	Mip-NeRF 360				
	PSNR↑	SSIM↑	LPIPS↓	FPS↑	Size (MB)↓
3D-GS	27.47	0.8123	0.2216	83.88	746.46
2×2	26.46	**0.7869**	0.2742	**218.59**	74.65
4×4 (Ours)	**26.67**	0.7862	**0.2719**	204.81	74.65
8×8	26.53	0.7775	0.2780	189.23	74.65

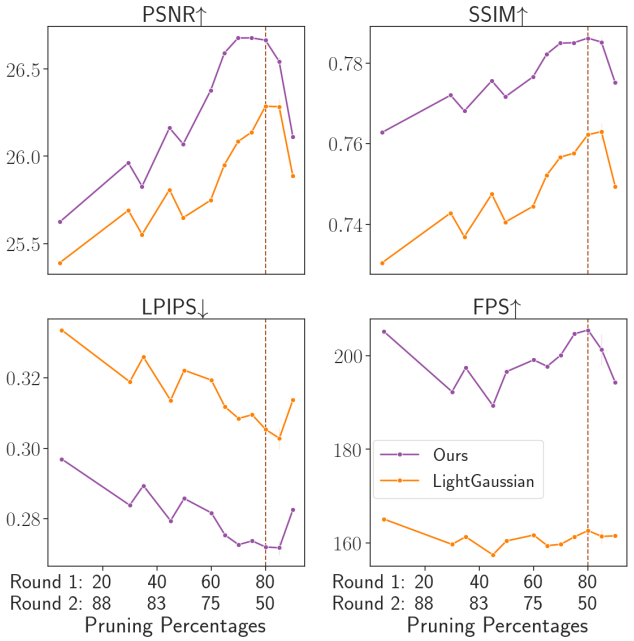

Figure 8. The average PSNR, SSIM, LPIPS, and FPS across the MipNeRF-360 dataset scene after pruning approximately 90% of Gaussians with different per-round percentages in our two-round pipeline. The dotted red line denotes our chosen per-round pruning percentages of 80% then 50%.

A.4. Scene Evaluations

PSNR, SSIM, LPIPS, and FPS for each scene from the Mip-NeRF 360, Tanks&Temples, and Deep Blending datasets that was used in 3D-GS [9] are recorded in Tables 7, 8, 9, and 10, respectively. Note that the sizes of the pruned scenes in this section are identical because exactly 90% of Gaussians were removed from each of them using our two step prune-refine method. FPS is collected using a Nvidia RTXA4000 GPU.

Table 7. PSNR on each scene after two steps of prune-refine.

Methods	Mip-NeRF 360									Tanks&Temples		Deep Blending	
	bicycle	bonsai	counter	flowers	garden	kitchen	room	stump	treehill	train	truck	drjohnson	playroom
Baseline (3D-GS)	25.09	32.25	29.11	21.34	27.28	31.57	31.51	26.55	22.56	22.10	25.43	28.15	29.81
LightGaussian	24.34	29.64	27.57	20.70	25.78	29.45	30.65	25.88	**22.49**	**21.35**	**24.81**	27.73	29.29
Ours	**24.72**	**30.64**	**28.00**	**20.86**	**26.23**	**29.83**	**31.03**	**26.30**	22.39	21.03	24.40	**28.00**	**29.71**

Table 8. SSIM on each scene after two steps of prune-refine.

Methods	Mip-NeRF 360									Tanks&Temples		Deep Blending	
	bicycle	bonsai	counter	flowers	garden	kitchen	room	stump	treehill	train	truck	drjohnson	playroom
Baseline (3D-GS)	0.7467	0.9457	0.9140	0.5875	0.8558	0.9317	0.9255	0.7687	0.6352	0.8134	0.8782	0.8778	0.8854
LightGaussian	0.6801	0.8921	0.8562	0.5343	0.7833	0.8830	0.8989	0.7256	0.5956	0.7349	0.8512	0.8546	0.8747
Ours	**0.7270**	**0.9261**	**0.8917**	**0.5548**	**0.8189**	**0.9128**	**0.9152**	**0.7570**	**0.6248**	**0.7600**	**0.8541**	**0.8762**	**0.8861**

Table 9. LPIPS on each scene after two steps of prune-refine.

Methods	Mip-NeRF 360									Tanks&Temples		Deep Blending	
	bicycle	bonsai	counter	flowers	garden	kitchen	room	stump	treehill	train	truck	drjohnson	playroom
Baseline (3D-GS)	0.2442	0.1811	0.1838	0.3602	0.1225	0.1165	0.1973	0.2429	0.3460	0.2077	0.1476	0.2895	0.2823
LightGaussian	0.3338	0.2568	0.2801	0.4258	0.2407	0.2042	0.2625	0.3132	0.4315	0.3227	0.2041	0.3383	0.3201
Ours	**0.2965**	**0.2281**	**0.2297**	**0.4211**	**0.1997**	**0.1545**	**0.2278**	**0.2836**	**0.4062**	**0.2967**	**0.1916**	**0.3067**	**0.2963**

Table 10. FPS on each scene after two steps of prune-refine. Results were collected with a Nvidia RTXA4000 GPU.

Methods	Mip-NeRF 360									Tanks&Temples		Deep Blending	
	bicycle	bonsai	counter	flowers	garden	kitchen	room	stump	treehill	train	truck	drjohnson	playroom
Baseline (3D-GS)	32.32	106.65	79.70	66.48	37.60	64.46	76.31	54.04	59.08	114.45	81.26	56.79	76.79
LightGaussian	110.94	208.15	168.48	182.28	133.77	165.98	172.46	169.98	147.03	378.50	279.55	206.94	261.27
Ours	**127.85**	**289.21**	**222.77**	**205.21**	**164.79**	**237.66**	**231.63**	**179.38**	**184.78**	**494.76**	**287.44**	**284.81**	**318.06**

A.5. Prune-Refining EAGLES

In Table 11, we ablate our PUP 3D-GS pipeline against LightGaussian's pipeline on an EAGLES [4] model of the Mip-NeRF 360 *bicycle* scene. Notice that the base EAGLES model produces similar metrics to vanilla 3D-GS despite being $2.51\times$ smaller than it. By applying PUP 3D-GS to the EAGLES model, we further reduce its size to $25.14\times$ smaller than the vanilla 3D-GS model while achieving better image quality and rendering speed than LightGaussian.

Table 11. Results from training EAGLES [4] on the *bicycle* scene and then running two steps of prune-refine with our and LightGaussian's methods.

Methods	Mip-NeRF 360 Bicycle Scene				
	PSNR↑	SSIM↑	LPIPS↓	FPS↑	Size (MB)↓
3D-GS	25.09	0.7467	0.2442	32.12	1345.58
Baseline (EAGLES)	25.07	0.7508	0.2433	47.82	535.21
EAGLES + LightGaussian	23.57	0.6082	0.4039	109.26	53.52
EAGLES + Ours	**24.01**	**0.6686**	**0.3566**	**144.56**	53.52

A.6. Additional Scene Visualizations

Figure 9 provides a visual comparison of the ground truth image against renderings from the 3D-GS model before and after pruning with our PUP 3D-GS and LightGaussian's pipelines. Notice that our method consistently achieves higher visual fidelity and retains more salient foreground information like individual leaves and legible text.

A.7. Scene Residuals

Figures 10 and 11 provide visual comparisons of the L1s residual of renderings from the 3D-GS model before and after pruning 90% of Gaussians with our PUP 3D-GS and LightGaussian's pipelines. Figure 10 compares the L1 residuals with respect to the ground truth image, while Figure 11 compares them with respect to renderings from the base 3D-GS model. In both cases, our PUP 3D-GS pipeline produces less L1 error than LightGaussian's.

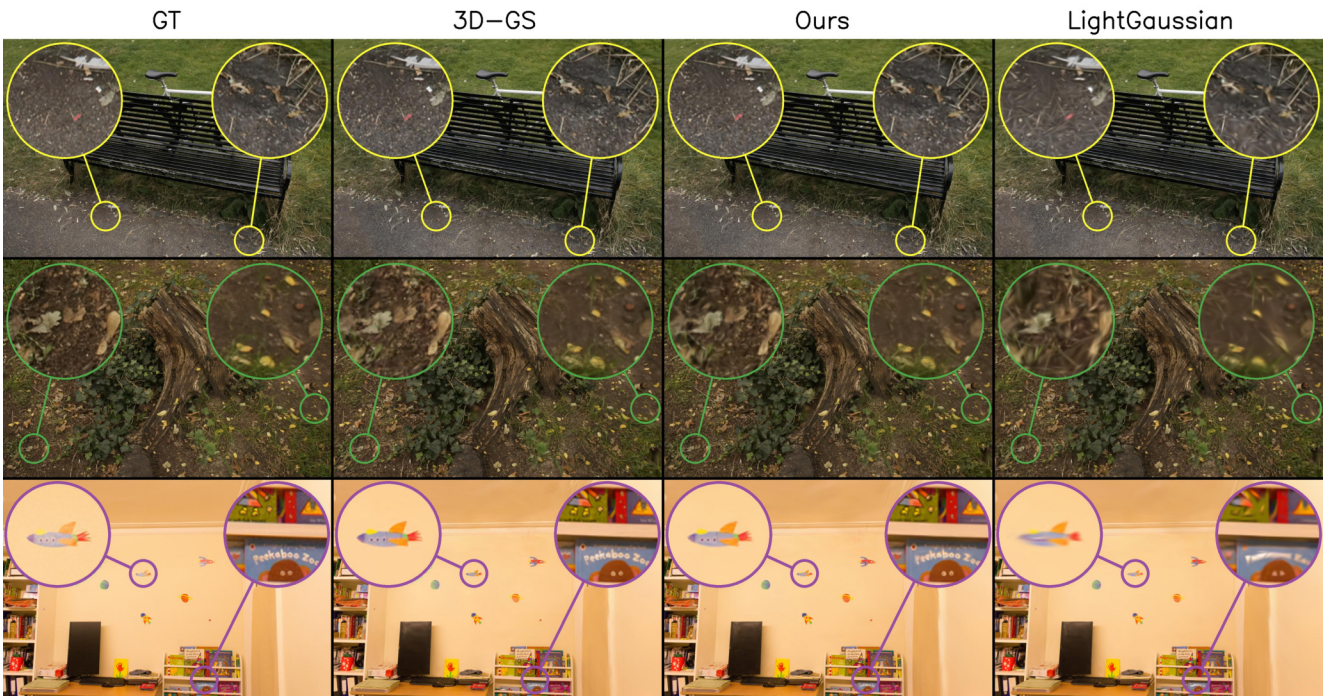

Figure 9. Visual comparison after two rounds of prune-refine using our method and LightGaussian's method. Top: *bicycle* from the Mip-Nerf 360 dataset. Middle: *stump* from the Mip-Nerf 360 dataset. Bottom: *playroom* from the Deep Blending dataset. A larger example image of *playroom* can be found in Figure 1.

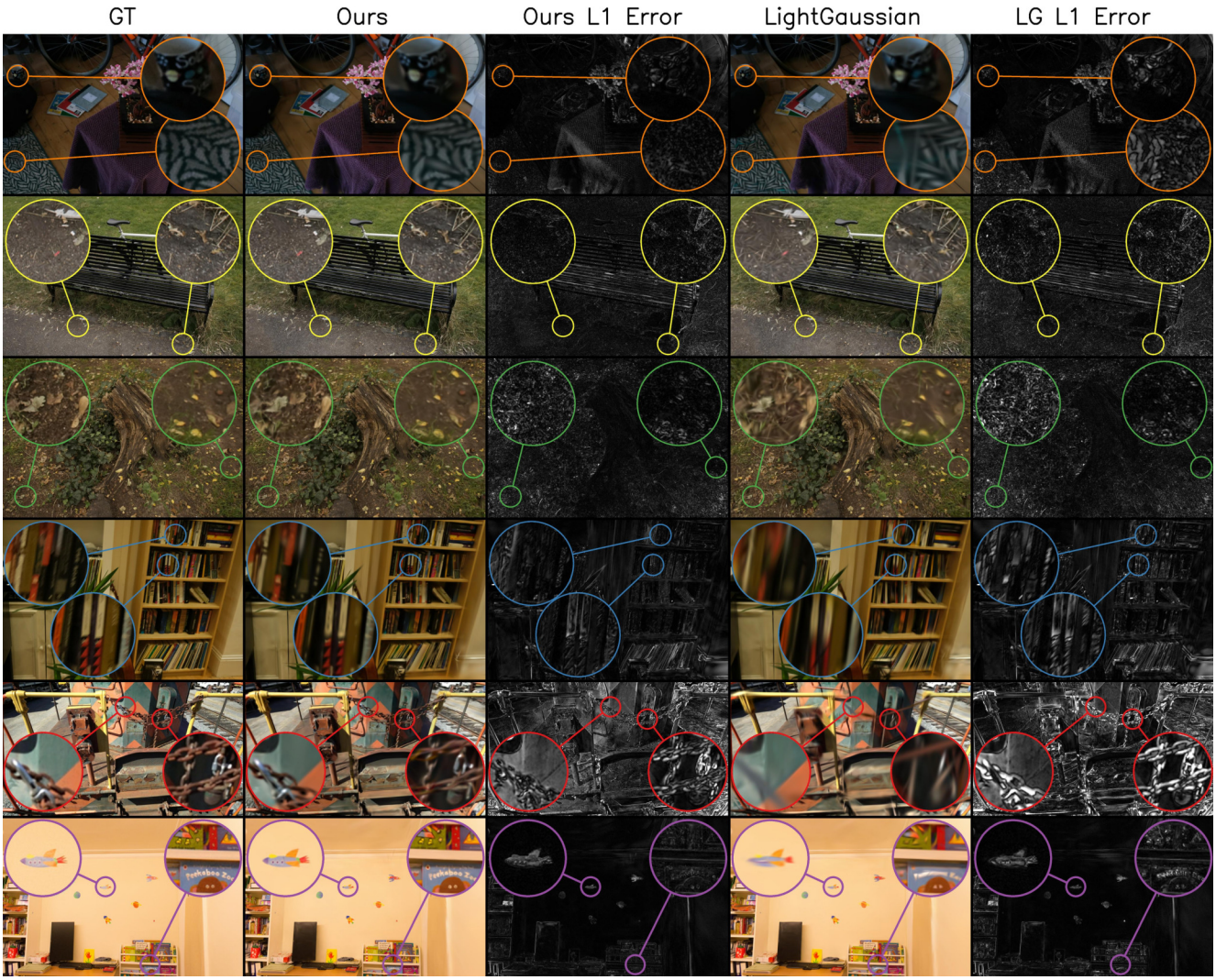

Figure 10. Visual comparison after two rounds of prune-refine using our method and LightGaussian's with additional L1 error visualizations against the ground truth images. Columns "GT", "Ours", and "LightGaussian" are the same as in Figure 4. Columns "Ours L1 Error" and "LG L1 Error" are the L1 Error images of our method and LightGaussian against the ground truth images.

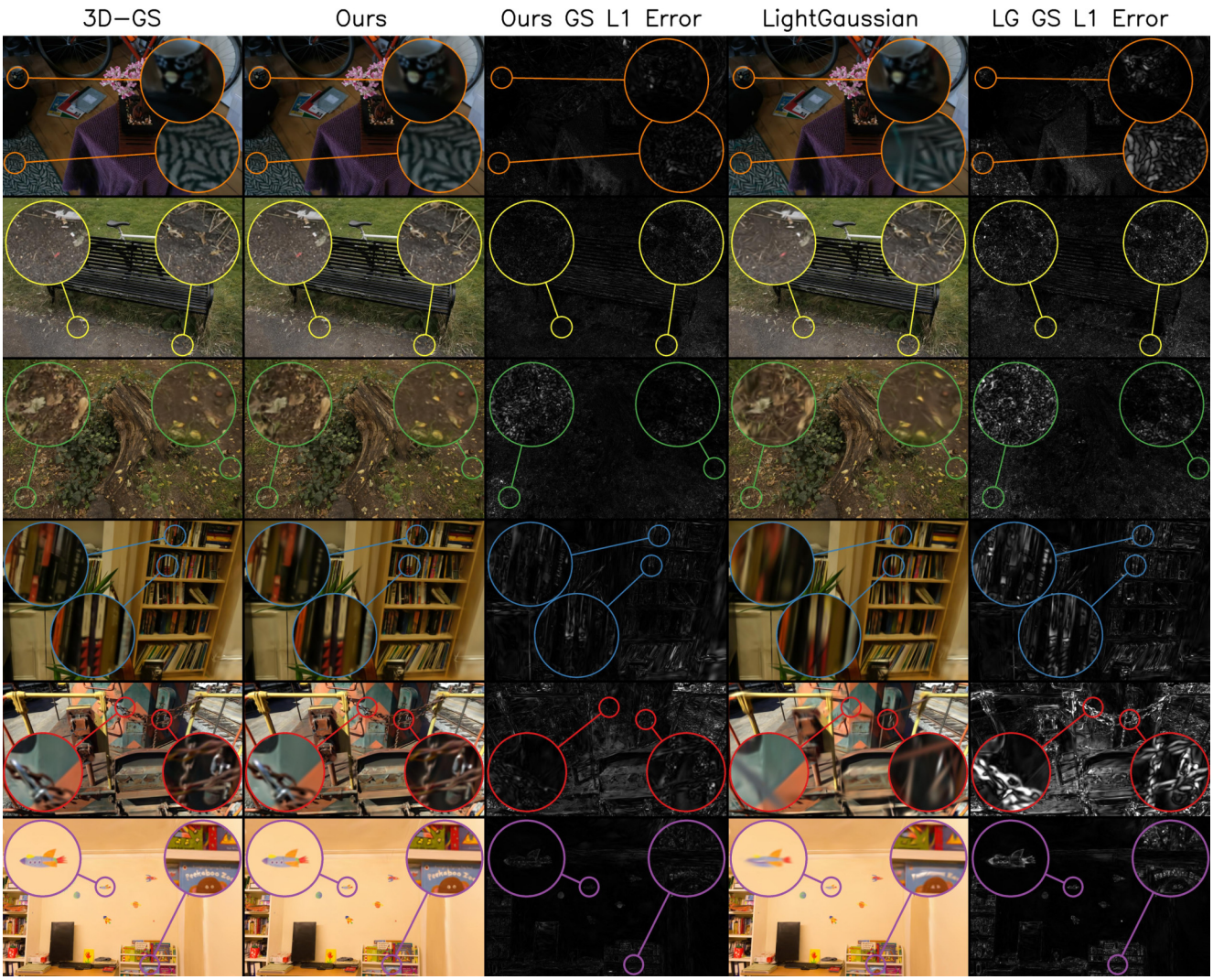

Figure 11. Visual comparison after two rounds of prune-refine using our method and LightGaussian's with additional L1 error visualizations against renderings from the base 3D-GS model. Columns "3D-GS", "Ours", and "LightGaussian" are the same as in Figure 4. Columns "Ours GS L1 Error" and "LG GS L1 Error" are the L1 error images of our method and LightGaussian against the original 3D-GS reconstruction of the scene.