MesonGS: Post-training Compression of 3D Gaussians via Efficient Attribute Transformation

Shuzhao Xie[1], Weixiang Zhang[1], Chen Tang[1,3], Yunpeng Bai[4], Rongwei Lu[1], Shijia Ge[1], and Zhi Wang[1,2,†]

[1] SIGS & TBSI, Tsinghua University, [2] Peng Cheng Laboratory
[3] MMLab, The Chinese University of Hong Kong, [4] The University of Texas at Austin
https://shuzhaoxie.github.io/mesongs/

Abstract. 3D Gaussian Splatting demonstrates excellent quality and speed in novel view synthesis. Nevertheless, the huge file size of the 3D Gaussians presents challenges for transmission and storage. Current works design compact models to replace the substantial volume and attributes of 3D Gaussians, along with intensive training to distill information. These endeavors demand considerable training time, presenting formidable hurdles for practical deployment. To this end, we propose *MesonGS*, a codec for post-training compression of 3D Gaussians. Initially, we introduce a measurement criterion that considers both view-dependent and view-independent factors to assess the impact of each Gaussian point on the rendering output, enabling the removal of insignificant points. Subsequently, we decrease the entropy of attributes through two transformations that complement subsequent entropy coding techniques to enhance the file compression rate. More specifically, we first replace rotation quaternions with Euler angles; then, we apply region adaptive hierarchical transform to key attributes to reduce entropy. Lastly, we adopt finer-grained quantization to avoid excessive information loss. Moreover, a well-crafted finetune scheme is devised to restore quality. Extensive experiments demonstrate that MesonGS significantly reduces the size of 3D Gaussians while preserving competitive quality.

Keywords: Compression · Gaussian Splatting · Novel View Synthesis

1 Introduction

Novel view synthesis is a fundamental task in 3D vision and has significant applications in virtual reality, augmented reality, and photography. This task involves using a collection of images captured from different viewpoints, along with their corresponding camera poses, with the objective of generating highly realistic images from arbitrary viewpoints. By reparameterizing the point with a 3D Gaussian function in the point cloud, 3D Gaussian Splatting (3D-GS) [23] shows excellent quality and real-time rendering speed in this task. A Gaussian point consists of a 3D coordinate, spherical harmonics (SH) coefficients to represent

† Corresponding author.

its color, an opacity parameter, a scale vector, and a rotation quaternion. 3D-GS utilizes scale vectors and rotation quaternions to characterize the covariance matrix of the 3D Gaussian function. The coordinates of Gaussians are typically referred to as *geometry*, while the other parameters of Gaussians are denoted as *attributes*. Despite the efficiency of 3D-GS, the sheer volume of Gaussians and the multi-channel attributes within each Gaussian result in a considerable file size. Notably, 5.27×10^6 Gaussians are required to represent the *bicycle* scene in the Mip-NeRF 360 dataset [1], occupying 1.3 GB of storage under 32-bit float precision. This sizable file poses challenges in transmission and storage. Hence, it is essential to design a tailored codec for 3D Gaussians.

Due to the diverse attributes and intricate rendering procedure of the 3D-GS, developing an efficient compression method for 3D Gaussians presents significant challenges. Previous studies on point cloud compression [8, 12, 44, 55] involve voxelizing the point cloud and applying transformations, quantization, and entropy encoding. However, these approaches cannot support fine-tuning to restore the quality of compressed 3D Gaussians and are limited to conventional point clouds with basic attributes like color and normals. In contrast, 3D-GS encompasses a broader spectrum of attributes. Blindly applying existing methods results in notable artifacts due to the high sensitivity of certain 3D-GS attributes. Hence, adapting traditional compression techniques for 3D Gaussians is non-trivial. Some concurrent works [11, 17, 30, 32, 33, 35, 36, 63] have explored compressing 3D Gaussians using vector quantization and finetuning. Nevertheless, these approaches typically separate geometry from attributes during compression, disregarding the potential similarities among attributes at the 3D geometric level. This oversight prevents efficiently utilizing geometry information to further reduce attribute redundancy. Additionally, the requirement for training puts tremendous pressure on real-world applications.

To address the aforementioned issues, we introduce MesonGS, a 3D Gaussians codec, which employs universal Gaussian pruning, attribute transformation, and block quantization. In *universal Gaussian pruning*, we consider both view-dependent and view-independent features to assess the importance of Gaussians. The gradient-based importance [33] tends to select Gaussians with large gradients rather than Gaussians that contribute significantly to the rendering results, making it unsuitable for the poorly learned 3D-GS. In contrast, our method accurately evaluates importance through forward propagation and is applicable in all scenarios. Besides, compared to opacity-based importance [11], we incorporate view-dependent color features to achieve more accurate evaluations. Regarding *attribute transformation*, we first transform the rotation quaternions (4 numbers) into Euler angles (3 numbers), a lossless process that reduces the storage requirement for each Gaussian by one number. Then, we adopt region adaptive hierarchical transform (RAHT) [8] to reduce the entropy of key attributes – opacity, scales, Euler angles, and 0-degree SH coefficients. RAHT involves transforming a channel of the attribute into a DC coefficient and several concentrated distributed AC coefficients. Since the entropy of AC coefficients is lower, the entropy coding methods can compress the attribute into a smaller size.

For *block quantization*, we divide each attribute channel into multiple blocks and perform quantization for each block individually. This approach prevents quantization from becoming the quality bottleneck and provides increased flexibility. We employ vector quantization [10] to significantly compress unimportant attributes – SH coefficients with degrees greater than 0. We utilize octree to compress geometry and pack all the elements with LZ77 [16,59,60] codec. To achieve a fair comparison with related works and restore the quality, we also propose an elaborated finetune scheme. Comprehensive experiments are conducted to demonstrate the exceptional compression quality of our method. Additionally, we evaluate our method against previous neural radiance field (NeRF) compression techniques.

Our contributions can be summarized as follows:

- We propose two transformations to reduce the redundancy and entropy in attributes. This involves using Euler angles to replace rotation quaternions and applying RAHT to key attributes. These transformations lead to a 13× increase in compression rate with negligible quality degradation.
- We derive a measure that considers both view-dependent and -independent factors to prune the insignificant Gaussians, suiting both bounded and unbounded scenes. We also adopt a finer-grained method to avoid excessive information loss caused by quantization. Finally, a well-crafted finetune scheme is devised to restore quality.
- Extensive experiments demonstrate the compression quality of our pipeline. Comparisons with concurrent works demonstrate the universality of our pruning and transformation strategies. Additionally, we have compared our work with compressed NeRF to reveal the strengths and limitations between Grid-based NeRF and 3D Gaussians in the 1MB storage.

2 Related Work

3D Gaussians Compression. Many concurrent works are proposed to compress 3D Gaussians. C3DGS [33] proposes a score to measure the sensitivity of the attributes and replace the quaternions and scales with covariance. They use vector quantization to compress color and geometry features separately. Lee *et al*. [25] give a learning-based pruning strategy, utilizes residual vector quantization to compress the scales and rotations, and compresses the SH coefficients with a NeRF. LightGaussian [11] compresses the geometry with an octree, prunes the Gaussians based on the cumulated opacity and volumes, distills the SHs to a lower degree, and finally compresses the remaining elements with vector quantization. Compact3D [32] uses vector quantization and compresses the indices further by sorting them and using a method similar to run-length encoding. EAGLES [17] treats the parameters of a Gaussian as a vector and proposes an encoder-decoder network to compress the vector into a latent code. SOGS [30] structures the Gaussian attributes into smooth 2D grids. ReducingGS [34] introduces mixed-band SH coefficients to further reduce the file size. These methods

require a significant amount of training time to distill the information of original 3D Gaussians into the compact model, making them unsuitable for resource-constrained scenarios and inapplicable for 3D-GS extensions [37]. Instead of relying on extensive training to identify redundancy in attributes for compression, we propose using RAHT to reveal spatial redundancy in attributes and then combining it with entropy coding to reduce the file size further. Besides, compared to [11, 33], we propose a Gaussian importance assessment index that considers both view-dependent and -independent factors, suiting both bounded and unbounded scenes. Furthermore, we replace the quaternions with the Euler angles, showing better quality than covariance-based replacement [33].

NeRF Compression. Neural Radiance Field (NeRF) compression primarily targets grid-based NeRF [2, 5, 6, 21]. The "grid" can be voxel-grids [15, 45], triplanes [4], point clouds [53], or hash grids [31]. Though grid-based NeRF achieves great acceleration, it introduces huge storage requirements, leading to the emergence of NeRF compression works. VQAD [46] proposes to generalize different versions of a NeRF with hierarchical coding methods. [9,26,57] propose to use frequency domain transformation to reduce the storage demand of the voxel grids. SHACIRA [18] and CAwa-NeRF [27] compress the hash grid of InstantNGP [31]. BiRF [43] introduces a binary quantization scheme. Masked-wavelet-NeRF [39] and ACRF [13] adopt wavelet transform and RAHT. ReRF [48] is proposed to compress the dynamic NeRF. To accelerate rendering, some works [7, 20, 38] quantize features to 8 bits and save them as 2D images. NeRF and 3D-GS can be interchanged in novel view synthesis, particularly on bounded scenes. Therefore, a comparison between them is valuable. Our work proposes such a comparison to reveal their core competitiveness.

Point Cloud Compression. Point cloud compression consists of geometry compression and attribute compression. The goal of geometry compression is to compress the 3D coordinates of points. Existing methods [28, 41] typically use octree to organize coordinates. Attribute compression generally comprises three steps: transform coding, quantization, and entropy coding. Transform coding involves designing a careful transformation of attributes into the frequency domain to minimize signal redundancy. For instance, [55] constructs a graph from the point cloud and applies the graph fourier transform to attributes. [8] introduces Haar wavelet transforms to attribute compression. Quantization is used to convert coefficients from transform coding into transmitted symbols and reduce the high-frequency components. Entropy coding [22, 40, 51, 52] aims to encode these symbols into a bitstream. 3DAC [12] and Song *et al*. [44] propose learning-based entropy models to further reduce the size.

3 Method

3.1 Preliminary

3D-GS [23] is an explicit 3D scene representation in the form of point clouds, utilizing Gaussians to model the scene. Each Gaussian is characterized by a

covariance matrix $\mathbf{\Sigma}$ and a center point $\mathbf{X}$, which is referred to as the mean value of the Gaussian:

$$G(x) = e^{-\frac{1}{2}\mathbf{X}^\top \mathbf{\Sigma}^{-1}\mathbf{X}}. \tag{1}$$

To maintain the positive definiteness of the covariance matrix $\mathbf{\Sigma}$, 3D-GS decomposes $\mathbf{\Sigma}$ into a scaling matrix $\mathbf{S} = \text{diag}(\mathbf{s}), \mathbf{s} \in \mathbb{R}^3$ and a rotation matrix $\mathbf{R}$: $\mathbf{\Sigma} = \mathbf{RSS}^\top \mathbf{R}^\top$. The rotation matrix $\mathbf{R}$ is parameterized by a rotation quaternion $\mathbf{q} \in \mathbb{R}^4$. The backpropagation process is illustrated in [23].

When rendering novel views, the technique of splatting [54,61] is employed for the Gaussians within the camera planes. As introduced by [62], using a viewing transform denoted as $\mathbf{W}$ and the Jacobian $\mathbf{J}$ of the affine approximation of the projective transformation, the covariance matrix $\mathbf{\Sigma}'$ in camera coordinates system can be computed by $\mathbf{\Sigma}' = \mathbf{JW\Sigma W}^\top \mathbf{J}^\top$.

In summary, each element of 3D Gaussians has the following parameters: (1) a 3D center $\mu \in \mathbb{R}^3$; (2) a rotation quaternion $\mathbf{q} \in \mathbb{R}^4$; (3) a scale vector $\mathbf{s} \in \mathbb{R}^3$; (4) a color feature defined by spherical harmonics coefficients $\mathbf{SH} \in \mathbb{R}^d$, with $d = 3(f+1)^2$, where f is the harmonics degree; and (5) an opacity logit $o \in \mathbb{R}$. Specifically, for each pixel, the color and opacity of all the Gaussians are computed using Eq. (1). The blending of N ordered points that overlap the pixel is given by:

$$C = \sum_{i \in N} c_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j). \tag{2}$$

Here, c_i and α_i represent the density and color of this point computed by a Gaussian with covariance $\mathbf{\Sigma}$ multiplied by an optimizable per-point opacity and SH color coefficients.

3.2 Overview

As shown in Fig. 1, we first prune insignificant 3D Gaussians based on view-dependent and -independent features. Then, we compress the remaining 3D Gaussians. For geometric compression, octree is employed to compress the positions of the Gaussians. For attribute compression, we begin by replacing rotation quaternions with Euler angles. Then, we categorize the attributes into important and unimportant ones, with the former including opacity, 0-D SH coefficients, scale vectors, and Euler angles and later including the SH coefficients in degrees greater than 0. We apply RAHT and block quantization to important attributes. Note that RAHT is not applied to the scale vectors at the 8-bit quantization. Unimportant attributes are significantly compressed through vector quantization. Finally, all components are packed using the LZ77 [16, 59, 60] codec.

3.3 Gaussians Pruning

As 3D-GS has a huge number of points, pruning unimportant Gaussians is a necessary step. We first define the importance score of each Gaussian. Here the importance score refers to the contribution to the final rendering results. For

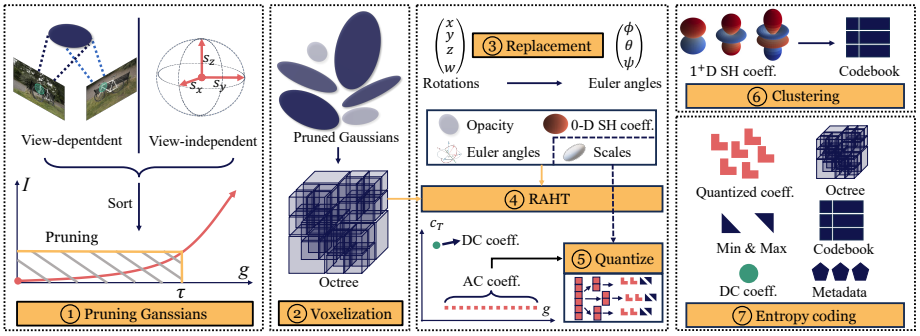


Fig. 1: Overview of MesonGS. ① We prune insignificant Gaussians by considering both view-dependent and view-independent features. ② Geometry compression is performed using an octree to generate voxelized coordinates for future transformations. ③ We replace rotation quaternions with Euler angles. ④ Further compression is achieved by applying RAHT and ⑤ block quantization to important attributes. Notably, RAHT is not applied to the scales when the quantization bit is 8 (refer to the dashed arrow). ⑥ To significantly compress the 1^+D SH coefficients, vector quantization is employed. ⑦ All components are packed by LZ77 [16, 59, 60] codec.

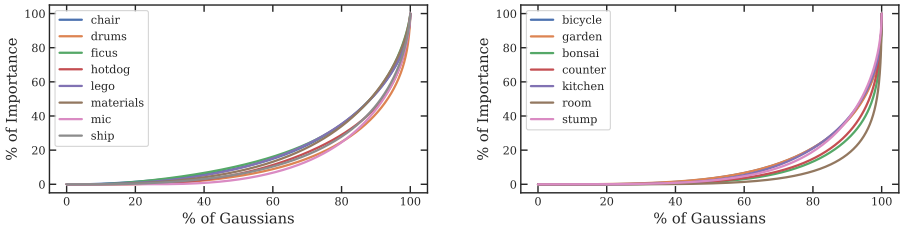
a Gaussian g, we define its importance score I_g as the product of the view-dependent importance score I_d and the view-independent importance score I_i: $I_g = I_d I_i$. Based on Eq. (2), we define view-dependent importance score I_d as:

$$I_d = \sum_{p \in \mathcal{P}} \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j). \tag{3}$$

Here, $\mathcal{P}$ is the pixel set that is overlapped by the Gaussian g, and i is the rank of Gaussian g in a set of Gaussians that overlap with the pixel p. In contrast to VQRF [26], where the importance score is the mean value of corresponding sample points, our method allows for the direct recording of the importance score throughout the testing phase. LightGaussian [11] uses the opacity (α_i in Eq. (3)) as the view-dependent score. They have not considered the masking caused by other Gaussians. The usage of backward gradients as importance scores in C3DGS [33] is not ideal for 3D-GS that is not well-learned. Typically, Gaussians that are not well-learned exhibit larger gradients. However, these poorly learned Gaussians may not be the significant ones, making the pruning strategy of C3DGS ineffective in inadequately learned 3D Gaussians.

The view-independent score I_i is given by $I_i = (V_{\text{norm}})^\beta$. Here, the volume V is the product of the scale vector. To obtain V_{norm}, we normalize the V by the 90% largest of all sorted Gaussians and clip the range between 0 and 1, β is the hyperparameter to control the size of I_i.

In Fig. 2, we sort the importance score and visualize its cumulative distribution function (CDF). We notice that 40% of the Gaussians contain over 80% of the importance. Hence, we use an importance threshold τ to prune Gaussians, meaning we cut the percent of τ of the sorted Gaussians.



(a) Bounded scenes. **(b)** Unbounded scenes.

Fig. 2: Quantile-quantile curve, which means x% of least important Guasisans contributes to y% percent of total importance. For both kinds of scenes, 40% of the Gaussians contribute over 80% of the importance. The importance refers to the contribution to the final rendering results.

3.4 Geometry Compression

After pruning, we compress the 3D positions with the octree structure. An octree recursively divides occupied voxels into eight sub-voxels until reaching the required resolution. The occupancy symbol is composed of 8 bits (1 to 255 in decimal), where each bit indicates the occupancy status of the corresponding subvoxel. We use the depth d to control the size of the octree. When multiple Gaussians exist within a voxel, we average the corresponding attributes for deduplication.

3.5 Attribute Transformation and Compression

We categorize attributes into important attributes and unimportant attributes. Important attributes include opacity, scales, rotations/Euler angles, and 0D-SH coefficients. Here, the 0D-SH coefficient refers to SH coefficients in degree 0. The unimportant attributes refer to the SH coefficients in degrees greater than 0.

Replacement. We replace the rotation quaternion (4 numbers) with the corresponding Euler angles (3 numbers). The Euler angles are three angles that describe the orientation of a rigid body with respect to a fixed coordinate system. This replacement can reduce the storage requirement by one floating-point number for each Gaussian Point.

Specifically, for a quaternion $\mathbf{q} = [w, x, y, z] \in \mathbb{R}^4$, we calculate the euler angle $\mathbf{e} = [\phi, \theta, \psi] \in \mathbb{R}^3$ with:

$$\begin{bmatrix} \text{atan2}(2(wx+yz), 1-2(x^2+y^2)) \\ -\frac{\pi}{2} + 2\text{atan2}(\sqrt{1+2(wy-xz)}, \sqrt{1-2(wy-xz)}) \\ \text{atan2}(2(wz+xy), 1-2(y^2+z^2)) \end{bmatrix}. \tag{4}$$

During decoding, we directly build the rotation matrix $\mathbf{R}$ from the Euler angles by:

$$\begin{bmatrix} \mathrm{C}_\theta\mathrm{C}_\psi & -\mathrm{C}_\phi\mathrm{S}_\psi + \mathrm{S}_\phi\mathrm{S}_\theta\mathrm{C}_\psi & \mathrm{S}_\phi\mathrm{S}_\psi + \mathrm{C}_\phi\mathrm{S}_\theta\mathrm{C}_\psi \\ \mathrm{C}_\theta\mathrm{S}_\psi & \mathrm{C}_\phi\mathrm{C}_\psi + \mathrm{S}_\phi\mathrm{S}_\theta\mathrm{S}_\psi & -\mathrm{S}_\phi\mathrm{C}_\psi + \mathrm{C}_\phi\mathrm{S}_\theta\mathrm{S}_\psi \\ -\mathrm{S}_\theta & \mathrm{S}_\phi\mathrm{C}_\theta & \mathrm{C}_\phi\mathrm{C}_\theta \end{bmatrix}. \tag{5}$$

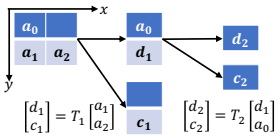


Fig. 3: 2D example of RAHT.

Fig. 4: Effect of 1^+D spherical harmonics coefficients. "FT" refers to finetune.

Here S_θ and C_θ represents sine and cosine of θ. Similarly, ϕ and ψ follow the same notation.

As the covariance matrix is symmetrical, replacing the scales and rotation quaternions (7 numbers) with the upper triangular part (6 numbers) of the covariance matrix seems to be an alternative. This strategy can also reduce the storage of one number. Our contemporary, C3DGS [33], proposes a similar covariance-based replacement strategy. However, the subsequent quantization steps will cause a large number of covariance matrices to become indefinite, resulting in degraded rendering outcomes. In contrast, using Euler angles can ensure that the positive definiteness of covariance is not compromised. The comparison results are illustrated in Fig. 7.

Region adaptive hierarchical transform. To reduce signal redundancy in important attributes, we employ RAHT [8]. RAHT involves taking voxelized coordinates from octree and converting the corresponding attributes into transformed coefficients. Each channel of the transformed coefficients consists of a direct current (DC) coefficient and several alternating current (AC) coefficients. The low entropy of the AC coefficients enables the subsequent entropy coding procedure to achieve a larger compression rate. Here, we briefly introduce the RAHT through a 2D example. Fig. 3 shows how to apply RAHT to a_0, a_1, and a_2. First, RAHT merges the coefficients along the x axis with the transform:

$$T_1 = \frac{1}{\sqrt{w_1 + w_2}} \begin{bmatrix} \sqrt{w_1} & \sqrt{w_2} \\ -\sqrt{w_2} & \sqrt{w_1} \end{bmatrix}, \tag{6}$$

where the weight coefficient w_i is the number of Gaussians that a_i contains. If a_i is a leaf node in octree, then $w_i = 1$. If a_i is not a leaf node, then w_i is the sum weight of its son leaf nodes. After applying T_1 to a_1 and a_2, we get a DC coefficient d_1 and an AC coefficient c_1. DC coefficient d_1 is going to do further transformation with a_0 while AC coefficients are saved for encoding. For coefficients that have no counterpart, like a_0, we transmit it to the next layer. At the end, we obtain a DC coefficient d_2 and two AC coefficients c_2 and c_1. During the decoding process, we obtain the weight coefficients from the octree.

Block quantization. After applying RAHT to important attributes, we save the DC coefficient in float and quantize the AC coefficients. We use block-wise quantization [14, 47, 50, 58] to prevent significant quality degradation caused by the coarse-grained channel-wise quantization. Specifically, we first partition a channel of attributes into multiple blocks. Then we quantize a block $\mathbf{c}$ with:

$$\mathbf{c}_q = \lfloor \text{clamp}(\frac{\mathbf{c}}{S_c} + Z_c, 0, 2^b - 1) \rceil, \tag{7}$$

where

$$S_c = \frac{\max(\mathbf{c}) - \min(\mathbf{c})}{2^b}, Z_c = \lfloor 2^b - \frac{\max(\mathbf{c})}{S_c} \rceil. \tag{8}$$

Here, b refers to the bit-width, $\lfloor \cdot \rceil$ represents the rounding-to-nearest function, and $\mathbf{c}_q$ refers to the quantized attributes. Besides, function clamp$(\cdot)$ specifies a range of values. Values below the minimum are set to the minimum. Values above the maximum are set to the maximum.

Note that we do not apply RAHT to the scale vectors when the quantization bit is 8. The reason is that the activation function of the scale vector is an exponential function, which magnifies the errors caused by transformation and quantization. We save the minimum and maximum values of all blocks of quantized attributes for decoding.

Clustering 1^+D SH coefficients. The size of spherical harmonics (SH) coefficients takes up 85.7% in a 3D Gaussians file. In the middle of Fig. 4, after applying 1-bit quantization to the 1^+D SH coefficients, only the reflectance is affected, while the overall structure and color remain unchanged. Therefore, quantization is not the optimal choice for 1^+D SH coefficients. We employ vector quantization to significantly reduce the size of these SH coefficients. Specifically, we use a codebook and a corresponding index mapping table to connect the origin vectors with the vectors in the codebook. The right side of Fig. 4 shows the final result of our method. To reduce memory occupation and encoding time, a batched clustering strategy [42] is used. We use multiple iterations to update the clustering results.

Encoding. The final file contains the following components: (1) Octree; (2) DC coefficients and Quantized coefficients; (3) Codebook and the corresponding mapping table; (4) Metadata: Min-Max values of each block of quantized coefficients, octree depth, block size. Notably, the DC coefficients, codebook, and metadata are stored in floating-point format, whereas other components are saved as integers. We compress them via LZ77 [16, 59, 60] codec.

Finetune. We propose a finetune scheme to achieve a fair comparison with baselines and solve the problem of former methods not supporting backpropagation. Specifically, we fix the coordinates of pruned 3D Gaussians and only finetune the attributes. We simulate the encoding and decoding processes during the forward process. To pass the gradients during the backward process, we employ the straight-through estimator [3] for quantization.

4 Experiments

Datasets and compression settings. (1) **Mip-NeRF 360.** The Mip-NeRF 360 dataset [1] contains five outdoor and four indoor scenes. Each scene contains 100 to 300 images. We use the images at 1600×1063. Note that the undocumented *flower* and *treehill* scenes are not included in our evaluation. (2) **Tank&Temples.** This dataset [24] contains two scenes, including *train* and

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

Method	Mip-NeRF 360				Tank&Temples				Deep Blending			
	PSNR	SSIM	LPIPS	Size (M)	PSNR	SSIM	LPIPS	Size (M)	PSNR	SSIM	LPIPS	Size (M)
3DGS [23]	28.98	0.865	0.193	641.70	23.36	0.838	0.187	421.90	29.56	0.898	0.250	703.77
C3DGS [33]	28.49	**0.858**	**0.205**	27.82	**23.32**	0.832	0.194	17.28	29.38	0.898	**0.238**	25.30
Lee *et al.* [25]	28.60	0.856	0.209	46.98	**23.32**	0.831	0.201	39.40	**29.79**	**0.901**	0.258	43.20
Our	27.70	0.838	0.224	**27.62**	22.85	0.822	0.208	**16.99**	29.08	0.895	0.260	**24.76**
Our-FT	**28.61**	0.856	0.206	**27.62**	**23.32**	**0.837**	**0.193**	**16.99**	29.51	**0.901**	0.251	**24.76**

Table 2: Quantitative comparison on Synthetic-NeRF. The best results overall are bolded in each metric, and the second-best results are underlined.

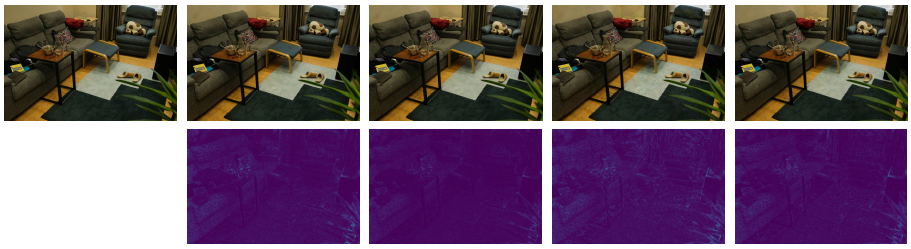
Method	PSNR	SSIM	LPIPS	Size (M)	Method	PSNR	SSIM	LPIPS	Size (M)
3DGS [23]	33.37	0.970	0.031	68.55	DVGO [45]	31.90	0.956	0.035	105.92
C3DGS [33]	32.94	0.967	**0.033**	3.68	VQRF [26]	31.77	0.954	**0.036**	1.43
Lee *et al.* [25]	**33.33**	**0.968**	0.034	8.61	ACRF [13]	**31.79**	0.954	0.037	1.15
Our	32.25	0.963	0.038	**3.65**	Our	29.37	0.947	0.051	**1.03**
Our-FT	32.92	**0.968**	**0.033**	3.66	Our-FT	31.75	**0.962**	0.042	**1.03**

truck. (3) **Deep Blending.** This dataset [19] contains two scenes, including *drjohnson* and *playroom*. (4) **Synthetic-NeRF.** This dataset was first introduced by [29] and has been widely adopted by subsequent work. It contains 8 scenes rendered at 800×800 by Blender. Each scene contains 100 rendered views as the training set and 200 views for testing. As for train-test split and camera parameters estimation, we follow the official implementation of 3D-GS. We evaluate rendering quality and compression performance using PSNR, SSIM [49], LPIPS [56], size, compression rate, and decoding time. All the metrics are evaluated on the test set if not otherwise specified. We set the bit-width as 8 if not otherwise specified. When calculating the important score, we only use the training dataset. To obtain the pre-trained 3D Gaussians for compression, we train 30,000 iterations and then save the checkpoints for both datasets. We set the background as white. Please check the supplementary material for more details.

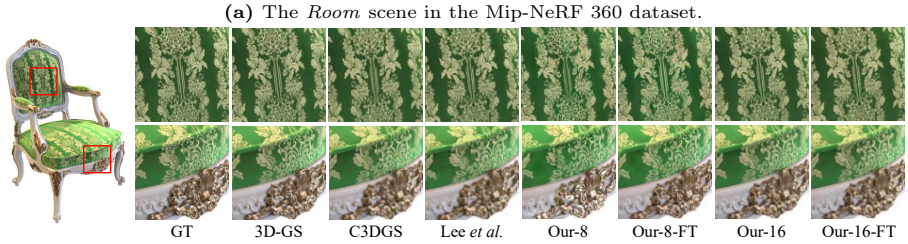
Baselines. We compare our method with the concurrent 3D Gaussian compression works [25, 33] and NeRF compression works [13, 26]. We use the evaluation results from their papers [13, 25, 33].

4.1 Experimental Results

Quantitative result. We compared our approach with original 3D-GS and concurrent works. All of the model sizes are calculated after a standard zip compression. As shown in Tab. 1 and Tab. 2, MesonGS realizes satisfactory performance across all the combinations of methods and datasets. The offline version of MesonGS can achieve a quality similar to the baselines. Moreover, after finetuning, MesonGS achieves performance comparable to the baselines, which reveals the efficiency of our finetune scheme.



(a) The *Room* scene in the Mip-NeRF 360 dataset.



(b) The *Chair* scene in the Synthetic-NeRF dataset.

Fig. 5: Qualitative comparison. We can hardly observe visual artifacts on the rendering result of the compressed model compared to its original model.

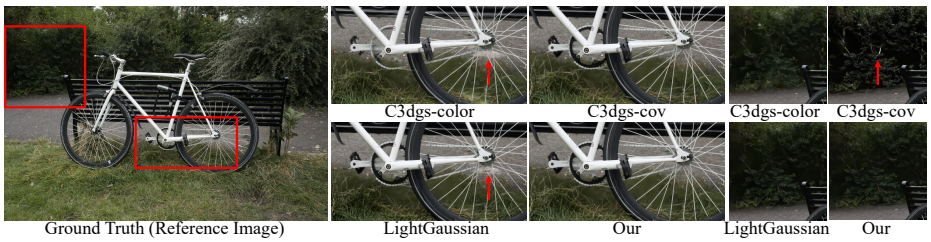


Fig. 6: Qualitative comparison on pruning strategy. The visual quality of our pruning strategy is better than others.

Comparisons with NeRF compression. By now, there is no work comparing compressed 3D Gaussians with compressed NeRF. The right side of Tab. 2 shows the comparison of our 3D Gaussian compression method on Synthetic-NeRF with ACRF and VQRF. We can observe that the quality metrics of baselines are slightly better. The reason is two-fold. On the one hand, our finetune scheme cannot support updating the coordinates. If appropriate updates could be made to the coordinates, the overall quality of our method will be improved. On the other hand, the file sizes of baselines are larger.

Visualization result. We compare the rendering results of all baselines in Fig. 5 across bounded and unbounded scenes. The visual difference is hard to observe in both *chair* and *room* scenes. We also display the per-pixel mean absolute error between ground truth and other baselines at the second line. Our finetune scheme efficiently restores the quality loss caused by offline compression and is highly consistent with the rendering results of 3D-GS.

Pruning strategy. Here we compare our pruning strategies with LightGaussian [11] and C3DGS [33]. To achieve a fair comparison, we prune 66% of the

Table 3: Quantitative comparison on pruning strategy. Our pruning strategy performs better on average. The best results overall are bolded in each metric, and the second-best results are underlined.

Methods	Synthetic-NeRF			Mip-NeRF 360			Average PSNR
	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	
C3DGS-color [33]	**30.79**	**0.9606**	**0.0375**	22.89	0.7852	0.2471	26.84
C3DGS-cov [33]	24.37	0.9104	0.0738	15.04	0.6992	0.3044	19.71
LightGaussian [11]	26.75	0.9372	0.0568	<u>26.93</u>	<u>0.8327</u>	<u>0.2290</u>	26.84
Our(I_i)	30.39	0.9591	0.0389	27.01	0.8411	0.2164	<u>28.70</u>
Our($I_i I_d$)	<u>30.52</u>	<u>0.9597</u>	<u>0.0383</u>	**27.52**	**0.8441**	**0.2139**	**29.02**

Table 4: Encoding time. Our method is the fastest.

Method	NeRF-Synthetic			MipNeRF-360		
	C3DGS [33]	Lee *et al.* [25]	MesonGS	C3DGS [33]	Lee *et al.* [25]	MesonGS
Encoding Time	30 s	480 s	**4 s**	5 min	33 min	**1 min**

Gaussians for all methods. Tab. 3 shows the superiority of our pruning strategy. Moreover, Tab. 3 indicates that it is necessary to incorporate view-dependent importance score I_d in the pruning procedure. Note that the output sizes of these strategies are the same because they prune the same percentage of Gaussians. We also provide a qualitative comparison in Fig. 6 and use the red arrows and circles to mark the artifacts.

Encoding time. As shown in Tab. 4, without finetuning, MesonGS can complete compression in only 20% of the time compared to concurrent works while maintaining similar rendering quality. When the number of Gaussian points is less than 20,000, MesonGS can complete compression quickly using only the CPU, while baseline methods require GPU assistance.

Composition of final storage. In the Synthetic-NeRF dataset, the proportions of the octree, metadata, important attributes, and unimportant attributes are 43%, 0.04%, 34%, and 23%, respectively. In the Mip-NeRF 360 dataset, the proportions of the above four elements are 39%, 0.02%, 47%, and 14%, respectively. The octree and important attributes take up more than half of the storage, while metadata occupies a tiny proportion. The hyperparameter setting is consistent with the results in Tab. 2 and Fig. 5a.

Replacement strategy. We set the bit width as 16 and show the close-up rendering results of Euler-angle-based replacement and covariance-based replacement in the left of Fig. 7. We adjust the final file size by compressing a portion of the covariance. We can see some white line artifacts on the right of the covariance-based strategy. The reason is that the lots of covariance matrices are not positive definite after the quantization, i.e., 92% for Euler angle-based vs. $\sim$50% for covariance-based replacement. Please find more discussion in the supplementary material.

RAHT and quantization. For 8-bit quantization, we recommend not to apply RAHT for scales. Due to the activation function of the scale being an exponential function, it is more sensitive than other attributes. The empirical evidences are shown in Fig. 8. After sequentially applying the RAHT to Opacity, 0D-SH coef-

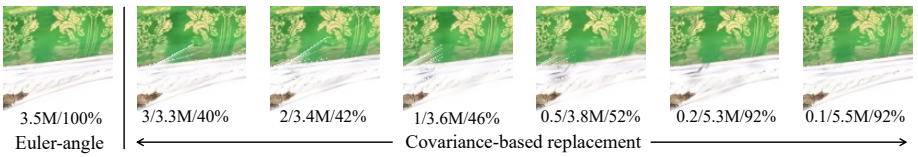


Fig. 7: Euler-angle-based vs. Covariance-based. "A/B/C" refers to the "λ_c / the size of the compressed file / the percent of positive-definite covariance matrices". Replacing scales and rotations with covariance leads to white line artifacts, which greatly affects the visual effect. We adjust the final file size by compressing a portion of the covariance using the λ_c.

Fig. 8: Visual results of different attribute transformation stages. The first image from the left shows the baseline, which means saving the rotations quaternions in the final storage. The second image shows the rendering results after replacing Rotation quaternions with Euler angles. "R+*" refers to applying RAHT to *.

ficients, and Euler angles, we observe minor alterations in the rendering quality. However, severe degradation in rendering quality occurs after applying RAHT to Scales. The information loss caused by the operation of RAHT + Quantization is more significant than only using Quantization, and the exponential function amplifies this error, leading to severe performance degradation.

Block quantization. As shown in Tab. 5, when employing the per-channel quantization strategy and lowering the pruning threshold from 66% to 50%, the compressed file size expands while the rendering quality diminishes. This decline in quality stems from the amplified information loss linked to the extended channel length. As the number of Gaussian points rises, the information loss further intensifies. In contrast, *block quantization* fixes the size of the quantization unit, thereby reducing information loss while providing more flexibility.

4.2 Ablation Study

We conducted several ablation studies on the Mip-NeRF 360 dataset and the Synthetic-NeRF dataset. If not otherwise specified, all the experimental results below have not undergone finetuning. Please find more ablation studies in the supplementary material.

Performance of different stages. We conduct an experiment to demonstrate the benefit of each module in MesonGS. We calculate the size after a zip com-

Table 5: The advantage of block quantization. By fixing the length of the vector requiring quantization, block quantization prevents quantization from becoming a performance bottleneck and provides more flexibility.

Strategy	τ	Synthetic-NeRF				Mip-NeRF 360			
		PSNR(dB)	SSIM	LPIPS	Size(MB)	PSNR(dB)	SSIM	LPIPS	Size(MB)
Channel	66%	29.47	0.9476	0.0511	1.14	25.30	0.7533	0.3074	11.64
	50%	30.65	0.9529	0.0475	1.59	25.35	0.7461	0.3147	16.47
Block	66%	29.60	0.9494	0.0490	1.21	26.28	0.8035	0.2598	12.46
	50%	30.97	0.9560	0.0441	1.73	27.20	0.8238	0.2402	18.42

Table 6: Ablation study of different stages. "TQ" refers to applying RAHT and quantization on important attributes. "Q-scales" refers to quantizing scales.

Stages	Synthetic-NeRF				Mip-NeRF 360			
	PSNR (dB)	SSIM	LPIPS	Size (MB)	PSNR (dB)	SSIM	LPIPS	Size (MB)
3D-GS	33.37	0.9696	0.0305	68.55	28.98	0.8647	0.1931	641.73
+Prune	30.52	0.9597	0.0383	20.99	28.69	0.8612	0.1970	265.94
+Voxel	30.44	0.9592	0.0388	20.72	28.68	0.8610	0.1971	260.58
+Replace	30.44	0.9592	0.0388	20.36	28.68	0.8610	0.1971	255.61
+Cluster	29.71	0.9513	0.0469	5.63	27.74	0.8427	0.2187	79.35
+TQ	29.37	0.9476	0.0510	1.95	27.20	0.8239	0.2401	30.68
+Q-scales	29.37	0.9474	0.0511	1.03	27.20	0.8238	0.2402	18.43
+Fine-tune	31.75	0.9618	0.0416	1.03	27.45	0.8273	0.2357	18.40

pression for fair comparison. As shown in Tab. 6, compared to the uncompressed baseline, Pruning achieves 5× compression but causes a significant PSNR drop for bounded 360 scenes. However, such a drop is slighter for unbounded scenes. The following stages are all influenced by the pruning stage. Of course, all of these stages, instead of Replacement, have caused varying degrees of damage to the rendering quality. We can see that the Replacement step is indeed a free lunch for the 3D-GS attribute compression.

Importance threshold τ. Pruning is a necessary step for 3D Gaussian compression. In Tab. 5, under the bit width of 8, we observe that pruning 33% of points resulted in lower performance compared to pruning 66% of points. The reason for this counterintuitive performance drop is that the bit depth of the quantization step is too low, and pruning 33% of points in the 3D-GS model of unbounded scenes still leaves too many points, resulting in excessive information loss after quantization. Therefore, the quantization bit-width is the bottleneck of overall performance here. In Tab. 5, after increasing the quantization bit-width to 16, the performance of the 33% surpasses other baselines.

5 Conclusion

In this paper, we propose an elaborated designed 3D Gaussians codec. We propose several key components, including the universal Gaussian pruning strategy, elaborated attribute transformation, and flexible block quantization. Extensive experiments demonstrate the superior performance of MesonGS.

Acknowledgments

This work is supported in part by National Key Research and Development Project of China (Grant No. 2023YFF0905502) and Shenzhen Science and Technology Program (Grant No. JCYJ20220818101014030). We thank anonymous reviewers for their valuable advice and JiangXingAI for sponsoring the research.

References

1. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5470–5479 (2022)
2. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Zip-nerf: Anti-aliased grid-based neural radiance fields. ICCV (2023)
3. Bengio, Y., Léonard, N., Courville, A.: Estimating or propagating gradients through stochastic neurons for conditional computation (2013), https://arxiv.org/abs/1308.3432
4. Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields. In: Avidan, S., Brostow, G.J., Cissé, M., Farinella, G.M., Hassner, T. (eds.) Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXXII. Lecture Notes in Computer Science, vol. 13692, pp. 333–350. Springer (2022). https://doi.org/10.1007/978-3-031-19824-3_20
5. Chen, A., Xu, Z., Wei, X., Tang, S., Su, H., Geiger, A.: Factor fields: A unified framework for neural fields and beyond (2023), https://arxiv.org/abs/2302.01226
6. Chen, Z., Li, Z., Song, L., Chen, L., Yu, J., Yuan, J., Xu, Y.: Neurbf: A neural fields representation with adaptive radial basis functions. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4182–4194 (2023)
7. Chen, Z., Funkhouser, T.A., Hedman, P., Tagliasacchi, A.: Mobilenerf: Exploiting the polygon rasterization pipeline for efficient neural field rendering on mobile architectures. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023. pp. 16569–16578. IEEE (2023). https://doi.org/10.1109/CVPR52729.2023.01590
8. De Queiroz, R.L., Chou, P.A.: Compression of 3d point clouds using a region-adaptive hierarchical transform. IEEE Transactions on Image Processing **25**(8), 3947–3956 (2016)
9. 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)
10. Equitz, W.H.: A new vector quantization clustering algorithm. IEEE transactions on acoustics, speech, and signal processing **37**(10), 1568–1575 (1989)
11. Fan, Z., Wang, K., Wen, K., Zhu, Z., Xu, D., Wang, Z.: Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. arXiv preprint arXiv:2311.17245 (2023)
12. Fang, G., Hu, Q., Wang, H., Xu, Y., Guo, Y.: 3dac: Learning attribute compression for point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14819–14828 (2022)

13. Fang, G., Hu, Q., Wang, L., Guo, Y.: ACRF: Compressing explicit neural radiance fields via attribute compression. In: International Conference on Learning Representations(ICLR) (2024)
14. Frantar, E., Ashkboos, S., Hoefler, T., Alistarh, D.: GPTQ: Accurate post-training compression for generative pretrained transformers. ICLR (2023)
15. Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., Kanazawa, A.: Plenoxels: Radiance fields without neural networks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022. pp. 5491–5500. IEEE (2022). https://doi.org/10.1109/CVPR52688.2022.00542
16. Gailly, J.l., Adler, M.: Zlib general purpose compression library. User manual for zlib version **1**(4) (2003)
17. Girish, S., Gupta, K., Shrivastava, A.: Eagles: Efficient accelerated 3d gaussians with lightweight encodings. arXiv preprint arXiv:2312.04564 (2023)
18. Girish, S., Shrivastava, A., Gupta, K.: Shacira: Scalable hash-grid compression for implicit neural representations. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 17513–17524 (2023)
19. Hedman, P., *et al.*: Deep blending for free-viewpoint image-based rendering. ToG (2018)
20. Hedman, P., Srinivasan, P.P., Mildenhall, B., Barron, J.T., Debevec, P.E.: Baking neural radiance fields for real-time view synthesis. In: 2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021. pp. 5855–5864. IEEE (2021). https://doi.org/10.1109/ICCV48922.2021.00582
21. Hu, W., Wang, Y., Ma, L., Yang, B., Gao, L., Liu, X., Ma, Y.: Tri-miprf: Tri-mip representation for efficient anti-aliasing neural radiance fields. In: ICCV (2023)
22. Huffman, D.A.: A method for the construction of minimum-redundancy codes. Proceedings of the IRE **40**(9), 1098–1101 (1952)
23. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics (ToG) **42**(4), 1–14 (2023)
24. Knapitsch, A., *et al.*: Tanks and temples: benchmarking large-scale scene reconstruction. ToG (2017)
25. Lee, J.C., Rho, D., Sun, X., Ko, J.H., Park, E.: Compact 3d gaussian representation for radiance field. arXiv preprint arXiv:2311.13681 (2023)
26. 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)
27. Mahmoud, O., Ladune, T., Gendrin, M.: Cawa-nerf: Instant learning of compression-aware nerf features. arXiv preprint arXiv:2310.14695 (2023)
28. Meagher, D.: Geometric modeling using octree encoding. Computer graphics and image processing **19**(2), 129–147 (1982)
29. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
30. Morgenstern, W., Barthel, F., Hilsmann, A., Eisert, P.: Compact 3d scene representation via self-organizing gaussian grids. arXiv preprint arXiv:2312.13299 (2023)
31. 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)

32. Navaneet, K., Meibodi, K.P., Koohpayegani, S.A., Pirsiavash, H.: Compact3d: Compressing gaussian splat radiance field models with vector quantization. arXiv preprint arXiv:2311.18159 (2023)
33. Niedermayr, S., Stumpfegger, J., Westermann, R.: Compressed 3d gaussian splatting for accelerated novel view synthesis. In: CVPR (2024)
34. 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) (May 2024), https://repo-sam.inria.fr/fungraph/reduced-3dgs/
35. Pranckevičius, A.: https://aras-p.info/blog/2023/09/13/Making-Gaussian-Splats-smaller/ (2023), accessed: 2023-10-28
36. Pranckevičius, A.: https://aras-p.info/blog/2023/09/27/Making-Gaussian-Splats-more-smaller/ (2023), accessed: 2023-10-28
37. Qin, M., Li, W., Zhou, J., Wang, H., Pfister, H.: Langsplat: 3d language gaussian splatting. arXiv preprint arXiv:2312.16084 (2023)
38. Reiser, C., Szeliski, R., Verbin, D., Srinivasan, P.P., Mildenhall, B., Geiger, A., Barron, J.T., Hedman, P.: MERF: memory-efficient radiance fields for real-time view synthesis in unbounded scenes. ACM Trans. Graph. **42**(4), 89:1–89:12 (2023). https://doi.org/10.1145/3592426
39. Rho, D., Lee, B., Nam, S., Lee, J.C., Ko, J.H., Park, E.: Masked wavelet representation for compact neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 20680–20690 (June 2023)
40. Richardson, I.E.: H. 264 and MPEG-4 video compression: video coding for next-generation multimedia. John Wiley & Sons (2004)
41. Schwarz, S., Preda, M., Baroncini, V., Budagavi, M., Cesar, P., Chou, P.A., Cohen, R.A., Krivokuća, M., Lasserre, S., Li, Z., et al.: Emerging mpeg standards for point cloud compression. IEEE Journal on Emerging and Selected Topics in Circuits and Systems **9**(1), 133–148 (2018)
42. Sculley, D.: Web-scale k-means clustering. In: Proceedings of the 19th International Conference on World Wide Web. p. 1177–1178. Association for Computing Machinery, New York, USA (2010). https://doi.org/10.1145/1772690.1772862
43. Shin, S., Park, J.: Binary radiance fields. arXiv preprint arXiv:2306.07581 (2023)
44. Song, R., Fu, C., Liu, S., Li, G.: Efficient hierarchical entropy model for learned point cloud compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14368–14377 (2023)
45. Sun, C., Sun, M., Chen, H.T.: Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5459–5469 (2022)
46. Takikawa, T., Evans, A., Tremblay, J., Müller, T., McGuire, M., Jacobson, A., Fidler, S.: Variable bitrate neural fields. In: Nandigjav, M., Mitra, N.J., Hertzmann, A. (eds.) SIGGRAPH: Special Interest Group on Computer Graphics and Interactive Techniques Conference, Vancouver, BC, Canada. pp. 41:1–41:9. ACM (2022). https://doi.org/10.1145/3528233.3530727
47. Tang, C., Ouyang, K., Wang, Z., Zhu, Y., Wang, Y., Ji, W., Zhu, W.: Mixed-precision neural network quantization via learned layer-wise importance. In: European Conference on Computer Vision (2022)
48. Wang, L., Hu, Q., He, Q., Wang, Z., Yu, J., Tuytelaars, T., Xu, L., Wu, M.: Neural residual radiance fields for streamably free-viewpoint videos. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 76–87 (2023)

49. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing **13**(4), 600–612 (2004)
50. Wei, X., Zhang, Y., Li, Y., Zhang, X., Gong, R., Guo, J., Liu, X.: Outlier suppression+: Accurate quantization of large language models by equivalent and effective shifting and scaling. In: Bouamor, H., Pino, J., Bali, K. (eds.) Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. pp. 1648–1665. Association for Computational Linguistics, Singapore (December 2023). https://doi.org/10.18653/v1/2023.emnlp-main.102, https://aclanthology.org/2023.emnlp-main.102
51. Weinberger, M.J., Seroussi, G., Sapiro, G.: The loco-i lossless image compression algorithm: Principles and standardization into jpeg-ls. IEEE Transactions on Image processing **9**(8), 1309–1324 (2000)
52. Witten, I.H., Neal, R.M., Cleary, J.G.: Arithmetic coding for data compression. Communications of the ACM **30**(6), 520–540 (1987)
53. Xu, Q., Xu, Z., Philip, J., Bi, S., Shu, Z., Sunkavalli, K., Neumann, U.: Point-nerf: Point-based neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5438–5448 (2022)
54. Yifan, W., Serena, F., Wu, S., Öztireli, C., Sorkine-Hornung, O.: Differentiable surface splatting for point-based geometry processing. ACM Transactions on Graphics (TOG) **38**(6), 1–14 (2019)
55. Zhang, C., Florencio, D., Loop, C.: Point cloud attribute compression with graph transform. In: 2014 IEEE International Conference on Image Processing (ICIP). pp. 2066–2070. IEEE (2014)
56. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018)
57. Zhao, T., Chen, J., Leng, C., Cheng, J.: Tinynerf: Towards 100 x compression of voxel radiance fields. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 3588–3596 (2023)
58. Zhao, Y., Lin, C.Y., Zhu, K., Ye, Z., Chen, L., Zheng, S., Ceze, L., Krishnamurthy, A., Chen, T., Kasikci, B.: Atom: Low-bit quantization for efficient and accurate llm serving. In: Gibbons, P., Pekhimenko, G., Sa, C.D. (eds.) Proceedings of Machine Learning and Systems. vol. 6, pp. 196–209 (2024), https://proceedings.mlsys.org/paper_files/paper/2024/file/5edb57c05c81d04beb716ef1d542fe9e-Paper-Conference.pdf
59. Ziv, J., Lempel, A.: A universal algorithm for sequential data compression. IEEE Transactions on information theory **23**(3), 337–343 (1977)
60. Ziv, J., Lempel, A.: Compression of individual sequences via variable-rate coding. IEEE transactions on Information Theory **24**(5), 530–536 (1978)
61. Zwicker, M., Pfister, H., van Baar, J., Gross, M.: Ewa volume splatting. In: Proceedings Visualization, 2001. VIS '01. pp. 29–538 (2001). https://doi.org/10.1109/VISUAL.2001.964490
62. Zwicker, M., Pfister, H., Van Baar, J., Gross, M.: Surface splatting. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques. pp. 371–378 (2001)
63. Červený, J.: https://gsplat.tech (2023), accessed: 2023-10-28