HAC: Hash-grid Assisted Context for 3D Gaussian Splatting Compression

Yihang Chen[1,2], Qianyi Wu[2], Weiyao Lin[1⋆], Mehrtash Harandi[2], and Jianfei Cai[2]

[1] Shanghai Jiao Tong University
[2] Monash University
{yhchen.ee, wylin}@sjtu.edu.cn,
{qianyi.wu, mehrtash.harandi, jianfei.cai}@monash.edu

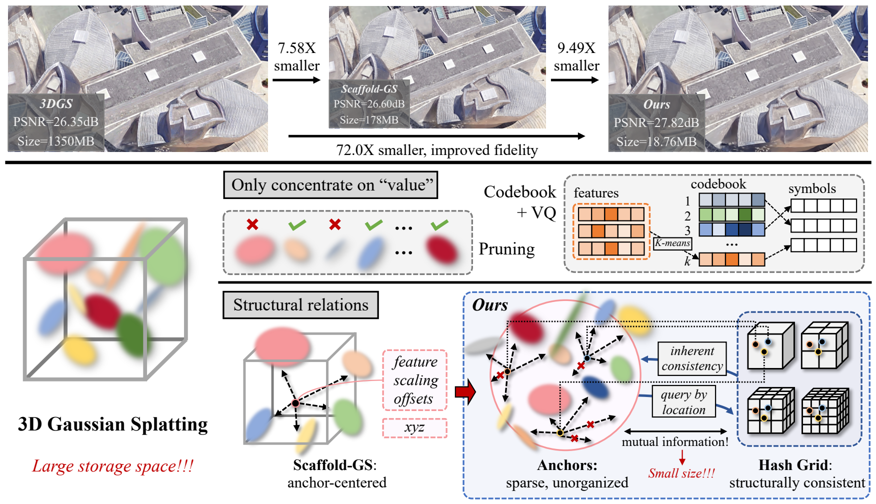


Fig. 1: Top: A toy example where our method makes the size of the vanilla 3D Gaussian splitting (3DGS) model 72× smaller (or 9.49× smaller compared to the SoTA Scaffold-GS [27]), with similar or better fidelity. **Bottom**: Most existing 3DGS compression methods concentrate solely on parameter "values" using pruning or vector quantization to reduce size, ignoring the structure relations among Gaussians. Scaffold-GS [27] introduces anchors to cluster and neural-predict the associated Gaussians while treating each anchor point independently. Our core idea is to further exploit the inherent consistencies of anchors via a structured hash grid for a more compact 3DGS representation.

Abstract. 3D Gaussian Splatting (3DGS) has emerged as a promising framework for novel view synthesis, boasting rapid rendering speed with high fidelity. However, the substantial Gaussians and their associated attributes necessitate effective compression techniques. Nevertheless, the

⋆ Corresponding author

sparse and unorganized nature of the point cloud of Gaussians (or anchors in our paper) presents challenges for compression. To address this, we make use of the relations between the unorganized anchors and the structured hash grid, leveraging their mutual information for context modeling, and propose a Hash-grid Assisted Context (HAC) framework for highly compact 3DGS representation. Our approach introduces a binary hash grid to establish continuous spatial consistencies, allowing us to unveil the inherent spatial relations of anchors through a carefully designed context model. To facilitate entropy coding, we utilize Gaussian distributions to accurately estimate the probability of each quantized attribute, where an adaptive quantization module is proposed to enable high-precision quantization of these attributes for improved fidelity restoration. Additionally, we incorporate an adaptive masking strategy to eliminate invalid Gaussians and anchors. Importantly, our work is the pioneer to explore context-based compression for 3DGS representation, resulting in a remarkable size reduction of over $75\times$ compared to vanilla 3DGS, while simultaneously improving fidelity, and achieving over $11\times$ size reduction over SoTA 3DGS compression approach Scaffold-GS. Our code is available here.

Keywords: 3D Gaussian Splatting · Compression · Context Models

1 Introduction

Over the past few years, significant advancements have been made in 3D scene representations for novel view synthesis. Neural Radiance Field (NeRF) [28] proposes rendering colors by accumulating RGB values along sampling rays using an implicit Multilayer Perceptron (MLP), aiming at reconstructing photo-realistic images. However, the extensive sampling of ray points has been a bottleneck, affecting both the speed of training and rendering. Recent advances of NeRF [5, 13, 30] introduce feature grids to enhance the rendering process, facilitating faster rendering speeds by reducing the MLP size. Despite the improvement, these approaches still suffer from relatively slow rendering speeds due to frequent ray point sampling.

In this context, very recently, a new paradigm of 3D representation, 3D Gaussian Splatting (3DGS) [19], emerged. 3DGS introduces learnable Gaussians to directly represent 3D space explicitly. These Gaussians, initialized from Structure-from-Motion (SfM) [35] and endowed with learnable shape and appearance parameters, can be directly splatted onto 2D planes for rapid and differentiable rendering within imperceptible intervals using tile-based rasterization [21]. As such, the time-consuming volume rendering used in NeRF can be completely removed. The advantages of rapid differentiable rendering with high photo-realistic fidelity have stimulated the fast and widespread adoption of 3DGS in the field.

However, 3DGS is not the ultimate solution. One major drawback is that it requires a considerable number of 3D Gaussians to well represent a large-scale scene (*e.g.*, at the scale of millions of Gaussians for city-scale scenes) and needs a large storage space (*e.g.*, a few GigaBytes (GB)) to store the associated

Gaussian attributes for each scene [42]. This motivates us to investigate effective compression techniques for 3DGS.

Due to their sparse and unorganized nature, compressing 3D Gaussians is challenging and difficult [6, 12]. Therefore, most existing 3DGS compression approaches focus solely on parameter "values" but overlook their structural relations. For example, as illustrated in Fig. 1 middle, parameter pruning can be used to mask out the Gaussians whose parameter values are below a certain threshold [11, 22]. Another straightforward technique is to apply vector quantization to cluster parameters with similar "values". Such an approach enables the direct compression of parameters by only retaining more representative ones while maintaining reconstruction fidelity [11, 22, 31, 32]. Nevertheless, solely concentrating on "values" fails to eliminate structural redundancies, which are pivotal for compact representations. To exploit such spatial relations of Gaussians, Scaffold-GS [27] introduces anchors to cluster related nearby 3D Gaussians and neural-predict their attributes from the anchors' attributes, resulting in significant storage savings. Despite the improvement, Scaffold-GS still treats each anchor independently, and there are still substantial anchors that are sparse, unorganized, and hard to compress, due to their point-cloud nature.

To further push the boundary of 3DGS compression, we draw inspiration from the NeRF series [28], contemplating the idea of representing 3D space using well-organized feature grids [5, 30]. We pose the question: *Is there inherent relations between the attributes of unorganized anchors in Scaffold-GS and the structured feature grids?* Our answer is affirmative since we observe large mutual information between anchor attributes and the hash grid features. Based on this observation, we propose a Hash-grid Assisted Context (HAC) framework, where our core idea is to jointly learn structured compact hash grid (binarized for each hash parameter) and use it for context modeling of anchor attributes. Specifically, with Scaffold-GS [27] as our base model, for each anchor, we query the hash grid by the anchor location to obtain an interpolated hash feature, which is then used to predict the value distributions of anchor attributes, facilitating entropy coding such as Arithmetic Coding (AE) [41] for a highly compact representation of the model. Note that we employ Scaffold-GS as our base model as its anchor-centered design provides a good foundation to establish relations with these interpolated hash features. Furthermore, we introduce an Adaptive Quantization Module (AQM), which dynamically adjusts different quantization step sizes for different anchor attributes for retaining of their original information. Learnable masks are also employed to mask out invalid Gaussians and anchors, further enhancing the compression ratio. Our main contributions can be summarized as follows:

1. To our knowledge, we are the first to model contexts for 3DGS compression, *i.e.*, using a structured hash grid to exploit the inherent consistencies among unorganized 3D Gaussians (or anchors in Scaffold-GS).
2. To facilitate efficient entropy encoding of anchor attributes, we propose to use the interpolated hash feature to neural-predict the value distribution of anchor attributes as well as neural-predicting quantization step refinement

with AQM. We also employ learnable masks to prune out ineffective Gaussians and anchors.
3. Extensive experiments on five datasets demonstrate the effectiveness of our HAC framework and each technical component. We achieve a compression ratio of 11× over our base model Scaffold-GS and 75× over the vanilla 3DGS model when averaged over all datasets, while with comparable or even improved fidelity.

2 Related Work

Neural Radiance Field and its compression. The emergence of Neural Radiance Field (NeRF) [28] has significantly advanced novel view synthesis by employing a single learnable implicit MLP to generate arbitrary views of 3D scenes through α-composed accumulation of RGB values along a ray. However, the dense querying of sampling points and the utilization of a large MLP hinder real-time rendering. To address this problem, subsequent approaches such as Instant-NGP [30], TensoRF [5], K-planes [13], and DVGO [39] adopt explicit grid-based representations to facilitate faster training and rendering by reducing the size of the MLP, which however comes at the cost of increased storage space.

To mitigate the storage increase, compression techniques focusing on reducing the size of explicit representations have been developed, which can be categorized into either "value"-based or structural-relation-based approaches. The former category includes pruning [10, 25], codebooks [25, 26], and methods like quantization or entropy constraint employed in BiRF [37] and SHACIRA [15]. On the other hand, the latter category explores structural relations via wavelet decomposition [34], rank-residual decomposition [40], or spatial prediction [38] to eliminate spatial redundancy, thanks to the well-structured characteristics of these feature grids. CNC [7] provides a solid proof of concept by sufficiently utilizing such structural information, achieving remarkable RD performance gain.
3D Gaussian Splatting and its compression. 3DGS [19] has innovatively addressed the challenge of slow training and rendering in NeRF while maintaining high-fidelity quality by representing 3D scenes with 3D Gaussians endowed with learnable shape and appearance attributes. By adopting differentiable splatting and tile-based rasterization [21], 3D Gaussians are optimized during training to best fit their local 3D regions. Despite its advantages, the substantial Gaussians and their associated attributes necessitate effective compression techniques.

Unlike NeRF-based feature grids, 3D Gaussians in 3DGS are sparse and unorganized, presenting significant challenges for establishing structural relations. Consequently, compression approaches have primarily focused solely on the "value" of model parameters, employing techniques such as pruning [11, 22], codebooks [11, 22, 31, 32], and entropy constraints [14]. To our knowledge, Scaffold-GS [27] and Morgenstern *et al.* [29] have explored the relations of Gaussians. In [27], authors introduce anchor-centered features to achieve reduced parameter numbers, while in [29] dimension collapsing is considered to compress Gaussians in an ordered 2D space. However, their investigation of spatial redundancy remains insufficient.

In this paper, we emphasize leveraging such structural relations for compression is crucial. For instance, approaches in image compression [8,16,17] and video compression [23,24,36] have demonstrated the effectiveness of eliminating structural redundancy by excavating spatial and temporal relations, thanks to their well-organized data structure. Motivated by this, with Scaffold-GS as our base model, we introduce a well-structured hash grid as context to model the inherent consistencies of the sparse and unorganized anchors, achieving much more compact 3DGS representation.

3 Methods

In Fig. 2, we conceptualize our HAC framework. In particular, HAC is based on the baseline Scaffold-GS [27] (Fig. 2 top), which introduces anchors with their attributes $\mathcal{A}$ (feature, scaling and offsets) to cluster and neural-predict 3D Gaussian attributes (opacity, RGB, scale, and quaternion). At the core of our HAC, we propose to jointly learn structured compact hash grid (binarized for each parameter) that can be queried at any anchor location to obtain the interpolated hash feature $\boldsymbol{f}^h$ (Fig. 2 middle). Instead of directly substituting anchor feature, $\boldsymbol{f}^h$ is used as context to predict the value distributions of anchor attributes, which is essential for the subsequent entropy coding, *i.e.*, Arithmetic Coding (AE). Our context model (Fig. 2 bottom) is a simple MLP that takes $\boldsymbol{f}^h$ as input and outputs $\boldsymbol{r}$ for the adaptive quantization module (AQM) (quantize anchor attribute values into a finite set) and the Gaussian parameters ($\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$) for modeling the value distributions of anchor attributes, from which we can compute the probability of each quantized attribute value for AE. Note that we draw two MLPs ($\mathrm{MLP_q}$ and $\mathrm{MLP_c}$) in Fig. 2 for easy explanation but they actually share the same MLP layers with outputs at different dimensions. Besides, an adaptive offset masking module (Fig. 2 top-left) is adopted to prune redundant Gaussians and anchors. In the following, we first introduce the background and then delve into the detailed technical components of our HAC.

3.1 Preliminaries

3D Gaussian Splatting (3DGS) [19] represents a 3D scene using numerous Gaussians and renders viewpoints through a differentiable splatting and tile-based rasterization. Each Gaussian is initialized from SfM and defined by a 3D covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times 3}$ and location (mean) $\boldsymbol{\mu} \in \mathbb{R}^3$,

$$G(\boldsymbol{x}) = \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right), \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^3$ is a random 3D point, and $\boldsymbol{\Sigma}$ is defined by a diagonal matrix $\boldsymbol{S} \in \mathbb{R}^{3\times 3}$ representing scaling and rotation matrix $\boldsymbol{R} \in \mathbb{R}^{3\times 3}$ to guarantee its positive semi-definite characteristics, such that $\boldsymbol{\Sigma} = \boldsymbol{R}\boldsymbol{S}\boldsymbol{S}^\top \boldsymbol{R}^\top$. To render an image from a random viewpoint, 3D Gaussians are first splatted to 2D, and render the pixel value $\boldsymbol{C} \in \mathbb{R}^3$ using α-composed blending,

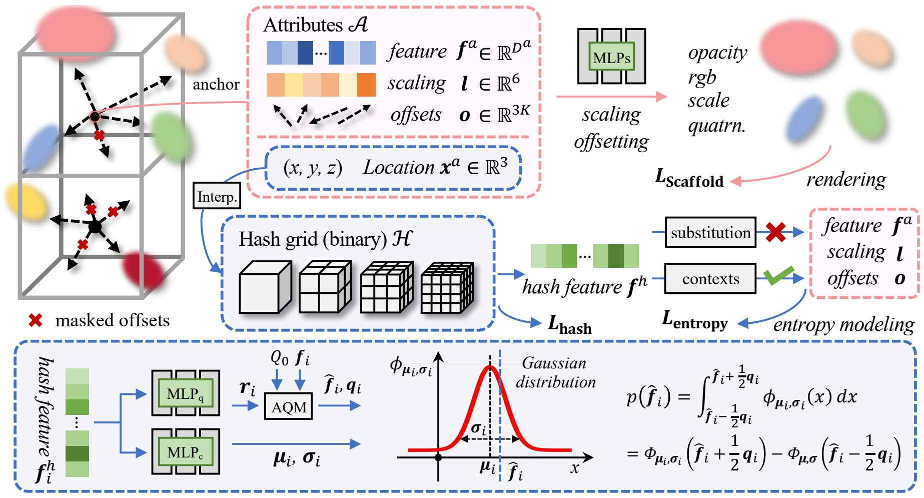

Attributes $\mathcal{A}$
feature $f^a \in \mathbb{R}^{D^a}$
scaling $l \in \mathbb{R}^6$
offsets $o \in \mathbb{R}^{3K}$
anchor
(x, y, z) Location $x^a \in \mathbb{R}^3$
Interp.
Hash grid (binary) $\mathcal{H}$
masked offsets
MLPs
scaling
offsetting
opacity
rgb
scale
quatrn.
rendering
L_{Scaffold}
hash feature f^h
substitution
contexts
L_{hash}
L_{entropy}
entropy modeling
hash feature f_i^h
MLP$_q$
MLP$_c$
Q_0 f_i
r_i
AQM
$\hat{f}_i, q_i$
μ_i, σ_i
ϕ_{μ_i,σ_i}
Gaussian distribution
σ_i
μ_i
$\hat{f}_i$
x
$p(\hat{f}_i) = \int_{\hat{f}_i-\frac{1}{2}q_i}^{\hat{f}_i+\frac{1}{2}q_i} \phi_{\mu_i,\sigma_i}(x)\,dx$
$= \Phi_{\mu_i,\sigma_i}\left(\hat{f}_i+\frac{1}{2}q_i\right) - \Phi_{\mu,\sigma}\left(\hat{f}_i-\frac{1}{2}q_i\right)$

Fig. 2: Overview of our HAC framework. It is based on Scaffold-GS [27] (**top**), which introduces anchors with their attributes to neural-predict 3D Gaussian attributes. **Middle**: Our HAC framework jointly learns structured compact hash grid (binarized for each parameter) that can be queried at any anchor location to obtain the interpolated hash feature $\boldsymbol{f}^h$. Instead of direct substitution, $\boldsymbol{f}^h$ is used as context to predict the value distributions of anchor attributes, which is essential for the subsequent entropy coding. **Bottom**: Our proposed context models take $\boldsymbol{f}^h$ as input and outputs $\boldsymbol{r}$ for the AQM (quantize anchor attribute values into a finite set) and the parameters ($\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$) to model the value distributions of anchor attributes.

$$\boldsymbol{C} = \sum_{i\in I} \boldsymbol{c}_i \alpha_i \prod_{j=1}^{i-1} (1-\alpha_j) \tag{2}$$

where $\alpha \in \mathbb{R}^1$ measures the opacity of each Gaussian after 2D projection, $\boldsymbol{c} \in \mathbb{R}^3$ is view-dependent color modeled by Spherical Harmonic (SH) coefficients, and I is the number of sorted Gaussians contributing to the rendering.

Scaffold-GS [27] adheres to the framework of 3DGS and introduces a more storage-friendly and fidelity-satisfying anchor-based approach. It utilizes anchors to cluster Gaussians and deduce their attributes from the attributes of attached anchors through MLPs, rather than directly storing them. Specifically, each anchor consists of a location $\boldsymbol{x}^a \in \mathbb{R}^3$ and anchor attributes $\mathcal{A} = \{\boldsymbol{f}^a \in \mathbb{R}^{D^a}, \boldsymbol{l} \in \mathbb{R}^6, \boldsymbol{o} \in \mathbb{R}^{3K}\}$, where each component represents anchor feature, scaling and offsets, respectively. During rendering, $\boldsymbol{f}^a$ is inputted into MLPs to generate attributes for Gaussians, whose locations are determined by adding $\boldsymbol{x}^a$ and $\boldsymbol{o}$, where $\boldsymbol{l}$ is utilized to regularize both locations and shapes of the Gaussians. While Scaffold-GS has demonstrated effectiveness via this anchor-centered design, we contend there is still significant redundancy among inherent consistencies of anchors that we can fully exploit for a more compact 3DGS representation.

3.2 Bridging Anchors and Hash Grid

We begin the analysis by intuitively considering neighboring Gaussians share similar parameters inferred from anchor attributes. This initial perception leads us to assume anchor attributes are also consistent in space. Our main idea is to leverage the well-structured hash grid to unveil the inherent spatial consistencies of the unorganized anchors. Please also refer to experiments in Fig. 7 to observe this consistency. To verify mutual information between the hash grid and anchors, we first explore substituting anchor features $\boldsymbol{f}^a$ with hash features $\boldsymbol{f}^h$ that are acquired by interpolation using the anchor location $\boldsymbol{x}^a$ on the hash grid $\mathcal{H}$, defined as $\boldsymbol{f}^h := \text{Interp}(\boldsymbol{x}^a, \mathcal{H})$. Here, $\mathcal{H} = \{\boldsymbol{\theta}_i^l \in \mathbb{R}^{D^h} | i = 1, \dots, T^l | l = 1, \dots, L\}$ represents the hash gird, where D^h is the dimension of vector $\boldsymbol{\theta}_i^l$, T^l is the table size of the grid for level l, and L is the number of levels. We conduct a preliminary experiment on the Synthetic-NeRF dataset [28] to assess its performance, as shown in the right panel of Fig. 3. Direct substitution using hash features appears to yield inferior fidelity and introduces drawbacks such as unstable training (due to its impact on anchor spawning processes) and decreased testing FPS (owing to the extra interpolation operation). These results may further degrade if $\boldsymbol{l}$ and $\boldsymbol{o}$ are also substituted for a more compact model. Nonetheless, we find the fidelity degradation remains moderate, suggesting the existence of rich mutual information between $\boldsymbol{f}^h$ and $\boldsymbol{f}^a$. This prompts us to ask: *Can we exploit such mutual relation and use the compact hash features to model the context of anchor attributes $\mathcal{A}$?* This leads to the context modeling as a conditional probability:

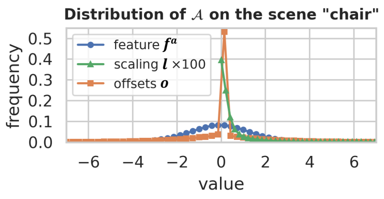
$$p(\mathcal{A}, \boldsymbol{x}^a, \mathcal{H}) = p(\mathcal{A}|\boldsymbol{x}^a, \mathcal{H}) \times p(\boldsymbol{x}^a, \mathcal{H}) \sim p(\mathcal{A}|\boldsymbol{f}^h) \times p(\mathcal{H}) \tag{3}$$

where $\boldsymbol{x}^a$ is omitted in the last term as we assume the independence of $\boldsymbol{x}^a$ and $\mathcal{H}$ (it can be anywhere), making $p(\mathcal{H}|\boldsymbol{x}^a) \sim p(\mathcal{H})$, and do not employ entropy constraints to $\boldsymbol{x}^a$. According to information theory [9], a higher probability corresponds to lower uncertainty (entropy) and fewer bits consumption. Thus, the large mutual information between $\mathcal{A}$ and $\boldsymbol{f}^h$ ensures a large $p(\mathcal{A}|\boldsymbol{f}^h)$. Our goal is to devise a solution to effectively leverage this relationship. Furthermore, $p(\mathcal{H})$ signifies that the size of the hash grid itself should also be compressed, which can be done by adopting the existing solution for Instant-NGP compression [7].

We underscore the significance of this conditional probability based approach since it ensures both rendering speed and fidelity upper-bound unaffected as it only utilizes hash features to estimate the entropy of anchor attributes for entropy coding but does not modify the original Scaffold-GS structure. In the following subsections, we delve into the technical details of our context models.

3.3 HAC: Hash-Grid Assisted Context Framework

The principle objective of HAC is to minimize the entropy of anchor attributes $\mathcal{A}$ with the assistance of hash feature $\boldsymbol{f}^h$ (*i.e.*, maximize $p(\mathcal{A}|\boldsymbol{f}^h)$), facilitating bit reduction when encoding anchor attributes using entropy coding like AE [41]. As shown in Fig. 2, anchor locations $\boldsymbol{x}^a$ are firstly inputted into the hash grid for interpolation, the obtained $\boldsymbol{f}^h$ are then employed as context for $\mathcal{A}$.



Synthetic-NeRF [28]	PSNR↑	SSIM↑	LPIPS↓
3DGS [19]	33.80	0.970	0.031
Scaffold-GS [27]	33.41	0.966	0.035
Substituting $\boldsymbol{f}^a$ with $\boldsymbol{f}^h$	32.85	0.963	0.041

Fig. 3: Left chart: Statistical analysis of the value distributions of $\mathcal{A}$ on the scene "chair" of the Synthetic-NeRF dataset [28]. All three components $\{\boldsymbol{f}^a, \boldsymbol{l}, \boldsymbol{o}\}$ exhibit statistical Gaussian distributions. Note that the values of $\boldsymbol{l}$ are scaled by a factor of 100 for better visualization. **Right table**: Experimental results of directly substituting anchor feature $\boldsymbol{f}^a$ with hash feature $\boldsymbol{f}^h$ on this dataset.

Adaptive Quantization Module. To facilitate entropy coding, values of $\mathcal{A}$ must be quantized to a finite set. Our empirical studies reveal that binarization, as that in BiRF [37], is unsuitable for $\mathcal{A}$ as it fails to preserve sufficient information. Thus, we opt for rounding them to maintain their comprehensive features. To ensure backpropagation, we utilize the "adding noise" operation during training and "rounding" during testing, as described in [1].

Nevertheless, the conventional rounding is essentially a quantization with a step size of "1", which is inappropriate for the scaling $\boldsymbol{l}$ and the offset $\boldsymbol{o}$, since they are usually decimal values. To address this, we further introduce an Adaptive Quantization Module (AQM), which adaptively determines quantization steps. In particular, for the ith anchor $\boldsymbol{x}_i^a$, we denote $\boldsymbol{f}_i$ as any of its $\mathcal{A}_i$'s components: $\boldsymbol{f}_i \in \{\boldsymbol{f}_i^a, \boldsymbol{l}_i, \boldsymbol{o}_i\} \in \mathbb{R}^D$, where $D \in \{D^a, 6, 3K\}$ is its respective dimension. The quantization can be written as,

$$\begin{aligned} \hat{\boldsymbol{f}}_i &= \boldsymbol{f}_i + \mathcal{U}\left(-\frac{1}{2}, \frac{1}{2}\right) \times \boldsymbol{q}_i, && \text{for training} \\ &= \text{Round}(\boldsymbol{f}_i/\boldsymbol{q}_i) \times \boldsymbol{q}_i, && \text{for testing} \end{aligned} \tag{4}$$

where

$$\begin{aligned} \boldsymbol{q}_i &= Q_0 \times (1 + \text{Tanh}(\boldsymbol{r}_i)) \\ \boldsymbol{r}_i &= \text{MLP}_\text{q}\left(\boldsymbol{f}_i^h\right). \end{aligned} \tag{5}$$

We use a simple MLP-based context model MLP_q to predict from hash feature $\boldsymbol{f}_i^h$ a refinement $\boldsymbol{r}_i \in \mathbb{R}^1$, which is used to adjust the predefined quantization step size Q_0. Note that Q_0 varies for $\boldsymbol{f}^a$, $\boldsymbol{l}$, and $\boldsymbol{o}$. Eq. (5) essentially restricts the quantization step size $\boldsymbol{q}_i \in \mathbb{R}^1$ to be chosen within $(0, 2Q_0)$, enabling $\hat{\boldsymbol{f}}_i$ to closely resemble the original characteristics of $\boldsymbol{f}_i$, maintaining a high fidelity.

Gaussian Distribution Modeling. To measure the bit consumption of $\hat{\boldsymbol{f}}_i$ during training, its probability needs to be estimated in a differentiable manner. As shown in Fig. 3 left, all three components of anchor attributes $\mathcal{A}$ exhibit statistical tendencies of Gaussian distributions, where $\boldsymbol{l}$ displays a single-sided pattern due to Sigmoid activation[1]. This observation establishes a lower bound

[1] We define $\boldsymbol{l}$ as the one *after* Sigmoid activation, which is slightly different from [27].

for probability prediction when all $\hat{\boldsymbol{f}}_i$s in $\mathcal{A}$ are estimated using the respective μ and σ of the statistical Gaussian Distribution of $\boldsymbol{f}^a$, $\boldsymbol{l}$ and $\boldsymbol{o}$. Nevertheless, employing a single set of μ and σ for all attributes may lack accuracy. Therefore, we assume anchor attributes $\mathcal{A}$'s values independent, and construct their respective Gaussian distributions, where their individual $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are estimated by a simple MLP-based context model $\mathrm{MLP_c}$ from $\boldsymbol{f}^h$. Specifically, for the ith anchor and its quantized anchor attribute vector $\hat{\boldsymbol{f}}_i$, with the estimated $\boldsymbol{\mu}_i \in \mathbb{R}^D$ and $\boldsymbol{\sigma}_i \in \mathbb{R}^D$, we can compute the probability of $\hat{\boldsymbol{f}}_i$ as,

$$\begin{aligned} p(\hat{\boldsymbol{f}}_i) &= \int_{\hat{\boldsymbol{f}}_i - \frac{1}{2}\boldsymbol{q}_i}^{\hat{\boldsymbol{f}}_i + \frac{1}{2}\boldsymbol{q}_i} \phi_{\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i}(x)\, dx \\ &= \Phi_{\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i}\left(\hat{\boldsymbol{f}}_i + \frac{1}{2}\boldsymbol{q}_i\right) - \Phi_{\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i}\left(\hat{\boldsymbol{f}}_i - \frac{1}{2}\boldsymbol{q}_i\right) \\ \boldsymbol{\mu}_i, \boldsymbol{\sigma}_i &= \mathrm{MLP_c}\left(\boldsymbol{f}_i^h\right). \end{aligned} \tag{6}$$

where ϕ and Φ represent the probability density function and the cumulative distribution function, respectively. Consequently, we define an entropy loss as the summation of bit consumption over all $\hat{\boldsymbol{f}}_i$s:

$$L_{\text{entropy}} = \sum_{\boldsymbol{f} \in \{\boldsymbol{f}^a, \boldsymbol{l}, \boldsymbol{o}\}} \sum_{i=1}^{N} \sum_{j=1}^{D} \left(-\log_2 p(\hat{f}_{i,j})\right) \tag{7}$$

N is the number of anchors and $\hat{f}_{i,j}$ is $\hat{\boldsymbol{f}}_i$'s j-th dimension value. Minimizing the entropy loss encourages a high probability estimation for $p(\hat{\boldsymbol{f}}_i)$, which in turn encourages accurate $\boldsymbol{\mu}_i$ and small $\boldsymbol{\sigma}_i$, guiding the learning of the context model.
Adaptive Offset Masking. From Fig. 3 left, we can also see that $\boldsymbol{o}$ exhibits an impulse at zero, suggesting the occurrence of substantial unnecessary Gaussians. Thus, we employ the technique introduced by Lee *et al.* [22] to prune invalid $\boldsymbol{o}$ by utilizing straight-through [3] estimated binary masks. Specifically, we apply the same marking loss L_m in [22] to encourage masking as many Gaussians as possible. This process effectively masks out invalid offsets and saves storage space directly. Additionally, we implement anchor pruning: if all the attached $\boldsymbol{o}$ are pruned on an anchor, then this anchor no longer contributes to rendering and should be pruned entirely (including its $\boldsymbol{x}^a$ and $\mathcal{A}$).
Hash Grid Compression. As shown in Eq. (3), the size of the hash grid $\mathcal{H}$ also significantly influences the final storage size. To this end, we binarize the hash table to $\{-1, +1\}$ using straight-through estimation (STE) [37] and calculate the occurrence frequency h_f [7] of the symbol "+1" to estimate its bit consumption:

$$L_{\text{hash}} = M_+ \times (-\log_2(h_f)) + M_- \times (-\log_2(1 - h_f)) \tag{8}$$

where M_+ and M_- are total numbers of "+1" and "−1" in the hash grid.

3.4 Training and Coding Process

During training, we incorporate both the rendering fidelity loss and the entropy loss to ensure the model improves rendering quality while controlling total bitrate consumption in a differentiable manner. Our overall loss is

$$Loss = L_{\text{Scaffold}} + \lambda_e \frac{1}{N(D^a + 6 + 3K)} (L_{\text{entropy}} + L_{\text{hash}}) + \lambda_m L_m. \qquad (9)$$

Here, L_{Scaffold} represents the rendering loss as defined in [27], which includes two fidelity penalty loss terms and one regularization term for the scaling $\boldsymbol{l}$. The second part in Eq. (9) is the estimated controllable bit consumption, including the estimated bits L_{entropy} for anchor attributes and L_{hash} for the hash grid. The last term L_m in Eq. (9) is the masking loss adopted from [22] to regularize the adaptive offset masking module. λ_e and λ_m are trade-off hyperparameters used to balance the loss components. Note that we incorporate different techniques or loss items at different iterations to stabilize the training process. Please refer to the supplementary Sec.A for more details.

For the encoding/decoding process, the binary hash grid $\mathcal{H}$ is first encoded/ decoded using AE with h_f. Then, hash feature $\boldsymbol{f}^h$ is obtained through interpolation based on $\mathcal{H}$ and $\boldsymbol{x}^a$. Once $\boldsymbol{f}^h$ is acquired, the context models MLP_q and MLP_c are then employed to estimate quantization refinement term $\boldsymbol{r}$ and parameters of the Gaussian Distribution (*i.e.*, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$) to derive the probability $p(\hat{\boldsymbol{f}})$ for entropy encoding/decoding with AE.

4 Experiments

In this section, we first present our HAC framework's implementation details and then conduct evaluation experiments to compare with existing 3DGS compression approaches. Additionally, we include ablation studies to demonstrate the effectiveness of each technical component of our method. Finally, we visualize the bit allocation map for better understanding.

4.1 Implementation Details

We implement our HAC based on the Scaffold-GS repository [27] using the PyTorch framework [33] and train the model on a single NVIDIA RTX 4090 GPU. We increase the dimension of the Scaffold-GS anchor feature $\boldsymbol{f}^a$ (*i.e.*, D^a) to 50, and disable its feature bank as we found it may lead to unstable training. For the hash grid $\mathcal{H}$, we utilize a mixed 3D-2D structured binary hash grid, with 12 levels of 3D embeddings ranging from 16 to 512 resolutions, and 4 levels of 2D embeddings ranging from 128 to 1024 resolutions. The maximum hash table sizes are 2^{13} and 2^{15} for the 3D and 2D grids, respectively, both with a feature dimension of $D^h = 4$. We set λ_m to $5e-4$, and change λ_e from $5e-4$ to $4e-3$ for variable bitrates. We set Q_0 as 1, 0.001 and 0.2 for $\boldsymbol{f}^a$, $\boldsymbol{l}$ and $\boldsymbol{o}$, respectively. We combine MLP_q and MLP_c to a single 3-layer MLP with ReLU activation.

4.2 Experiment Evaluation

Baselines. We compare our HAC with existing 3DGS compression approaches. Notably, [11, 22, 31, 32] mainly adopt codebook-based or parameter pruning

Table 1: Quantitative results. 3DGS [19] and Scaffold-GS [27] are two baselines. Approaches in the middle chunk are designed for 3DGS compression. For our approach, we give two results of different size and fidelity tradeoffs by adjusting λ_e. A smaller λ_e results in a larger size but improved fidelity, and vice versa. The best and 2nd best results are highlighted in red and yellow cells. The size is measured in MB.

Datasets	**Synthetic-NeRF [28]**				**Mip-NeRF360 [2]**				**Tank&Temples [20]**			
methods	psnr↑	ssim↑	lpips↓	size↓	psnr↑	ssim↑	lpips↓	size↓	psnr↑	ssim↑	lpips↓	size↓
3DGS [19]	33.80	0.970	0.031	68.46	27.49	0.813	0.222	744.7	23.69	0.844	0.178	431.0
Scaffold-GS [27]	33.41	0.966	0.035	19.36	27.50	0.806	0.252	253.9	23.96	0.853	0.177	86.50
Lee *et al.* [22]	33.33	0.968	0.034	5.54	27.08	0.798	0.247	48.80	23.32	0.831	0.201	39.43
Compressed3D [32]	32.94	0.967	0.033	3.68	26.98	0.801	0.238	28.80	23.32	0.832	0.194	17.28
EAGLES [14]	32.54	0.965	0.039	5.74	27.15	0.808	0.238	68.89	23.41	0.840	0.200	34.00
LightGaussian [11]	32.73	0.965	0.037	7.84	27.00	0.799	0.249	44.54	22.83	0.822	0.242	22.43
Morgen. *et al.* [29]	31.05	0.955	0.047	2.20	26.01	0.772	0.259	23.90	22.78	0.817	0.211	13.05
Navaneet *et al.* [31]	33.09	0.967	0.036	4.42	27.16	0.808	0.228	50.30	23.47	0.840	0.188	27.97
Ours-lowrate	33.24	0.967	0.037	1.18	27.53	0.807	0.238	15.26	24.04	0.846	0.187	8.10
Ours-highrate	33.71	0.968	0.034	1.86	27.77	0.811	0.230	21.87	24.40	0.853	0.177	11.24

Datasets	**DeepBlending [18]**				**BungeeNeRF [42]**			
methods	psnr↑	ssim↑	lpips↓	size↓	psnr↑	ssim↑	lpips↓	size↓
3DGS [19]	29.42	0.899	0.247	663.9	24.87	0.841	0.205	1616
Scaffold-GS [27]	30.21	0.906	0.254	66.00	26.62	0.865	0.241	183.0
Lee *et al.* [22]	29.79	0.901	0.258	43.21	23.36	0.788	0.251	82.60
Compressed3D [32]	29.38	0.898	0.253	25.30	24.13	0.802	0.245	55.79
EAGLES [14]	29.91	0.910	0.250	62.00	25.24	0.843	0.221	117.1
LightGaussian [11]	27.01	0.872	0.308	33.94	24.52	0.825	0.255	87.28
Morgen. *et al.* [29]	28.92	0.891	0.276	8.40	–	–	–	–
Navaneet *et al.* [31]	29.75	0.903	0.247	42.77	24.63	0.823	0.239	104.3
Ours-lowrate	29.98	0.902	0.269	4.35	26.48	0.845	0.250	18.49
Ours-highrate	30.34	0.906	0.258	6.35	27.08	0.872	0.209	29.72

strategies, while Scaffold-GS [27] explores Gaussian relations for compact representation. Additionally, EAGLES [14] and Morgenstern *et al.* [29] employ non-contextual entropy constraints and dimension collapse techniques, respectively.

Datasets. We follow Scaffold-GS to perform evaluations on multiple datasets, including the small-scale Synthetic-NeRF [28] and the four large-scale real-scene datasets: BungeeNeRF [42], DeepBlending [18], Mip-NeRF360 [2], and Tanks&Temples [20]. Note that we evaluate the entire 9 scenes from Mip-NeRF360 dataset [2]. Covering diverse scenarios, these datasets allow us to comprehensively demonstrate the effectiveness of our approach.

Metrics. To comprehensively evaluate compression Rate-Distortion (RD) performance, we calculate relative rate (size) change of our approach over others under a similar fidelity. Note that BD-rate [4] is incalculable as other methods can typically only output a single rate, while four are needed for its calculation.

Results. Quantitative results are shown in Tab. 1 and Fig. 4, the qualitative outputs are presented in Fig. 5. Please refer to the supplementary Sec. C for detailed metrics of each scene. Our HAC has demonstrated significant size reduction of over 75× compared to the vanilla 3DGS [19] with even improved fidelity. The size reduction also exceeds 11× over the base model Scaffold-GS [27]. Notably, our highest fidelity surpasses Scaffold-GS, primarily due to two factors: 1) the entropy loss effectively regularizes the model to prevent overfitting, and 2) we

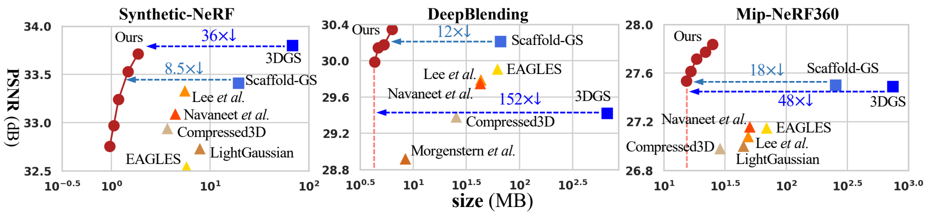


Fig. 4: RD curves for quantitative comparisons. We vary λ_e to achieve variable bitrates. Note that $\log_{10}$ scale is used for x-axis for better visualization.

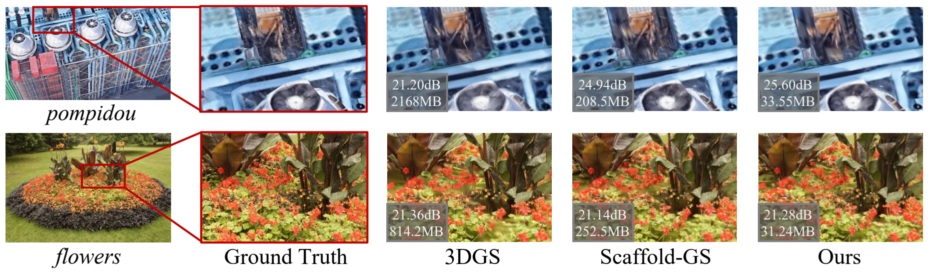


Fig. 5: Qualitative comparisons of "pompidou" from BungeeNeRF [42] and "flowers" from Mip-NeRF360 [2]. PSNR and size results are given at lower-left.

increase the dimension of the anchor feature (*i.e.*, D^a) to 50, resulting in a larger model volume. Although other compression approaches (mid chunk) can reduce the model size by primarily using pruning and codebooks, they still exhibit significant spatial redundancy. Specifically, Morgenstern [29] achieves a comparably small size, but significantly sacrifices fidelity due to the dimension collapsing.

Bitstream. Our bitstream consists of five components: anchor attributes $\mathcal{A}$ (comprising $\boldsymbol{f}^a$, $\boldsymbol{l}$ and $\boldsymbol{o}$), binary hash grid $\mathcal{H}$, offset masks, anchor locations $\boldsymbol{x}^a$ and MLPs. Among them, $\mathcal{A}$ is encoded using entropy codec AE [41] with estimated probabilities from HAC. It accounts for the dominant portion of the storage. The hash grid $\mathcal{H}$ and the masks are binary data and are encoded by AE using the respective occurrence frequency. The last two components are stored directly in 16 and 32 bits, respectively. When analyzing the bit allocation of each component, they are 14.90MB (8.76MB, 2.52MB, 3.62MB), 0.15MB, 0.52MB, 2.77MB, and 0.16MB for these five components on the most challenging BungeeNeRF dataset [42] with $\lambda_e = 4e-3$. With scenes become simpler, the storage share of $\mathcal{A}$ decreases as the value distribution become easier to predict.

4.3 Ablation Study

In this subsection, we conduct ablation studies to demonstrate effectiveness of each technical component. We conduct experiments on both the most challenging large-scale BungeeNeRF dataset [42] and the small-scale Synthetic-NeRF

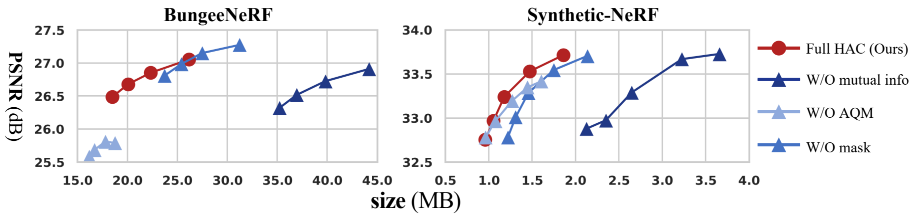


Fig. 6: Ablations of different components in HAC. We vary λ_e for variable rates.

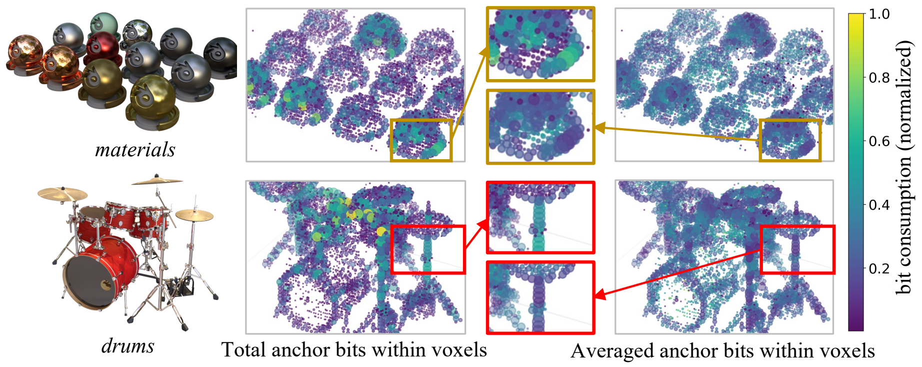

materials
drums
Total anchor bits within voxels
Averaged anchor bits within voxels
bit consumption (normalized)
1.0
0.8
0.6
0.4
0.2

Fig. 7: Visualization of bit allocation maps for the scenes “materials” and “drums” on Synthetic-NeRF dataset [28]. The 3D space is voxelized, with each voxel represented by a ball and the radius of a ball indicating the number of anchors in the voxel. For the 2nd column, the color of a ball indicates the *total* bit consumption of all anchors in the voxel, while for the 4th column, the color represents the *averaged* bit consumption per anchor within a voxel. The 3rd column gives zoom-in views. It shows more anchors are allocated to important regions while the bit consumption for each anchor is smooth.

dataset [28] to support convincing and solid results. We assess the effectiveness of individual technical components by disabling either of the following: 1) mutual information from the hash grid, 2) the adaptive quantization module, 3) adaptive offset masking. The results are presented in Fig. 6. Firstly, we set the hash grid to all zeros to eliminate mutual information. This leads to a degradation of conditional probability from $p(\mathcal{A}|\boldsymbol{f}^h)$ to $p(\mathcal{A})$, which indicates that probability of $\hat{\boldsymbol{f}}$ can only be estimated by the statistic μ and σ from the left part of Fig. 3. Consequently, the bit consumption drastically increases as the probability can no longer be accurately estimated. Regarding the latter two components, they contribute from different perspectives. Disabling AQM (we remove $\boldsymbol{r}$ while retaining Q_0 to ensure a necessary decimal quantization step) results in a significant drop in fidelity, especially in more complex scenes or at higher rates, as $\hat{\boldsymbol{f}}$ fails to retain sufficient information for rendering after quantization. Differently, offset masking can achieve remarkable rate savings in simpler scenes or lower rate segments due to more significant positional redundancy in Gaussians. Overall, all three components provide a worthwhile tradeoff for improved RD performance.

4.4 Visualization of Bit Allocation

While HAC measures the parameters' bit consumption, we are interested in the bit allocation across different local areas in the space. In Fig. 7, we utilize scenes in Synthetic-NeRF dataset [28] for visualization, and represent bit allocation conditions by voxelized colored balls. As observed from the 2nd column of visualized sub-figures, the model tends to allocate more total bits to areas with complex appearances or sharp edges. For instance, specular objects in "materials" and instrument stands in "drums" exhibit higher total bit consumption due to the complex textures. The analysis of the 4th column from an averaging viewpoint reveals varied trends in bit consumption per anchor. In high bit-consumption voxels, creating more anchors for precise modeling averages the bit per anchor, smoothing or reducing bit consumption for each. This aligns with our assumption that anchors demonstrate inherent consistency in the 3D space where nearby anchors exhibit similar values of attributes, making it easier for the hash grid to accurately estimate their value probabilities.

4.5 Training and Execution Time

Training time. Use of additional models in HAC results in increased training time, approximately 0.9× longer than Scaffold-GS. For the challenging BungeeNeRF dataset [42], the training times are 38.2 m for 3DGS [19], 15.1 m for Scaffold-GS [27] and 27.6 m for HAC. For the small-scale Synthetic-NeRF dataset [28], training times are 3.4 m, 4.4 m and 9.0 m, respectively. This increase of training time in our model over Scaffold-GS is our main limitation, but it is still fast.
Coding time. The encoding/decoding process takes approximately 0.87 seconds and 26.7 seconds on Synthetic-NeRF and BungeeNeRF dataset under $\lambda_e = 4e{-}3$, respectively. The dominant time consumption occurs during Codec execution of AE on the CPU (over 90%), as we only use a single thread.
Inference time. The inference process benefits from the design of context modeling, allowing for the removal of the hash grid once $\mathcal{A}$ is decoded. Consequently, no additional operations are required during rendering, resulting in a similar FPS with Scaffold-GS. The rendering FPS are 75, 232 and 283 for 3DGS, Scaffold-GS and HAC on BungeeNeRF, and 401, 326 and 341 on Synthetic-NeRF. The improved FPS of our model compared to Scaffold-GS is likely due to the pruning of invalid Gaussians/anchors, which in turn facilitates faster rendering.

5 Conclusion

We pioneered an investigation into the relationship between unorganized and sparse Gaussians (or anchors) and well-structured hash grids, leveraging their mutual information for compact 3DGS representations. Our Hash-grid Assisted Context (HAC) framework has achieved SoTA compression performance with remarkable leading over concurrent works. Extensive experiments have demonstrated the effectiveness of our HAC and its technical components. Overall, our work has successfully mitigated the major challenging of 3DGS models, *i.e.*, large storage, enabling its adoption in large-scale scenes and diverse devices.

Acknowledgement

The paper is supported in part by The National Natural Science Foundation of China (No. 62325109, U21B2013).
MH is supported by funding from The Australian Research Council Discovery Program DP230101176.

References

1. Ballé, J., Minnen, D., Singh, S., Hwang, S.J., Johnston, N.: Variational image compression with a scale hyperprior. In: International Conference on Learning Representations (2018)
2. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5470–5479 (2022)
3. Bengio, Y., Léonard, N., Courville, A.: Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432 (2013)
4. Bjontegaard, G.: Calculation of average psnr differences between rd-curves. ITU SG16 Doc. VCEG-M33 (2001)
5. Chen, A., Xu, Z., Geiger, A., Yu, J., Su, H.: Tensorf: Tensorial radiance fields. In: European Conference on Computer Vision. pp. 333–350. Springer (2022)
6. Chen, G., Wang, W.: A survey on 3d gaussian splatting. arXiv preprint arXiv:2401.03890 (2024)
7. Chen, Y., Wu, Q., Harandi, M., Cai, J.: How far can we compress instant-ngp-based nerf? In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2024)
8. Cheng, Z., Sun, H., Takeuchi, M., Katto, J.: Learned image compression with discretized gaussian mixture likelihoods and attention modules. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 7939–7948 (2020)
9. Cover, T.M.: Elements of information theory. John Wiley & Sons (1999)
10. 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)
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. Fei, B., Xu, J., Zhang, R., Zhou, Q., Yang, W., He, Y.: 3d gaussian as a new vision era: A survey. arXiv preprint arXiv:2402.07181 (2024)
13. Fridovich-Keil, S., Meanti, G., Warburg, F.R., Recht, B., Kanazawa, A.: K-planes: Explicit radiance fields in space, time, and appearance. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12479–12488 (2023)
14. Girish, S., Gupta, K., Shrivastava, A.: Eagles: Efficient accelerated 3d gaussians with lightweight encodings. arXiv preprint arXiv:2312.04564 (2023)
15. 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)

16. He, D., Yang, Z., Peng, W., Ma, R., Qin, H., Wang, Y.: Elic: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5718–5727 (2022)
17. He, D., Zheng, Y., Sun, B., Wang, Y., Qin, H.: Checkerboard context model for efficient learned image compression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14771–14780 (2021)
18. Hedman, P., Philip, J., Price, T., Frahm, J.M., Drettakis, G., Brostow, G.: Deep blending for free-viewpoint image-based rendering. ACM Transactions on Graphics (ToG) **37**(6), 1–15 (2018)
19. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. ACM Transactions on Graphics **42**(4) (2023)
20. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics (ToG) **36**(4), 1–13 (2017)
21. Lassner, C., Zollhofer, M.: Pulsar: Efficient sphere-based neural rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1440–1449 (2021)
22. Lee, J.C., Rho, D., Sun, X., Ko, J.H., Park, E.: Compact 3d gaussian representation for radiance field. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2024)
23. Li, J., Li, B., Lu, Y.: Deep contextual video compression. Advances in Neural Information Processing Systems **34**, 18114–18125 (2021)
24. Li, J., Li, B., Lu, Y.: Hybrid spatial-temporal entropy modelling for neural video compression. In: Proceedings of the 30th ACM International Conference on Multimedia. pp. 1503–1511 (2022)
25. 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)
26. Li, L., Wang, Z., Shen, Z., Shen, L., Tan, P.: Compact real-time radiance fields with neural codebook. In: ICME (2023)
27. Lu, T., Yu, M., Xu, L., Xiangli, Y., Wang, L., Lin, D., Dai, B.: Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2024)
28. 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)
29. Morgenstern, W., Barthel, F., Hilsmann, A., Eisert, P.: Compact 3d scene representation via self-organizing gaussian grids. arXiv preprint arXiv:2312.13299 (2023)
30. 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)
31. 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)
32. Niedermayr, S., Stumpfegger, J., Westermann, R.: Compressed 3d gaussian splatting for accelerated novel view synthesis. arXiv preprint arXiv:2401.02436 (2023)
33. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)

34. 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. pp. 20680–20690 (2023)
35. Schonberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016)
36. Sheng, X., Li, J., Li, B., Li, L., Liu, D., Lu, Y.: Temporal context mining for learned video compression. IEEE Transactions on Multimedia (2022)
37. Shin, S., Park, J.: Binary radiance fields. Advances in neural information processing systems (2023)
38. Song, Z., Duan, W., Zhang, Y., Wang, S., Ma, S., Gao, W.: Spc-nerf: Spatial predictive compression for voxel based radiance field. arXiv preprint arXiv:2402.16366 (2024)
39. 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)
40. Tang, J., Chen, X., Wang, J., Zeng, G.: Compressible-composable nerf via rank-residual decomposition. Advances in Neural Information Processing Systems **35**, 14798–14809 (2022)
41. Witten, I.H., Neal, R.M., Cleary, J.G.: Arithmetic coding for data compression. Communications of the ACM **30**(6), 520–540 (1987)
42. Xiangli, Y., Xu, L., Pan, X., Zhao, N., Rao, A., Theobalt, C., Dai, B., Lin, D.: Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering. In: European conference on computer vision. pp. 106–122. Springer (2022)