RacerGS: Residual-Autoregressive Context for 3D Gaussian Splatting Compression

Wenqing Wang **Huimin Zeng** **Yun Fu**
Northeastern University
{wang.wenqin, zeng.huim, raymondfu}@northeastern.edu

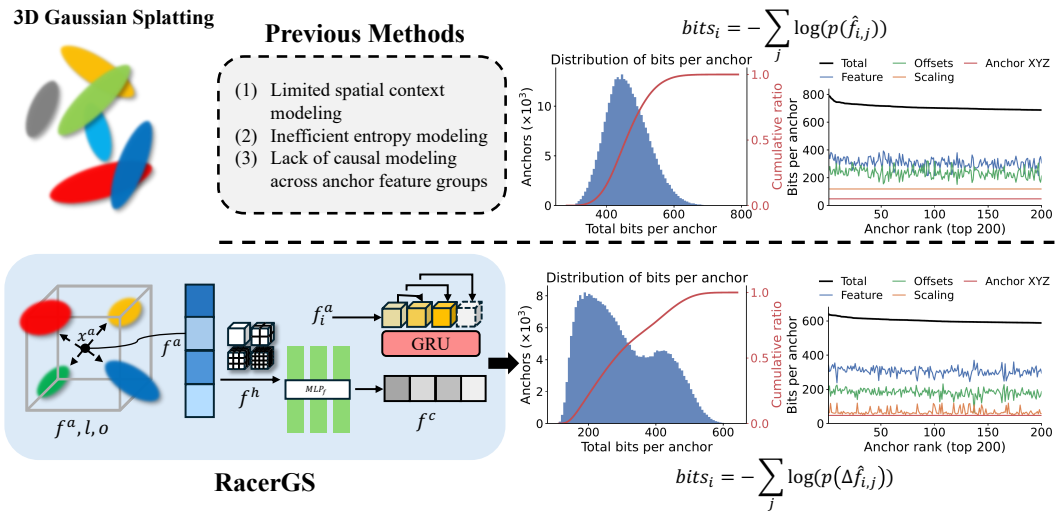

Figure 1: **RacerGS for efficient 3DGS compression.** Previous methods have limited spatial context, inefficient entropy modeling, and lack of causal dependencies across anchor feature groups, leading to high coding costs. RacerGS addresses these limitations through compact context projection, residual coding, and autoregressive GRU refinement. This shifts the bitrate distribution toward fewer bits per anchor and lowers total coding costs while preserving rendering fidelity.

Abstract

3D Gaussian Splatting (3DGS) has emerged as a promising approach for fast, high-quality novel-view rendering. However, its large parameter size remains a bottleneck for storage and transmission. We propose *RacerGS*, a compact 3DGS compression framework that achieves scene reconstruction at significantly reduced storage sizes with preserved fidelity. To obtain compact and spatially coherent features, we design a hash projection model that projects the anchor hash encodings into compact context features. Additionally, we introduce a FiLM-modulated autoregressive GRU context model that utilizes causal dependencies among anchor feature groups and uses recurrent hidden states to refine entropy-model parameters, improving bitrate efficiency. Furthermore, we adaptively entropy-code quantized anchor-attribute residuals relative to their context-predicted means, leveraging spatial consistency to reduce symbol entropy. Overall, *RacerGS* achieves more than $\mathbf{150}\times$ compression over vanilla 3DGS and $\mathbf{28}\times$ over Scaffold-GS, while providing comparable rendering quality. Extensive experiments across five datasets demonstrate that *RacerGS* outperforms prior 3DGS compression methods in both storage reduction and rate-distortion performance.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Novel view synthesis has advanced rapidly. Neural Radiance Fields (NeRF) [25] enable high-quality rendering by modeling scenes as continuous radiance fields, but NeRF-based methods [25, 27, 2, 10, 38, 39] incur high rendering costs due to dense sampling and repeated field evaluations along each ray. 3D Gaussian Splatting (3DGS) [15] provides an efficient alternative by explicitly representing scenes with learnable 3D Gaussians and rendering them through tile-based rasterization. This approach enables substantially faster rendering while maintaining high visual quality. However, large-scale scenes can require millions of Gaussians, creating substantial storage and transmission demands.

To compress these large, unstructured Gaussian sets, value-centric methods [17, 30, 12, 8, 37, 29, 18, 22, 45] reduce parameter count and bitwidth via pruning, quantization, and codebooks, but ignore spatial redundancy. Structure-aware methods [26, 9, 47, 43, 40, 48, 49] instead predict attributes from local neighborhoods. In particular, Scaffold-GS [24] groups Gaussians around learned anchors and predicts their attributes from anchor features, improving rate–distortion (RD) trade-offs. Despite these gains, Scaffold-GS models anchors independently, producing sparse, point-cloud-like representations that remain difficult to compress. HAC methods [4, 5] capture spatial relations via hash grids, but feed concatenated features directly to entropy networks, increasing cost with dimensionality without explicitly modeling interactions across dimensions. They also quantize anchor attributes directly, ignoring the spatial regularity of residuals relative to predicted means and inflating storage. While HAC++ [5] builds intra-anchor context by repeatedly concatenating decoded chunks, it neither forms a compact history nor captures higher-order dependencies.

We empirically observe that prior methods consume a large number of bits per anchor (Fig. 1), suggesting room to improve entropy estimation and bit allocation. These observations highlight three challenges: 1) limited spatial context modeling, where hash or anchor features are used directly without capturing structured dependencies; 2) inefficient entropy modeling that ignores the stronger regularity in attribute residuals; and 3) the lack of causal modeling across anchor feature groups, preventing the entropy model from exploiting intra-anchor dependencies.

To address these challenges, we propose ***RacerGS***, a unified 3DGS compression framework that jointly leverages structured spatial context, residual entropy modeling, and recurrent causal context refinement. A **hash projection model** projects anchor hash encodings into compact, fused context features for a stronger prior. A **FiLM-modulated autoregressive GRU context model** refines entropy parameters via recurrent hidden states over causally ordered feature groups. Finally, **adaptive residual quantization** encodes residuals around their context-predicted means, yielding lower-entropy symbols and improved RD performance.

Our contributions are summarized as follows:

- We propose *RacerGS*, a 3DGS compression framework that achieves highly compact representations while preserving high-fidelity rendering quality.
- We address existing 3DGS compression challenges with a hash projection model, a FiLM-modulated autoregressive GRU context model, and adaptive residual quantization for entropy coding, which jointly enhance context modeling and tighten entropy estimation.
- Extensive experiments on diverse datasets demonstrate that our method achieves superior 3DGS compression (e.g., $\mathbf{150}\times$ over 3DGS; $\mathbf{28}\times$ over Scaffold-GS) while preserving rendering quality (e.g., a $\mathbf{2.31}$ **dB** PSNR gain over 3DGS; $\mathbf{0.56}$ **dB** over Scaffold-GS).

2 Related Work

Neural radiance fields and compression. Neural Radiance Fields [25] achieve high-quality novel view synthesis but rely on dense sampling and large MLPs that limit real-time rendering. Explicit representations such as K-planes [10], Instant-NGP [27], and TensoRF [2] replace much of the MLP with structured grids, trading compute for storage. NeRF compression broadly splits into *value-based* approaches (pruning [7, 19], vector quantization [34], codebooks [20], and entropy-aware training [11]) and *structure-based* approaches that exploit spatial correlations via factorization [36, 33], predictive coding [35], or transform coding [3, 21, 14]. Value-based approaches primarily reduce the precision or redundancy of stored parameters, while structure-based approaches expose dependencies through the organization of scene information. Both seek to balance representation

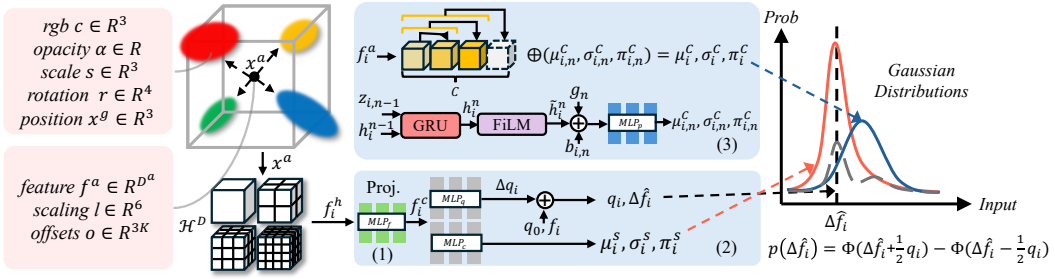

Figure 2: **Pipeline overview.** *RacerGS* compresses 3DGS through three components: **(1)** a hash projection model maps hash encodings $\boldsymbol{f}_i^h$ to compact spatial context features $\boldsymbol{f}_i^c$; **(2)** adaptive residual quantization encodes anchor attributes relative to context-predicted means using learned quantization steps; **(3)** an autoregressive GRU summarizes previously decoded feature chunks, followed by FiLM modulation and a shared predictor conditioned on base statistics and chunk embeddings. The refined parameters are combined with the spatial prior to form an accurate Gaussian mixture model for residual entropy coding.

size against reconstruction quality, with compression efficiency depending on how effectively the representation captures and exploits redundancy. These principles motivate compression methods for explicit 3D scene representations.

3D Gaussian splatting and compression. 3D Gaussian Splatting (3DGS) [15] enables fast, high-fidelity rendering with learnable Gaussians, but its large parameter count leads to high storage costs. Existing compression methods can be broadly divided into value-based [8, 28, 18, 30, 29, 22, 45] and structure-based approaches [26, 47, 9, 43, 24, 40, 5, 23, 48, 49]. Value-based methods reduce attribute redundancy via codebooks and vector quantization [29], entropy modeling [22], or feature-plane projections for standard codecs [18]. Structure-aware methods exploit spatial organization, such as multi-level grids [43], anchor-based prediction [24], anchor context modeling [40], and hash-grid context for anchor quantization [4, 5]. However, HAC-style methods rely on poorly scaled hash contexts, miss higher-order causal dependencies, and encode raw attributes instead of spatially regular residuals, increasing entropy and bitrate. We address these limitations with structured spatial context (a learned projection rather than raw hash features), residual entropy modeling (prediction-centered residuals rather than attributes), and recurrent causal context refinement (a constant-cost recurrent state rather than growing concatenation).

3 Preliminaries

3D Gaussian Splatting [15] models a scene with a collection of 3D Gaussians characterized by opacity α, color c, a 3D covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$, and a center position $\boldsymbol{\mu} \in \mathbb{R}^3$:

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

where $\boldsymbol{x} \in \mathbb{R}^3$ denotes the 3D point position, and $\boldsymbol{\Sigma}$ is defined as $\boldsymbol{RSS}^\top \boldsymbol{R}^\top$ with the rotation matrix $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ and scaling matrix $\boldsymbol{S} \in \mathbb{R}^{3\times3}$. The pixel color C is computed by alpha compositing: $C = \sum_{i\in N} c_i\alpha_i \prod_{j=1}^{i-1}(1-\alpha_j)$.

Scaffold-GS [24] introduces an anchor-based 3DGS approach that optimizes the storage requirements and preserves fidelity. It utilizes anchors to group 3D Gaussians and predict their attributes from their associated anchor attributes with MLPs. Each anchor is defined by a position $\boldsymbol{x}^a \in \mathbb{R}^3$ and attributes $\mathcal{A} = \{\boldsymbol{f}^a \in \mathbb{R}^{D^a}, \boldsymbol{l} \in \mathbb{R}^6, \boldsymbol{o} \in \mathbb{R}^{3K}\}$, corresponding to anchor features, scaling parameters, and offset parameters, respectively. During rendering, $\boldsymbol{f}^a$ is fed to MLPs to predict the attributes of the K Gaussians, whose positions are $\boldsymbol{x}^a$ plus the scaled offsets $\boldsymbol{o}$, and $\boldsymbol{l}$ regulates their positions and shapes.

4 Method

4.1 Overview

As illustrated in Fig. 2, *RacerGS* compresses 3DGS by projecting anchor hash encodings into compact context features and predicting entropy model parameters for anchor attribute residuals. For each anchor, a hash projection model extracts context features, while a FiLM-modulated autoregressive GRU recurrently refines per-group distribution parameters using previously decoded feature groups. RacerGS then entropy-encodes quantized residuals around their context-predicted means, reducing coding entropy.

4.2 Context feature projection

Using the raw hash features $\boldsymbol{f}^h$ directly as the entropy context [4, 5] ties the input width of every entropy MLP to the hash dimensionality and forces one encoding to serve both rendering and entropy modeling, whose capacity demands conflict. To obtain spatially coherent context features at the anchor position $\boldsymbol{x}^a \in \mathbb{R}^3$, we instead first query the multi-dimensional hash grid $\mathcal{H}$ to obtain the anchor hash encodings $\boldsymbol{f}_i^h = \mathrm{Interp}(\boldsymbol{x}_i^a, \mathcal{H}) \in \mathbb{R}^{D^h}$. Then, we project $\boldsymbol{f}_i^h$ with the hash projection model $\mathrm{MLP_f}$ into the compact fused context features:

$$\boldsymbol{f}_i^c = \mathrm{MLP_f}(\boldsymbol{f}_i^h) \in \mathbb{R}^{D^c}. \tag{2}$$

$\boldsymbol{f}_i^c$ is the only signal the downstream entropy model receives from the hash grid. The non-linear MLP explicitly mixes inter-dimensional cues, and joint training with the entropy loss preserves only the bits-relevant directions of $\boldsymbol{f}^h$. This enables the hash grid to grow its representational capacity freely without inflating entropy-parameter cost, while the entropy context becomes structured and low-dimensional.

4.3 Adaptive residual entropy coding

4.3.1 Adaptive residual quantization

Prior methods [4, 5, 40] entropy-code anchor attributes directly, ignoring the much stronger regularity of residuals around spatial-context-predicted means. We reduce entropy by encoding residuals of anchor attributes $\mathcal{A}$ with respect to their context-predicted means. For the i-th anchor $\boldsymbol{x}_i^a$, let $f_i \in \{f_i^a, l_i, o_i\}$ denote any component of $\mathcal{A}_i$. To operate on the discrete symbol space for entropy coding, we quantize the anchor attribute residual Δf_i with adaptive residual quantization. Specifically, we first predict the quantization step adjustment Δq_i and the entropy model parameters $\boldsymbol{\mu}_i^{\mathrm{s}},\ \boldsymbol{\sigma}_i^{\mathrm{s}},\ \boldsymbol{\pi}_i^{\mathrm{s}}$ for f_i from the fused context $\boldsymbol{f}_i^c$.

$$\begin{aligned} \Delta q_i &= \mathrm{MLP_q}(\boldsymbol{f}_i^c), \\ \boldsymbol{\mu}_i^{\mathrm{s}},\ \boldsymbol{\sigma}_i^{\mathrm{s}},\ \boldsymbol{\pi}_i^{\mathrm{s}} &= \mathrm{MLP_c}(\boldsymbol{f}_i^c). \end{aligned} \tag{3}$$

Then, we dynamically quantize the anchor attribute residual Δf_i in terms of the predicted mean $\boldsymbol{\mu}_i^s$:

$$\begin{aligned} \Delta \boldsymbol{f}_i &= \boldsymbol{f}_i - \boldsymbol{\mu}_i^s, \qquad & \boldsymbol{r}_i &= \lfloor \Delta \boldsymbol{f}_i \oslash \boldsymbol{q}_i \rceil \in \mathbb{Z}^{D^f}, \\ \Delta \hat{\boldsymbol{f}}_i &= \boldsymbol{r}_i \odot \boldsymbol{q}_i, \qquad & \hat{\boldsymbol{f}}_i &= \boldsymbol{\mu}_i^s + \Delta \hat{\boldsymbol{f}}_i, \end{aligned} \tag{4}$$

where $\odot$ and $\oslash$ denote element-wise product and division, $\boldsymbol{q}_i = q_0 \cdot \big(1 + \tanh(\Delta q_i)\big)$, and q_0 is the attribute-specific initial quantization step size. $\boldsymbol{r}_i$ is the integer symbol encoded by the arithmetic coder, $\Delta \hat{\boldsymbol{f}}_i$ the quantized residual, and $\hat{\boldsymbol{f}}_i$ the reconstruction. Decoding requires only $\boldsymbol{r}_i$ and $\boldsymbol{f}_i^c$, which predicts $\boldsymbol{\mu}_i^s$ and $\boldsymbol{q}_i$. During training, we quantize by injecting uniform noise with bin width q_i: $\Delta \hat{\boldsymbol{f}}_i^{\text{train}} = \Delta f_i + \boldsymbol{\epsilon}_i$, where $\boldsymbol{\epsilon}_i \sim \mathcal{U}\big(-\frac{q_i}{2}, \frac{q_i}{2}\big)$. This yields the straight-through gradient $\frac{\partial \Delta \hat{\boldsymbol{f}}_i^{\text{train}}}{\partial \Delta f_i} = \mathbf{I}$, enabling gradient backpropagation. Compared with direct quantization, centering on $\boldsymbol{\mu}_i^{\mathrm{s}}$ contracts the dynamic range and lowers residual entropy, which improves entropy coding efficiency.

4.3.2 Residual distribution modeling

For an anchor $\boldsymbol{x}_i^a$ and its quantized anchor attribute residual $\Delta\hat{\boldsymbol{f}}_i \in \mathbb{R}^{D^f}$, the probability of $\Delta\hat{\boldsymbol{f}}_i$ is computed using the cumulative distribution function difference over the quantization bins:

$$\begin{aligned} p(\Delta\hat{\boldsymbol{f}}_i) &= \prod_{j=1}^{D^f} \int_{\Delta\hat{f}_{i,j}-\frac{q_{i,j}}{2}}^{\Delta\hat{f}_{i,j}+\frac{q_{i,j}}{2}} \phi\big(x; 0, \sigma_{i,j}^{\mathrm{s}}\big)\, dx \\ &= \prod_{j=1}^{D^f} \Big[\Phi\Big(\Delta\hat{f}_{i,j} + \tfrac{q_{i,j}}{2}; 0, \sigma_{i,j}^{\mathrm{s}}\Big) - \Phi\Big(\Delta\hat{f}_{i,j} - \tfrac{q_{i,j}}{2}; 0, \sigma_{i,j}^{\mathrm{s}}\Big)\Big], \end{aligned} \tag{5}$$

where ϕ and Φ are the probability density function and the cumulative distribution function of a zero-mean Gaussian distribution, since the predicted residual $\Delta\hat{\boldsymbol{f}}_i$ is centered near zero. We then entropy code $p(\Delta\hat{\boldsymbol{f}}_i)$ using arithmetic coding [42] and store the resulting bytestream payload of the compressed probability symbols. With per-element code length $-\log_2 p(\Delta\hat{f}_{i,j})$, this approach substantially reduces the storage footprint and the coding entropy.

4.4 Causal entropy model refinement

To further improve bitrate efficiency and reduce the parameter redundancies of the entropy model, we introduce a FiLM-modulated autoregressive GRU context model that refines the entropy model parameters by exploiting the causal relations inside the anchor feature groups and operating on the recurrent context hidden states. For each anchor $\boldsymbol{x}_i^a$, we split its anchor feature $\boldsymbol{f}_i^a$ into N^C chunks $\boldsymbol{f}_i^a = [\boldsymbol{f}_{i,1}^a; \ldots; \boldsymbol{f}_{i,N^C}^a]$, where the chunk index $n \in \{1, \ldots, N^C\}$ follows the decoding order and each chunk contains C contiguous feature channels. Then, we autoregressively predict the distribution parameters $\mu_{i,n}^C, \sigma_{i,n}^C, \pi_{i,n}^C$ for $\boldsymbol{f}_{i,n}^a$, which are utilized to compute the distribution parameters for $\boldsymbol{f}_i^a$ as follows:

$$\mu_i^C, \sigma_i^C, \pi_i^C = \bigoplus_{n=1}^{N^C} \left(\mu_{i,n}^C, \sigma_{i,n}^C, \pi_{i,n}^C\right). \tag{6}$$

Existing methods [48, 5] build per-chunk causal context by concatenating raw decoded chunks, which causes the predictor input to scale linearly with the chunk count; higher-order interactions are captured only implicitly through the predictor's first MLP layer. In contrast, our method has constant per-step input cost, explicit gating-based modeling of inter-chunk interactions, and parameter-efficient chunk specialization.

4.4.1 Recurrent autoregressive GRU context model

We refine the chunkwise entropy parameters by conditioning on previously decoded feature groups within the same anchor. Let $\hat{\boldsymbol{f}}_{i,<n}^a = [\hat{\boldsymbol{f}}_{i,1}^a; \ldots; \hat{\boldsymbol{f}}_{i,n-1}^a]$ denote the already decoded chunks. We first form a base context vector $\boldsymbol{s}_i = \left[\boldsymbol{\mu}_i^s; \boldsymbol{\sigma}_i^s; \boldsymbol{\pi}_i^s\right]$ by concatenating the per-channel base statistics predicted from $\boldsymbol{f}_i^c$, and encode it into an initial hidden state:

$$\boldsymbol{h}_i^0 = \mathrm{MLP}_{\mathrm{cxt}}(\boldsymbol{s}_i). \tag{7}$$

To inject causal information, we embed each previously decoded chunk into a token sequence. Let $\boldsymbol{z}_{i,t} = \mathrm{MLP}_{\mathrm{prev}}(\hat{\boldsymbol{f}}_{i,t}^a)$ for $t < n$, and update a single-layer GRU by exactly one step per chunk:

$$\boldsymbol{h}_i^1 = \boldsymbol{h}_i^0, \qquad \boldsymbol{h}_i^n = \mathrm{GRU}\big(\boldsymbol{z}_{i,n-1},\ \boldsymbol{h}_i^{n-1}\big), \quad n = 2, \ldots, N^C, \tag{8}$$

where $\boldsymbol{h}_i^n$ is the chunk-specific latent summarizing all preceding decoded chunks. The cached hidden state advances through one constant-cost GRU step per chunk, without reprocessing decoded chunks. Operating on the recurrent hidden state $\boldsymbol{h}_i^n$ provides a compact summary of all previously decoded chunks, allowing the model to capture higher-order intra-anchor dependencies. Unlike in existing methods, the size of $\boldsymbol{h}_i^n$ is independent of n, so the predictor input cost is constant per chunk. Also, the GRU's gating models multiplicative inter-chunk interactions explicitly rather than relying on the predictor's first MLP layer to discover them.

4.4.2 Anchor-conditional FiLM modulation

While $\boldsymbol{h}_i^n$ aggregates causal tokens, its optimal scale and bias depend on the anchor-specific base statistics $\boldsymbol{s}_i$. Directly feeding $\boldsymbol{h}_i^n$ into the parameter prediction model can be restrictive, since the same hidden magnitude and offset must simultaneously fit anchors with very different predicted uncertainty patterns and mixture preferences. To provide an efficient anchor-adaptive reparameterization, we apply feature-wise linear modulation (FiLM) [32] to the chunk latent. We predict per-channel modulation vectors from $\boldsymbol{h}_i^n$,

$$\boldsymbol{\gamma}_i^n, \boldsymbol{\beta}_i^n = \mathrm{MLP}_{\mathrm{FiLM}}(\boldsymbol{h}_i^n)\,, \tag{9}$$

and modulate a copy of the hidden representation as:

$$\tilde{\boldsymbol{h}}_i^n = \boldsymbol{h}_i^n \odot (\mathbf{1} + \tanh(\boldsymbol{\gamma}_i^n)) + \boldsymbol{\beta}_i^n. \tag{10}$$

This modulation acts as a lightweight chunkwise conditioning layer that both improves optimization stability and increases the effective capacity of the refinement module without deepening the GRU, which preserves the parameter footprint and improves coding efficiency.

4.4.3 Weight-shared chunk-positional predictor

To enhance sample efficiency and minimize model size, we utilize a shared parameter prediction model across all feature chunks. However, different chunks $\boldsymbol{f}_{i,n}^a$ correspond to different channel subsets and exhibit distinct statistics. To enable the shared parameter model to specialize for each chunk without duplicating weights, we introduce a learnable group positional embedding $\boldsymbol{g}_n = \mathrm{Emb}(n)$. Conditioned on the modulated causal hidden state $\tilde{\boldsymbol{h}}_i^n$, the base parameter embedding $\boldsymbol{b}_{i,n} = \mathrm{MLP}_{\mathrm{base}}\big(\boldsymbol{\mu}_{i,n}^s\,,\ \boldsymbol{\sigma}_{i,n}^s\,,\ \boldsymbol{\pi}_{i,n}^s\big)$, and the group positional embedding $\boldsymbol{g}_n$, we apply the shared parameter model to predict the distribution parameters for $\boldsymbol{f}_{i,n}^a$:

$$\mu_{i,n}^C, \sigma_{i,n}^C, \pi_{i,n}^C = \mathrm{MLP}_{\mathrm{p}}(\tilde{\boldsymbol{h}}_i^n \oplus \boldsymbol{b}_{i,n} \oplus \boldsymbol{g}_n). \tag{11}$$

Here, $\boldsymbol{\mu}_{i,n}^C$ and $\boldsymbol{\sigma}_{i,n}^C$ parameterize the refined Gaussian distribution component for channels in chunk n, while $\boldsymbol{\pi}_{i,n}^C$ is the mixture logit that controls the probability mass assigned to the refined component (Eq. (12)).

4.4.4 Residual mixture distribution with refined context

To improve the probability estimation accuracy of the anchor feature residual $\Delta\boldsymbol{f}_i^a$, we leverage a Gaussian mixture distribution that combines the base entropy model parameters predicted from the hash context feature $\boldsymbol{f}_i^c$ and the refinement entropy model parameters predicted from the causal context relations inside the anchor feature $\boldsymbol{f}_i^a$. The resulting mixture probability distribution is defined as:

$$p(\Delta\hat{\boldsymbol{f}}_i^a) = \prod_{j=1}^{D^f} \sum_{k\in\{\mathrm{s,c}\}} \pi_{i,j}^k \Big[\Phi\Big(\Delta\hat{f}_{i,j}^a + \tfrac{q_{i,j}}{2};\, \mu_{i,j}^k,\, \sigma_{i,j}^k\Big) - \Phi\Big(\Delta\hat{f}_{i,j}^a - \tfrac{q_{i,j}}{2};\, \mu_{i,j}^k,\, \sigma_{i,j}^k\Big)\Big]. \tag{12}$$

This mixture entropy model enables the base predictor to capture coarse residual statistics from $\boldsymbol{f}_i^c$, while the refinement component adjusts the probability mass using causal dependencies within anchor feature groups, improving probability accuracy and bitrate efficiency.

4.5 Optimization

4.5.1 Residual entropy loss

To improve the estimation accuracy of $p(\Delta\hat{\boldsymbol{f}}_i)$, we minimize the predicted entropy of the anchor attribute residuals. Therefore, we define the residual entropy loss as the total expected bit consumption over all N anchors:

$$\mathcal{L}_{\Delta\mathrm{entropy}} = \sum_{i=1}^{N} m_i^a \Big[\sum_{\boldsymbol{f}\in\{\boldsymbol{f}^a,\boldsymbol{l}\}} \sum_{j=1}^{D^f} -\log_2 p\Big(\Delta\hat{f}_{i,j}\Big) + \sum_{k=1}^{K} m_{i,k} \sum_{j=1}^{3} -\log_2 p(\Delta\hat{o}_{i,k,j})\Big], \tag{13}$$

where K is the number of offset vectors attached to each anchor, and m_i^a and $m_{i,k}$ are learnable binary masks [17] used to prune inactive anchors indexed by i and their k-th offsets, respectively.

Table 1: **Quantitative comparisons.** Storage sizes are measured in MB. The best and second-best compression results are highlighted in green and blue, respectively. Overall, RacerGS outperforms the baselines in size reduction and fidelity preservation across diverse datasets.

Datasets	Synthetic-NeRF				Mip-NeRF360				Tanks&Temples			
Methods	psnr↑	ssim↑	lpips↓	size↓	psnr↑	ssim↑	lpips↓	size↓	psnr↑	ssim↑	lpips↓	size↓
3DGS	33.80	0.970	0.031	68.46	27.46	0.812	0.222	750.9	23.69	0.844	0.178	431.0
Scaffold-GS	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.*	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	32.94	0.967	0.033	3.68	26.98	0.801	0.238	28.80	23.32	0.832	0.194	17.28
EAGLES	32.51	0.964	0.039	4.26	27.14	0.809	0.231	58.91	23.28	0.835	0.203	28.99
SizeGS	32.27	0.956	0.043	2.24	27.19	0.788	0.253	17.05	24.11	0.833	0.191	10.51
LightGaussian	32.73	0.965	0.037	7.84	27.00	0.799	0.249	44.54	22.83	0.822	0.242	22.43
SOG	31.37	0.959	0.043	2.00	26.56	0.791	0.241	16.70	23.15	0.828	0.198	9.30
Navaneet *et al.*	32.99	0.966	0.037	3.10	27.12	0.806	0.240	19.33	23.44	0.838	0.198	12.50
Reduced3DGS	33.02	0.967	0.035	2.11	27.19	0.807	0.230	29.54	23.57	0.840	0.188	14.00
CAT-3DGS Pro	32.41	0.959	0.041	2.07	27.33	0.793	0.245	15.62	24.26	0.838	0.188	9.74
RDOGaussian	33.12	0.967	0.035	2.31	27.05	0.802	0.239	23.46	23.34	0.835	0.195	12.03
HEMGS	32.54	0.962	0.038	1.91	27.46	0.797	0.241	14.38	24.39	0.842	0.185	9.03
MesonGS-FT	32.92	0.968	0.033	3.66	26.99	0.796	0.247	27.16	23.32	0.837	0.193	16.99
HAC (lr)	33.24	0.967	0.037	1.18	27.53	0.807	0.238	15.26	24.04	0.846	0.187	8.10
HAC (hr)	33.71	0.968	0.034	1.86	27.77	0.811	0.230	21.87	24.40	0.853	0.177	11.24
ContextGS (lr)	32.79	0.965	0.040	1.01	27.62	0.808	0.237	12.68	24.20	0.852	0.184	7.05
ContextGS (hr)	33.51	0.968	0.035	1.56	27.75	0.811	0.231	18.41	24.29	0.855	0.176	11.80
CompGS (lr)	-	-	-	-	26.37	0.778	0.267	8.83	23.11	0.815	0.236	5.89
CompGS (hr)	-	-	-	-	27.26	0.803	0.239	16.50	23.70	0.837	0.208	9.60
HAC++ (lr)	33.03	0.966	0.039	0.88	27.60	0.803	0.253	8.34	24.22	0.849	0.190	5.18
HAC++ (hr)	33.76	0.969	0.033	1.84	27.82	0.811	0.231	18.48	24.32	0.854	0.178	8.63
RacerGS (lr)	32.61	0.964	0.041	0.28	27.51	0.801	0.262	3.85	24.45	0.847	0.195	4.19
RacerGS (hr)	33.78	0.970	0.035	1.44	27.84	0.811	0.243	13.61	24.50	0.855	0.183	8.42

Datasets	DeepBlending				BungeeNeRF			
Methods	psnr↑	ssim↑	lpips↓	size↓	psnr↑	ssim↑	lpips↓	size↓
3DGS	29.42	0.899	0.247	663.9	24.87	0.841	0.205	1616
Scaffold-GS	30.21	0.906	0.254	66.00	26.62	0.865	0.241	183.0
Lee *et al.*	29.79	0.901	0.258	43.21	23.36	0.788	0.251	82.60
Compressed3D	29.38	0.898	0.253	25.30	24.13	0.802	0.245	55.79
EAGLES	29.72	0.906	0.249	52.34	25.89	0.865	0.197	115.2
SizeGS	29.67	0.898	0.257	6.57	26.15	0.831	0.251	19.88
LightGaussian	27.01	0.872	0.308	33.94	24.52	0.825	0.255	87.28
SOG	29.12	0.892	0.270	5.70	22.43	0.708	0.339	48.25
Navaneet *et al.*	29.90	0.907	0.251	13.50	24.70	0.815	0.266	33.39
Reduced3DGS	29.63	0.902	0.249	18.00	24.57	0.812	0.228	65.39
CAT-3DGS Pro	29.82	0.901	0.255	5.98	26.29	0.836	0.248	18.52
RDOGaussian	29.63	0.902	0.252	18.00	23.37	0.762	0.286	39.06
HEMGS	29.96	0.904	0.253	5.51	26.43	0.840	0.241	17.34
MesonGS-FT	29.51	0.901	0.251	24.76	23.06	0.771	0.235	63.11
HAC (lr)	29.98	0.902	0.269	4.35	26.48	0.845	0.250	18.49
HAC (hr)	30.34	0.906	0.258	6.35	27.08	0.872	0.209	29.72
ContextGS (lr)	30.11	0.907	0.265	3.45	26.90	0.866	0.222	14.00
ContextGS (hr)	30.39	0.909	0.258	6.60	27.15	0.875	0.205	21.80
CompGS (lr)	29.30	0.895	0.293	6.03	-	-	-	-
CompGS (hr)	29.69	0.901	0.269	8.77	-	-	-	-
HAC++ (lr)	30.16	0.907	0.256	2.91	26.78	0.858	0.235	11.75
HAC++ (hr)	30.34	0.911	0.254	5.28	27.17	0.879	0.196	20.82
RacerGS (lr)	30.08	0.906	0.271	1.75	26.51	0.845	0.255	7.19
RacerGS (hr)	30.36	0.913	0.256	4.48	27.18	0.867	0.216	16.15

4.5.2 Training process

During training, we jointly optimize the rendering losses and the residual entropy loss, improving rendering quality while constraining the overall bitrate. The total training loss is defined as:

$$\mathcal{L}_{\text{total}} = (1 - \lambda_{\text{SSIM}})\,\mathcal{L}_1 + \lambda_{\text{SSIM}}\,\mathcal{L}_{\text{SSIM}} + \lambda_{\text{vol}}\,\mathcal{L}_{\text{vol}} + \frac{\lambda_{\text{r}}}{NN_p}(\mathcal{L}_{\Delta\text{entropy}} + \mathcal{L}_{\text{hash}}), \tag{14}$$

where λ_{r} is the rate weight for balancing the bitrate trade-off, N_p is the number of coded parameters per anchor, $\mathcal{L}_{\text{hash}}$ is the bit consumption of the binarized hash grid [5], and $\mathcal{L}_{\text{vol}} = \frac{1}{N_{\text{vis}}} \sum_{i=1}^{N_{\text{vis}}} \prod_{c=1}^{3} s_{i,c}$ [24] is the mean volume per Gaussian for N_{vis} visible Gaussians.

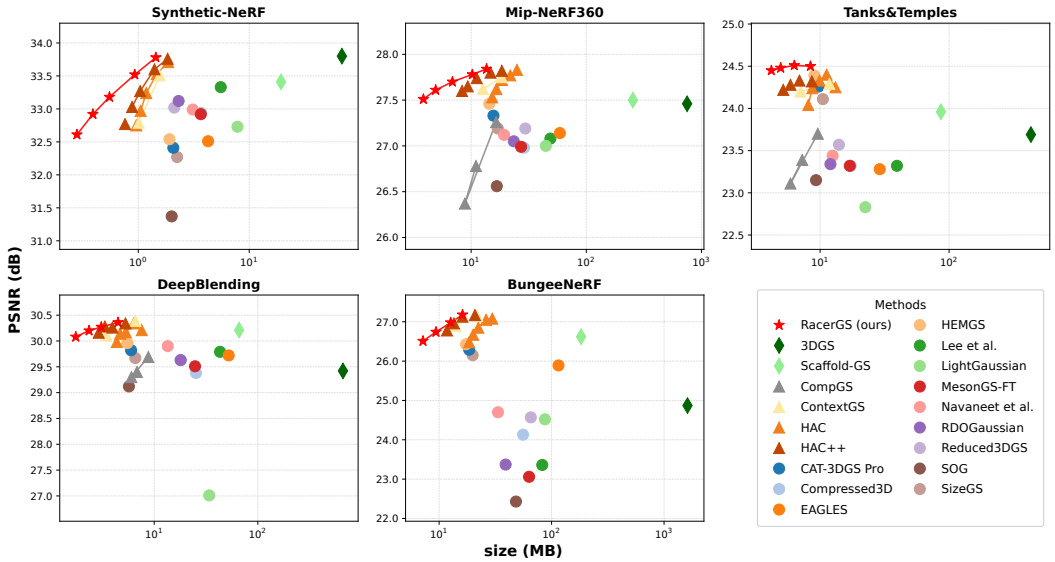

Figure 3: **RD curve comparisons.** RacerGS achieves better RD performance than the baseline methods across datasets.

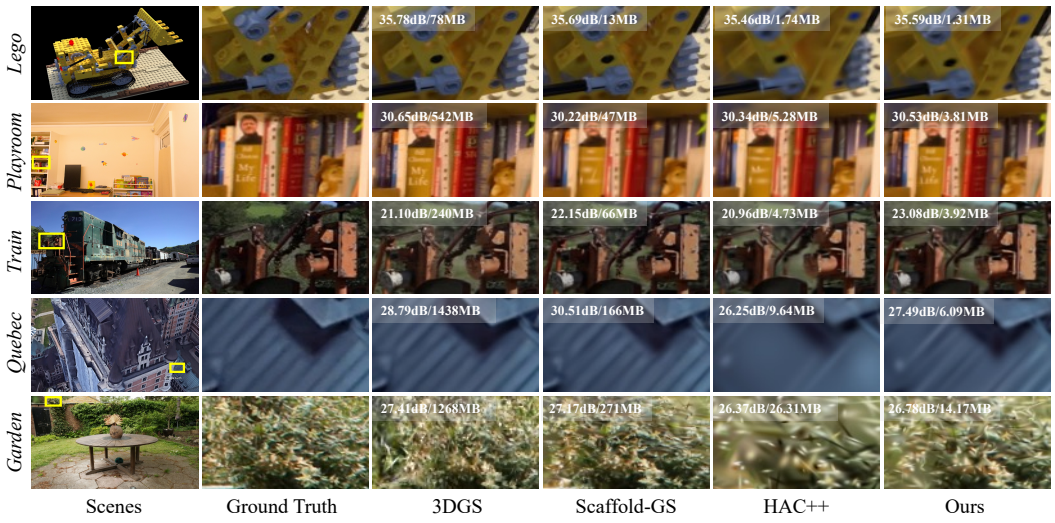

Figure 4: **Qualitative comparisons.** RacerGS preserves details better and produces fewer artifacts than the baseline methods across scenes from diverse datasets.

5 Experiments

5.1 Implementation

Datasets. We systematically conduct experiments on five comprehensive datasets, including one synthetic dataset, Synthetic-NeRF [25], and four real-world datasets: BungeeNeRF [44], DeepBlending [13], Mip-NeRF360 [1], and Tanks&Temples [16].

Metrics. We evaluate rendering quality with PSNR, SSIM [41], and LPIPS [50], and use storage size (MB) as the bitrate measure. Image metrics are averaged over test views and then over scenes in each dataset. Reported sizes include coded anchor positions, attributes, the hash grid, and masks, plus decoder-side network parameters and bounds; Table 2 gives the component breakdown. RD curves are obtained by varying the trade-off coefficient λ_r. We also report training, encoding, and decoding times, peak GPU memory, and rendering speed in FPS.

Table 2: **Storage sizes (MB) of different components** on Mip-NeRF360 (see Appendix 6 for results with additional λ_r values).

λ_r	**Total Size**	$\boldsymbol{x}^a$	$\boldsymbol{f}^a$	$\boldsymbol{l}$	$\boldsymbol{o}$	$\mathcal{H}$	$\boldsymbol{m}$	MLP
$4e$-3	3.85	0.19	1.89	0.80	0.48	0.05	0.10	0.34
$0.5e$-3	13.61	0.44	8.15	2.44	1.83	0.14	0.27	0.34

Table 3: **Encoding and decoding times** across all datasets.

Datasets	**Enc. Time (s)**		**Dec. Time (s)**	
	low rate	high rate	low rate	high rate
Synthetic-NeRF	0.63	1.28	0.89	1.75
Mip-NeRF360	4.24	8.59	7.40	14.20
Tanks&Temples	4.53	7.48	7.80	12.63
DeepBlending	1.85	3.01	4.03	6.25
BungeeNeRF	10.03	16.26	16.43	25.98

Table 4: **Training time, peak GPU memory, and FPS** on Mip-NeRF360.

Method	**Train (s)**↓	**Mem. (GB)**↓	**FPS**↑
3DGS	1607	–	102
Scaffold-GS	**1350**	15.8	142
HAC++	2447	17.4	146
RacerGS	2850	**13.9**	**171**

Baselines. We compare RacerGS with a diverse set of 3D Gaussian compression baselines, including codebook and pruning methods [17, 30, 28, 8, 31, 37, 46, 22, 49, 45], entropy- and sorting-based approaches [12, 26], and anchor-based methods [24, 5, 4, 23, 40].

Implementation details. All experiments are run on a single NVIDIA RTX A6000 GPU. Following prior work, we train for $30{,}000$ iterations. We obtain different bitrate operating points by varying the rate weight λ_r from 4×10^{-3} (low rate) to 0.5×10^{-3} (high rate; 1×10^{-3} on Tanks&Temples, DeepBlending, and BungeeNeRF). We set $\lambda_{\text{SSIM}} = 0.2$ and $\lambda_{\text{vol}} = 0.001$. The anchor feature dimension is $D^a = 50$, with $K = 10$ offsets per anchor. For causal entropy refinement, each anchor feature is split into $N^C = 5$ chunks. The base quantization step q_0 is set to 1, 0.001, and 0.2 for anchor features $\boldsymbol{f}^a$, scaling $\boldsymbol{l}$, and offsets $\boldsymbol{o}$, respectively.

5.2 Experimental results

Quantitative. As shown in Table 1, RacerGS achieves highly competitive compression while preserving rendering quality across all datasets. On average, it provides over $\mathbf{150}\times$ compression over vanilla 3DGS [15] and $\mathbf{28}\times$ over Scaffold-GS [24], while outperforming recent codebook/pruning [17, 30, 28, 8, 31, 37, 46], entropy/sorting-based [12, 26], and anchor-based [5, 4, 23] methods at smaller model sizes. At the high-rate Mip-NeRF360 point, it uses 13.61 MB versus HAC++'s 18.48 MB, with a PSNR of 27.84 versus 27.82 dB. Fig. 3 shows the RD behavior across the rate sweep. These gains come from the hash projection model, residual quantization, and FiLM-modulated autoregressive GRU context modeling, which jointly improve context representation and entropy estimation.

Qualitative. As shown in Fig. 4, RacerGS produces higher visual fidelity across diverse scenes, with fewer leaf artifacts in *garden*, clearer book title details in *playroom*, and sharper metal structures in *train*. Even at comparable or smaller sizes, it better preserves edges and material details while keeping backgrounds smooth. These comparisons highlight its ability to retain fine structures and textured regions under tight storage constraints.

Efficiency analysis. As shown in Table 4, RacerGS achieves the fastest rendering speed of **171 FPS** on Mip-NeRF360 and the lowest peak GPU memory. Encoding and decoding remain efficient across datasets, taking **0.63 s–16.26 s** and **0.89 s–25.98 s**, respectively (Table 3). Table 2 further shows that storage is dominated by anchor features, while the hash grid and MLP modules add only minor overhead, indicating effective entropy control and smooth bitrate allocation. Although RacerGS takes longer to train than the compared baselines, it offers lower peak memory usage and faster rendering, providing a favorable trade-off for applications with repeated scene rendering.

Table 5: **Ablation studies** on the Mip-NeRF360. The best results are in **bold**.

Ablation Items	PSNR ↑	SSIM ↑	LPIPS ↓	Size (MB) ↓	FPS ↑	Train (s) ↓
W/o residual quantization	27.83	0.810	**0.242**	18.24	146	2856
W/o hash projection	27.62	0.804	0.248	15.21	107	2865
Concat. context instead of GRU	27.81	0.806	0.244	16.87	156	**2841**
GRU context w/o FiLM	27.75	0.802	0.247	17.32	149	2854
RacerGS (full)	**27.84**	**0.811**	0.243	**13.61**	**171**	2850

Table 6: Controlled comparison of intra-anchor context models on Mip-NeRF360.

Context model	Ctx params	PSNR↑	Size↓
No intra-anchor context	0	27.69	18.20
Concat. (HAC++-style)	≈40k	27.81	16.87
Concat. (capacity-matched)	≈48k	27.82	15.24
GRU w/o FiLM	≈48k	27.75	17.32
GRU + FiLM (ours)	≈48k	**27.84**	**13.61**

Table 7: Anchor-feature bits per anchor by chunk index n on Mip-NeRF360.

Context model	n=1	n=2	n=3	n=4	n=5	Total
No intra-anchor context	68.0	80.5	86.0	90.5	95.5	420.5
Concat. (HAC++-style)	68.0	74.5	76.0	78.0	80.1	376.6
GRU + FiLM (ours)	68.0	**61.0**	**54.0**	**47.0**	**39.0**	**269.0**

5.3 Ablation study

Hash projection model. Removing the hash projection model degrades both quality and compression: PSNR drops from 27.84 to 27.62, and model size grows from 13.61 MB to 15.21 MB. Projecting multi-dimensional hash encodings into compact fused context features provides a stronger spatial prior and complementary cues for entropy parameter prediction. A single linear projection captures only part of the gain (27.76 dB / 14.28 MB). The context-dimension sweep in Appendix Table 8 varies D^c from 12 to 48 and changes size by at most 0.35 MB and PSNR by at most 0.04 dB, suggesting limited sensitivity within that range. Together, these results suggest that learning a structured context representation contributes more to the gains than the precise choice of context dimensionality.

Residual quantization. Without residual quantization, model size rises sharply from 13.61 MB to 18.24 MB, and PSNR slightly decreases, showing that centering anchor attributes around the predicted means effectively reduces residual entropy and dynamic range for more efficient entropy coding. The anchor-feature symbol analysis in Appendix Table 8 is consistent with that result: prediction-centered symbols have an empirical entropy of 5.22 rather than 8.27 bits per element, and the arithmetic-coded cost falls from 8.44 to 5.38 bits per element. The reduction in both quantities confirms that the lower symbol entropy translates into practical coding savings while preserving reconstruction quality.

Autoregressive GRU context model. Replacing the GRU with HAC++-style decoded-chunk concatenation under the same pipeline increases size from 13.61 to 16.87 MB at nearly equal PSNR (Table 6, Table 5). In Table 7, both cost 68 bits per anchor for the first chunk, before feature history is available. By the fifth chunk, the GRU costs 39.0 versus 80.1 bits per anchor for concatenation, consistent with a greater benefit from accumulated context. Removing FiLM raises size to 17.32 MB (Table 5), worse even than concatenation; recurrence and modulation work together in this configuration. By summarizing previously decoded feature groups through recurrent hidden states, the GRU supplies additional conditional information that lowers coding entropy and improves RD performance.

6 Conclusion

We presented RacerGS, a 3DGS compression framework that reduces storage while preserving rendering fidelity. RacerGS proposes compact hash-projected context features, adaptive residual quantization for lower entropy, and a FiLM-modulated autoregressive GRU context model to refine entropy estimates through causal anchor dependencies. Experiments and ablations demonstrate that RacerGS achieves substantially lower storage requirements than existing methods with comparable rendering quality across diverse datasets. Future work includes parallelizable context models and content-adaptive control to enable more efficient 3D rendering. By reducing the cost of storing and distributing 3D scenes, this work advances the goal of efficient visual computing: making richer representations of the world accessible with fewer resources.

References

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

[2] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *European Conference on Computer Vision*, pages 333–350. Springer, 2022.

[3] Yihang Chen, Qianyi Wu, Mehrtash Harandi, and Jianfei Cai. How far can we compress Instant-NGP-Based NeRF? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[4] Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. HAC: Hash-grid assisted context for 3D Gaussian splatting compression. In *European Conference on Computer Vision*, 2024.

[5] Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. HAC++: Towards 100x compression of 3D Gaussian splatting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(11):10210–10226, 2025. doi: 10.1109/TPAMI.2025.3594066.

[6] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[7] Chenxi Lola Deng and Enzo Tartaglione. Compressing explicit voxel grid representations: fast nerfs become also small. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1236–1245, 2023.

[8] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Light-Gaussian: Unbounded 3D Gaussian compression with 15x reduction and 200+ FPS. *Advances in neural information processing systems*, 2024.

[9] Guangchi Fang and Bing Wang. Mini-Splatting: Representing scenes with a constrained number of Gaussians. In *European Conference on Computer Vision*, 2024.

[10] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023.

[11] Sharath Girish, Abhinav Shrivastava, and Kamal Gupta. SHACIRA: Scalable HAsh-grid compression for implicit neural representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17513–17524, 2023.

[12] Sharath Girish, Kamal Gupta, and Abhinav Shrivastava. Eagles: Efficient accelerated 3D Gaussians with lightweight encodings. In *European Conference on Computer Vision*, 2024.

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

[14] Gyeongjin Kang, Younggeun Lee, Seungjun Oh, and Eunbyung Park. CodecNeRF: Toward fast encoding and decoding, compact, and high-quality novel-view synthesis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 4203–4211, 2025. doi: 10.1609/aaai.v39i4.32441.

[15] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian splatting for Real-Time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023.

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

[17] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3D Gaussian representation for radiance field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[18] Soonbin Lee, Fangwen Shu, Yago Sánchez, Thomas Schierl, and Cornelius Hellge. Compression of 3D Gaussian splatting with optimized feature planes and standard video codecs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 25496–25505, 2025. doi: 10.1109/ICCV51701.2025.02365.

[19] Lingzhi Li, Zhen Shen, Zhongshu Wang, Li Shen, and Liefeng Bo. Compressing volumetric radiance fields to 1 mb. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4222–4231, 2023.

[20] Lingzhi Li, Zhongshu Wang, Zhen Shen, Li Shen, and Ping Tan. Compact real-time radiance fields with neural codebook. In *ICME*, 2023.

[21] Sicheng Li, Hao Li, Yiyi Liao, and Lu Yu. NeRFCodec: Neural feature compression meets neural radiance fields for Memory-Efficient scene representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21274–21283, 2024.

[22] Lei Liu, Zhenghao Chen, Wei Jiang, Wei Wang, and Dong Xu. HEMGS: A hybrid entropy model for 3D Gaussian splatting data compression. *arXiv preprint arXiv:2411.18473*, 2024.

[23] Xiangrui Liu, Xinju Wu, Pingping Zhang, Shiqi Wang, Zhu Li, and Sam Kwong. Compgs: Efficient 3D scene representation via compressed Gaussian splatting. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 2936–2944, 2024.

[24] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-GS: Structured 3D Gaussians for View-Adaptive rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

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

[26] Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. Compact 3D scene representation via Self-Organizing Gaussian grids. In *European Conference on Computer Vision (ECCV)*, pages 18–34. Springer, 2024. doi: 10.1007/978-3-031-73013-9_2.

[27] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4): 1–15, 2022.

[28] KL Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. Compact3d: Compressing Gaussian splat radiance field models with vector quantization. In *European Conference on Computer Vision*, 2024.

[29] KL Navaneet, Kossar Pourahmadi Meibodi, Soroush Abbasi Koohpayegani, and Hamed Pirsiavash. Compgs: Smaller and faster Gaussian splatting with vector quantization. In *European Conference on Computer Vision*, pages 330–349. Springer, 2024.

[30] Simon Niedermayr, Josef Stumpfegger, and Rüdiger Westermann. Compressed 3D Gaussian splatting for accelerated novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10349–10358, 2024. doi: 10.1109/CVPR52733.2024.00985.

[31] Panagiotis Papantonakis, Georgios Kopanas, Bernhard Kerbl, Alexandre Lanvin, and George Drettakis. Reducing the memory footprint of 3D Gaussian splatting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 7(1):1–17, 2024.

[32] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11671.

[33] Daniel Rho, Byeonghyeon Lee, Seungtae Nam, Joo Chan Lee, Jong Hwan Ko, and Eunbyung Park. Masked wavelet representation for compact neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20680–20690, 2023.

[34] Seungjoo Shin and Jaesik Park. Binary radiance fields. *Advances in neural information processing systems*, 2023.

[35] Zetian Song, Jiaqi Zhang, Wenhong Duan, Yuhuai Zhang, Xinfeng Zhang, and Siwei Ma. SPC-NeRF: Spatial predictive compression for voxel-based radiance field. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 22(4):1–22, 2026. doi: 10.1145/3795528.

[36] Jiaxiang Tang, Xiaokang Chen, Jingbo Wang, and Gang Zeng. Compressible-composable nerf via rank-residual decomposition. *Advances in Neural Information Processing Systems*, 35: 14798–14809, 2022.

[37] Henan Wang, Hanxin Zhu, Tianyu He, Runsen Feng, Jiajun Deng, Jiang Bian, and Zhibo Chen. End-to-end rate-distortion optimized 3D Gaussian representation. In *European Conference on Computer Vision*, pages 76–92. Springer, 2025.

[38] Wenqing Wang and Yun Fu. Emogene: Audio-driven emotional 3d talking-head generation. In *2025 IEEE 19th International Conference on Automatic Face and Gesture Recognition (FG)*, pages 1–10, 2025. doi: 10.1109/FG61629.2025.11099460.

[39] Wenqing Wang and Yun Fu. Realtalk: Realistic emotion-aware lifelike talking-head synthesis. In *2025 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 1492–1501, 2025. doi: 10.1109/ICCVW69036.2025.00159.

[40] Yufei Wang, Zhihao Li, Lanqing Guo, Wenhan Yang, Alex C Kot, and Bihan Wen. ContextGS: Compact 3D Gaussian splatting with anchor level context model. *Advances in neural information processing systems*, 2024.

[41] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4): 600–612, 2004.

[42] Ian H Witten, Radford M Neal, and John G Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

[43] Minye Wu and Tinne Tuytelaars. Implicit Gaussian splatting with efficient Multi-Level Tri-Plane representation. *arXiv preprint arXiv:2408.10041*, 2024.

[44] Yuanbo Xiangli, Linning Xu, Xingang Pan, Nanxuan Zhao, Anyi Rao, Christian Theobalt, Bo Dai, and Dahua Lin. Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering. In *European conference on computer vision*, pages 106–122. Springer, 2022.

[45] Shuzhao Xie, Jiahang Liu, Weixiang Zhang, Shijia Ge, Sicheng Pan, Chen Tang, Yunpeng Bai, Cong Zhang, Xiaoyi Fan, and Zhi Wang. Sizegs: Size-aware compression of 3D Gaussian splatting via mixed integer programming. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pages 8214–8223, 2025.

[46] Shuzhao Xie, Weixiang Zhang, Chen Tang, Yunpeng Bai, Rongwei Lu, Shijia Ge, and Zhi Wang. Mesongs: Post-training compression of 3D Gaussians via efficient attribute transformation. In *European Conference on Computer Vision*, pages 434–452. Springer, 2025.

[47] Runyi Yang, Zhenxin Zhu, Zhou Jiang, Baijun Ye, Xiaoxue Chen, Yifei Zhang, Yuantao Chen, Jian Zhao, and Hao Zhao. Spectrally pruned Gaussian fields with neural compensation. *arXiv preprint arXiv:2405.00676*, 2024.

[48] Yu-Ting Zhan, Cheng-Yuan Ho, Hebi Yang, Yi-Hsin Chen, Jui Chiu Chiang, Yu-Lun Liu, and Wen-Hsiao Peng. CAT-3DGS: A context-adaptive triplane approach to rate-distortion-optimized 3DGS compression, 2025. URL `https://arxiv.org/abs/2503.00357`.

[49] Yu-Ting Zhan, He-bi Yang, Cheng-Yuan Ho, Jui-Chiu Chiang, and Wen-Hsiao Peng. Cat-3dgs pro: A new benchmark for efficient 3dgs compression. In *2025 33rd European Signal Processing Conference (EUSIPCO)*, pages 1367–1371. IEEE, 2025.

[50] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

A Entropy coding

Entropy coding is a lossless compression technique that represents discrete symbols with code lengths proportional to their information content. Given a discrete random variable Y with probability mass function $p(y)$, the optimal code length ℓ for a symbol y satisfies $\ell(y) \approx -\log_2 p(y)$, and the expected code length is bounded below by the Shannon entropy $\mathbb{E}[\ell(Y)] \geq H(Y) = -\sum_y p(y) \log_2 p(y)$ [6]. Entropy coders such as arithmetic coding [42] achieve expected lengths arbitrarily close to $H(Y)$ when the model probability p matches the true symbol distribution.

B Hash-grid-assisted design

HAC [4] introduces an anchor-based 3DGS method that reduces storage size by leveraging spatial consistency through hash features $\boldsymbol{f}^h = \mathrm{Interp}(\boldsymbol{x}^a, \mathcal{H})$ queried at the anchor position $\boldsymbol{x}^a$. The hash grid for dimension $d \in \{3\mathrm{D}, 2\mathrm{D}\}$ is defined as $\mathcal{H}^d = \left\{ \boldsymbol{\Theta}_d^l \in \mathbb{R}^{T_d^l \times d_h} \right\}_{l=1}^{L_d}$, where Θ is the hash embedding, T is the table size, and l indexes the L multi-resolution levels. Although this hash-grid-assisted design achieves strong performance, substantial redundancy remains due to the spatial consistency of anchor attribute residuals and causal dependencies among anchor feature groups. We further exploit this spatial consistency and capture these causal dependencies through recurrent hidden states to produce a more compact 3DGS representation.

C Per-component coding time

We analyze the per-component coding time of RacerGS on the Mip-NeRF360 dataset in Tables 10 and 11. Overall, the total coding time increases as the rate weight λ_r decreases, since a smaller λ_r produces more anchors and representations with higher bitrates. The anchor feature component $\boldsymbol{f}^a$ dominates both encoding and decoding time, accounting for about $46\% \sim 51\%$ of encoding time and $50\% \sim 53\%$ of decoding time, because it contains the largest number of parameters and requires entropy modeling with the autoregressive GRU context model. The scaling parameters $\boldsymbol{l}$ and offsets $\boldsymbol{o}$ account for the next largest portion of the coding time, while the anchor positions $\boldsymbol{x}^a$ contribute only a small fraction of the overall computation. The remaining components introduce negligible overhead due to their compact representations. These results indicate that the computational cost is mainly concentrated in the entropy coding of anchor attributes, while the other components introduce minimal overhead, demonstrating that RacerGS maintains efficient coding complexity.

D Quantitative per-scene results

Per-scene results for the Synthetic-NeRF dataset are shown in Table 13. Per-scene results for the Mip-NeRF360 dataset are shown in Table 14. Per-scene results for the Tanks&Temples dataset are shown in Table 15. Per-scene results for the DeepBlending dataset are shown in Table 16. Per-scene results for the BungeeNeRF dataset are shown in Table 17.

E Additional analyses

Table 8 extends the ablation study on Mip-NeRF360 (hr): (a) a nonlinear projection outperforms a single linear layer; (b) performance remains stable across a $4\times$ range of context dimensions D^c, and halving the default dimension increases size by only 2.6%; (c) residual coding reduces both the anchor-feature alphabet and its coding cost; and (d) the standard deviations across three seeds are 0.02 dB and 0.10 MB for PSNR and size, respectively. Table 9 shows that decoding time grows with scene size. The recurrence is sequential only across the $N^C = 5$ chunks within each anchor, while chunk n is decoded for all anchors in a single batched step. Further speedups are possible through improvements to the arithmetic codec, such as chunk-parallel entropy coding.

Table 8: **Design analyses** on Mip-NeRF360. (a) Entropy-context projection. (b) Sensitivity to context dimension D^c. (c) Anchor-feature symbol statistics under attribute-lattice and residual coding. (d) Variability across three seeds (mean $\pm$ std).

(a) Projection

	PSNR (dB)	Size (MB)
None (raw $\boldsymbol{f}^h$)	27.62	15.21
Linear	27.76	14.28
MLP (ours)	**27.84**	**13.61**

(b) Context dimension

D^c	PSNR (dB)	Size (MB)
12	27.80	13.96
24	**27.84**	**13.61**
48	27.82	13.78

(c) Anchor-feature symbols

	Attribute	**Residual**
Symbol range	1,023	**287**
Empirical entropy (bits)	8.27	**5.22**
Coded bits / element	8.44	**5.38**

(d) Three seeds

Method	PSNR (dB)	Size (MB)
RacerGS	27.84 ± 0.02	13.63 ± 0.10

Table 9: **Decoding time breakdown**. Arithmetic coding accounts for about 60% of decoding time across all datasets, with the rest spent on network evaluation.

Dataset	**Total (s)**	**Codec (s)**	**Network (s)**
Synthetic-NeRF	1.75	1.08	0.67
Mip-NeRF360	14.20	8.78	5.42
BungeeNeRF	25.98	16.07	9.91

F Limitations

While RacerGS achieves substantial compression gains and high rendering fidelity, it has several limitations. The FiLM-modulated autoregressive GRU context model decodes anchor feature groups sequentially, which limits its applicability to real-time streaming scenarios. Additionally, like other anchor-based 3DGS methods, RacerGS is optimized per scene, so generalization to unseen scenes still requires retraining. Addressing these limitations through parallelizable context modeling and cross-scene generalization is a promising direction for future work.

G Societal impact

Positive impacts. RacerGS substantially reduces the storage and transmission costs of high-fidelity 3D scene representations, which can lower the bandwidth and energy footprint of immersive applications such as virtual and augmented reality, telepresence, digital cultural heritage preservation, robotics simulation, and remote collaboration. By making compact 3D content more accessible on edge and mobile devices, our method can help democratize high-quality 3D media for users with limited compute or network resources.

Negative impacts. As a general-purpose 3D scene compression framework, RacerGS could be misused to more efficiently store, distribute, or stream 3D reconstructions of private spaces or individuals captured without consent, raising privacy and surveillance concerns. The lower bitrate also reduces the cost of disseminating photorealistic 3D content that may be repurposed for misleading or deceptive media. We encourage downstream practitioners to follow established consent, watermarking, and provenance practices, and to combine RacerGS with appropriate access control when deploying it on sensitive content.

Table 10: Encoding time for different components on the Mip-NeRF360 dataset.

λ_r	**Total Time (s)**	$\boldsymbol{x}^a$	$\boldsymbol{f}^a$	$\boldsymbol{l}$	$\boldsymbol{o}$	$\mathcal{H}$	$\boldsymbol{m}$	Others
$4e$-3	4.24	0.72 (17%)	1.93 (46%)	0.93 (22%)	0.45 (11%)	0.01 (0%)	0.01 (0%)	0.19 (4%)
$3e$-3	5.36	0.83 (15%)	2.59 (48%)	1.18 (22%)	0.56 (10%)	0.01 (0%)	0.01 (0%)	0.18 (3%)
$2e$-3	6.57	0.94 (14%)	3.25 (49%)	1.39 (21%)	0.69 (11%)	0.01 (0%)	0.01 (0%)	0.28 (4%)
$1e$-3	7.61	1.01 (13%)	3.88 (51%)	1.55 (20%)	0.82 (11%)	0.01 (0%)	0.01 (0%)	0.33 (4%)
$0.5e$-3	8.59	1.07 (12%)	4.40 (51%)	1.74 (20%)	0.96 (11%)	0.01 (0%)	0.01 (0%)	0.40 (5%)

Table 11: Decoding time for different components on the Mip-NeRF360 dataset.

λ_r	**Total Time (s)**	$\boldsymbol{x}^a$	$\boldsymbol{f}^a$	$\boldsymbol{l}$	$\boldsymbol{o}$	$\mathcal{H}$	$\boldsymbol{m}$	Others
$4e$-3	7.40	0.28 (4%)	3.69 (50%)	2.30 (31%)	0.94 (13%)	0.01 (0%)	0.01 (0%)	0.17 (2%)
$3e$-3	9.15	0.31 (3%)	4.71 (51%)	2.79 (30%)	1.21 (13%)	0.01 (0%)	0.01 (0%)	0.11 (1%)
$2e$-3	10.98	0.34 (3%)	5.72 (52%)	3.21 (29%)	1.54 (14%)	0.01 (0%)	0.01 (0%)	0.15 (1%)
$1e$-3	12.63	0.37 (3%)	6.65 (53%)	3.54 (28%)	1.82 (14%)	0.01 (0%)	0.01 (0%)	0.23 (2%)
$0.5e$-3	14.20	0.40 (3%)	7.55 (53%)	3.86 (27%)	1.98 (14%)	0.01 (0%)	0.01 (0%)	0.39 (3%)

Table 12: Storage size (MB) of different components on the Mip-NeRF360 dataset.

λ_r	**Total Size**	$\boldsymbol{x}^a$	$\boldsymbol{f}^a$	$\boldsymbol{l}$	$\boldsymbol{o}$	$\mathcal{H}$	$\boldsymbol{m}$	MLP
$4e$-3	3.85	0.19	1.89	0.80	0.48	0.05	0.10	0.34
$3e$-3	4.92	0.20	2.57	1.01	0.63	0.06	0.11	0.34
$2e$-3	6.88	0.26	3.81	1.37	0.88	0.08	0.14	0.34
$1e$-3	10.24	0.35	5.94	1.93	1.35	0.10	0.23	0.34
$0.5e$-3	13.61	0.44	8.15	2.44	1.83	0.14	0.27	0.34

Table 13: RacerGS results on the Synthetic-NeRF dataset for different λ_r values.

λ_r	Scenes	PSNR ↑	SSIM ↑	LPIPS ↓	Size (MB) ↓
$4e$-3	chair	33.54	0.960	0.025	0.28
	drums	26.75	0.950	0.032	0.31
	ficus	34.36	0.982	0.018	0.20
	hotdog	35.65	0.971	0.033	0.24
	lego	32.76	0.966	0.029	0.36
	materials	30.21	0.950	0.025	0.31
	mic	34.73	0.992	0.015	0.24
	ship	32.90	0.937	0.147	0.27
$3e$-3	chair	34.10	0.966	0.021	0.38
	drums	26.59	0.950	0.033	0.42
	ficus	34.66	0.983	0.017	0.32
	hotdog	35.90	0.974	0.029	0.37
	lego	33.26	0.969	0.026	0.48
	materials	30.61	0.953	0.024	0.42
	mic	35.10	0.992	0.013	0.33
	ship	33.14	0.940	0.141	0.40
$2e$-3	chair	34.47	0.971	0.019	0.54
	drums	26.48	0.949	0.033	0.60
	ficus	34.78	0.984	0.015	0.49
	hotdog	36.21	0.977	0.027	0.52
	lego	33.76	0.971	0.024	0.65
	materials	30.92	0.955	0.022	0.60
	mic	35.43	0.991	0.012	0.47
	ship	33.39	0.946	0.136	0.53
$1e$-3	chair	34.83	0.974	0.017	0.93
	drums	26.32	0.946	0.034	1.02
	ficus	34.82	0.981	0.015	0.88
	hotdog	36.78	0.982	0.026	0.91
	lego	34.42	0.971	0.022	1.01
	materials	31.51	0.954	0.021	1.04
	mic	35.72	0.985	0.016	0.56
	ship	33.76	0.959	0.129	1.08
$0.5e$-3	chair	34.83	0.981	0.025	1.40
	drums	26.32	0.950	0.035	1.48
	ficus	34.82	0.984	0.015	1.38
	hotdog	36.78	0.993	0.025	1.33
	lego	35.59	0.955	0.021	1.31
	materials	32.42	0.943	0.024	1.48
	mic	35.72	0.980	0.017	0.61
	ship	33.76	0.973	0.121	2.54
$4e$-3	**Avg.**	32.61	0.964	0.041	0.28
$3e$-3	**Avg.**	32.92	0.966	0.038	0.39
$2e$-3	**Avg.**	33.18	0.968	0.036	0.55
$1e$-3	**Avg.**	33.52	0.969	0.035	0.93
$0.5e$-3	**Avg.**	33.78	0.970	0.035	1.44

Table 14: RacerGS results on the Mip-NeRF360 dataset for different λ_r values.

λ_r	Scenes	PSNR ↑	SSIM ↑	LPIPS ↓	Size (MB) ↓
$4e$-3	bicycle	24.19	0.743	0.292	5.42
	bonsai	32.00	0.927	0.188	2.87
	counter	29.14	0.884	0.261	2.50
	flowers	24.02	0.807	0.213	4.09
	garden	26.20	0.759	0.295	5.14
	kitchen	31.51	0.905	0.157	2.73
	room	31.71	0.901	0.195	1.61
	stump	25.61	0.694	0.396	5.43
	treehill	23.17	0.586	0.362	4.86
$3e$-3	bicycle	24.39	0.746	0.286	6.90
	bonsai	32.08	0.929	0.186	3.48
	counter	29.20	0.888	0.252	3.11
	flowers	24.34	0.809	0.209	5.18
	garden	26.29	0.763	0.289	6.58
	kitchen	31.54	0.907	0.153	3.36
	room	31.85	0.905	0.189	1.95
	stump	25.60	0.698	0.382	7.00
	treehill	23.20	0.590	0.356	6.72
$2e$-3	bicycle	24.58	0.748	0.278	9.74
	bonsai	32.14	0.930	0.186	4.96
	counter	29.32	0.892	0.244	4.50
	flowers	24.72	0.810	0.205	7.14
	garden	26.37	0.769	0.276	9.27
	kitchen	31.51	0.908	0.149	4.74
	room	31.98	0.908	0.184	2.81
	stump	25.52	0.702	0.372	9.84
	treehill	23.16	0.596	0.347	8.92
$1e$-3	bicycle	24.72	0.748	0.272	16.60
	bonsai	32.17	0.929	0.193	9.82
	counter	29.43	0.895	0.236	7.85
	flowers	25.20	0.810	0.202	9.60
	garden	26.38	0.782	0.259	12.63
	kitchen	31.43	0.905	0.150	7.08
	room	32.23	0.911	0.183	5.32
	stump	25.40	0.702	0.366	11.89
	treehill	23.06	0.599	0.344	11.37
$0.5e$-3	bicycle	24.72	0.754	0.257	22.33
	bonsai	32.17	0.903	0.193	19.86
	counter	29.43	0.905	0.212	15.42
	flowers	25.31	0.798	0.266	10.23
	garden	26.78	0.800	0.237	14.17
	kitchen	31.43	0.910	0.140	8.96
	room	32.26	0.917	0.206	6.34
	stump	25.40	0.709	0.350	12.65
	treehill	23.06	0.605	0.328	12.57
$4e$-3	**Avg.**	27.51	0.801	0.262	3.85
$3e$-3	**Avg.**	27.61	0.804	0.256	4.92
$2e$-3	**Avg.**	27.70	0.807	0.249	6.88
$1e$-3	**Avg.**	27.78	0.809	0.245	10.24
$0.5e$-3	**Avg.**	27.84	0.811	0.243	13.61

Table 15: RacerGS results on the Tanks&Temples dataset for different λ_r values.

λ_r	Scenes	PSNR ↑	SSIM ↑	LPIPS ↓	Size (MB) ↓
$4e$-3	truck	25.82	0.861	0.178	4.46
	train	23.08	0.833	0.212	3.92
$3e$-3	truck	25.88	0.868	0.171	5.28
	train	23.08	0.836	0.208	4.62
$2e$-3	truck	25.94	0.876	0.166	6.71
	train	23.08	0.838	0.205	5.91
$1e$-3	truck	25.92	0.885	0.158	8.90
	train	23.08	0.825	0.208	7.94
$4e$-3	**Avg.**	24.45	0.847	0.195	4.19
$3e$-3	**Avg.**	24.48	0.852	0.190	4.95
$2e$-3	**Avg.**	24.51	0.857	0.186	6.31
$1e$-3	**Avg.**	24.50	0.855	0.183	8.42

Table 16: RacerGS results on the DeepBlending dataset for different λ_r values.

λ_r	Scenes	PSNR ↑	SSIM ↑	LPIPS ↓	Size (MB) ↓
$4e$-3	playroom	30.02	0.904	0.268	1.52
	drjohnson	30.13	0.907	0.274	1.97
$3e$-3	playroom	30.12	0.906	0.261	2.02
	drjohnson	30.28	0.910	0.265	2.68
$2e$-3	playroom	30.22	0.908	0.258	2.64
	drjohnson	30.32	0.910	0.260	3.52
$1e$-3	playroom	30.53	0.913	0.256	3.81
	drjohnson	30.18	0.912	0.255	5.14
$4e$-3	**Avg.**	30.08	0.906	0.271	1.75
$3e$-3	**Avg.**	30.20	0.908	0.263	2.35
$2e$-3	**Avg.**	30.27	0.909	0.259	3.08
$1e$-3	**Avg.**	30.36	0.913	0.256	4.48

Table 17: RacerGS results on the BungeeNeRF dataset for different λ_r values.

λ_r	Scenes	PSNR ↑	SSIM ↑	LPIPS ↓	Size (MB) ↓
$4e$-3	amsterdam	26.75	0.854	0.239	8.23
	bilbao	26.32	0.878	0.251	7.32
	hollywood	26.52	0.875	0.358	8.03
	pompidou	26.73	0.822	0.255	6.27
	quebec	27.49	0.809	0.175	6.09
	rome	25.25	0.831	0.252	7.17
$3e$-3	amsterdam	27.01	0.870	0.215	10.42
	bilbao	26.78	0.888	0.228	8.95
	hollywood	26.38	0.876	0.338	9.21
	pompidou	26.84	0.840	0.241	8.05
	quebec	27.86	0.822	0.162	8.01
	rome	25.55	0.846	0.233	11.64
$2e$-3	amsterdam	27.28	0.872	0.203	13.82
	bilbao	26.97	0.886	0.217	11.35
	hollywood	26.42	0.874	0.322	12.17
	pompidou	26.91	0.848	0.235	11.58
	quebec	28.36	0.826	0.156	10.78
	rome	25.94	0.857	0.223	16.32
$1e$-3	amsterdam	27.28	0.876	0.179	17.45
	bilbao	27.03	0.890	0.182	14.23
	hollywood	26.42	0.767	0.313	15.15
	pompidou	26.91	0.864	0.256	17.33
	quebec	28.75	0.967	0.152	15.04
	rome	26.69	0.840	0.212	17.71
$4e$-3	**Avg.**	26.51	0.845	0.255	7.19
$3e$-3	**Avg.**	26.74	0.857	0.236	9.38
$2e$-3	**Avg.**	26.98	0.861	0.226	12.67
$1e$-3	**Avg.**	27.18	0.867	0.216	16.15

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the RacerGS components and compression claims, and the experiments in Sec. 5 report the corresponding rate-distortion, quality, and efficiency results.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: It discusses remaining directions and current limitations of the approach.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper proposes an empirical compression method and does not present formal theoretical results or proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper method section describes the model and losses, and the experiment section specifies datasets, baselines, metrics, training iterations, hyperparameters, and hardware.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The paper provides open access to the public datasets.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Sec. 5.1 lists the datasets, baselines, metrics, training schedule, RD weights, loss weights, feature dimensions, chunk count, and quantization steps.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper reports error bars and relevant statistics.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Sec. 5.1 reports use of a single NVIDIA RTX A6000 GPU, and Sec. 5.2 plus the supplement report training, encoding, decoding, and FPS measurements.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses standard public datasets and compression experiments, and we are not aware of deviations from the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses both potential positive societal impacts and negative societal impacts of the work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.

- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The work does not include high-risk generative models, language models, or scraped datasets requiring special safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Existing datasets, baselines, metrics, and prior methods are credited through citations, and the licenses are properly respected.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper introduces a method but does not include a new asset as part of the submission.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The research does not involve crowdsourcing experiments or human-subject studies.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The research does not involve crowdsourcing experiments or human-subject studies, so IRB approval is not applicable.

    Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core methodology does not use LLMs as an important, original, or non-standard research component.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.