MITIGATING SPATIAL REDUNDANCY: A PREDICTIVE COMPRESSION FRAMEWORK FOR 3D GAUSSIAN SPLATTING

Anonymous authors
Paper under double-blind review

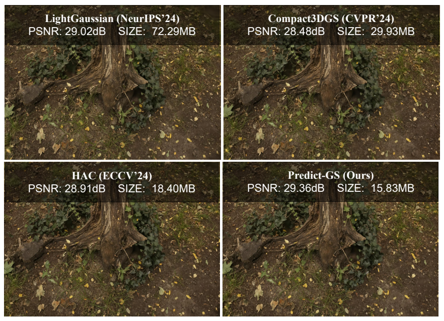
ABSTRACT

3D Gaussian Splatting (3DGS) has emerged as a promising framework for Novel View Synthesis (NVS) due to its superior rendering quality and real-time performance. However, its widespread adoption is hindered by the substantial storage and transmission costs associated with the massive number of primitives. Notably, existing 3DGS compression approaches encode every primitive in its entirety, failing to utilize spatial continuity to compress shared content across primitives. In this work, we propose **Predict-GS**, a predictive compression framework for anchor-based Gaussian to mitigate spatial redundancy among anchors. Specifically, we construct a Spatial Feature Pool (SFP) based on a hybrid representation of multi-resolution 3D grids and 2D planes, which serves to predict coarse Gaussians for scene reconstruction. To refine these predictions, we introduce a residual compensation module equipped with a Multi-head Gaussian Residual Decoder (MGRD) that models corrections for shape and appearance, thereby transforming coarse Gaussians into high-fidelity ones. Furthermore, we revisit the inherent characteristics of our framework and design a prediction-tailored progressive training strategy to enhance its effectiveness. Extensive experiments on public benchmarks demonstrate the effectiveness of our framework, achieving a remarkable size reduction of over 58× compared to vanilla 3DGS on Mip-NeRF360 and outperforming the state-of-the-art (SOTA) compression method.

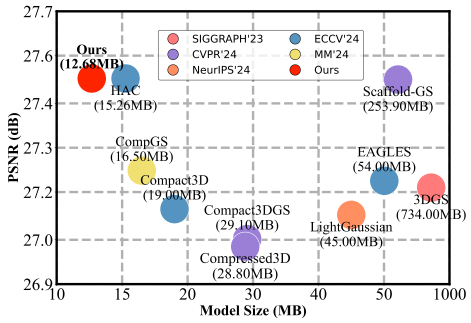
1 INTRODUCTION

Novel View Synthesis (NVS) plays a pivotal role in Augmented Reality (AR) Franke et al. (2024), Virtual Reality (VR) Zhai et al. (2025) and Mixed Reality (MR) Speicher et al. (2019). Approaches based on implicit Neural Radiance Fields (NeRF) Mildenhall et al. (2021) have demonstrated remarkable capabilities in generating photorealistic novel views by accumulating RGB values along sampled rays. However, NeRF-based methods Mildenhall et al. (2021); Tosi et al. (2023) heavily rely on repeated queries to a Multi-Layer Perceptron (MLP), resulting in substantial computational overhead and prohibitive training and rendering time. Recently, 3D Gaussian Splatting (3DGS) Kerbl et al. (2023) has emerged as a compelling alternative, which models scenes using anisotropic 3D Gaussian primitives with learnable attributes. Due to the high fidelity and real-time performance, 3DGS has inspired a series of works Huang et al. (2024); Guédon & Lepetit (2024); Zhang et al. (2024); Ye et al. (2024); Kheradmand et al. (2025); Wu et al. (2024); Yang et al. (2024b); Liang et al. (2024); Tang et al. (2023); Wu et al. (2025); Yan et al. (2024); Rota Bulò et al. (2024); Lu et al. (2024a); Yu et al. (2024b;a); Yang et al. (2024a) in NVS community and is widely considered a promising representation for 3D reconstruction.

However, 3DGS requires storing a large number of 3D Gaussian primitives along with their complex attributes, resulting in substantial storage overhead—often reaching several hundred megabytes or even exceeding 1 gigabyte per scene Bagdasarian et al. (2024). The prohibitive storage and transmission costs pose challenges to the deployment in scenarios involving resource-constrained devices or real-time streaming requirements. Existing 3DGS compression methods can be broadly classified into four primary strategies: **(a)** Designing compact representation Lu et al. (2024b); Ren et al. (2024), **(b)** Pruning redundant Gaussian primitives Mallick et al. (2024); Fang & Wang (2024a;b); Lee et al. (2024), **(c)** Adaptive quantization Lee et al. (2024); Fan et al. (2024), and


LightGaussian (NeurIPS'24)
PSNR: 29.02dB SIZE: 72.29MB
Compact3DGS (CVPR'24)
PSNR: 28.48dB SIZE: 29.93MB
HAC (ECCV'24)
PSNR: 28.91dB SIZE: 18.40MB
Predict-GS (Ours)
PSNR: 29.36dB SIZE: 15.83MB

(a)



(b)

Figure 1: **(a)** presents an intuitive comparison between our method and representative 3DGS compression methods, LightGaussian Fan et al. (2024), Compact3DGS Lee et al. (2024) and HAC Chen et al. (2024b) on *stump* scene in Mip-NeRF360 Barron et al. (2022) dataset. **(b)** compares the performance of representative methods in the field of 3DGS compression. Methods positioned closer to the top-left corner indicate better compression performance. The results are evaluated on Mip-NeRF360 Barron et al. (2022) dataset. Our method compresses vanilla 3DGS by 58× and even achieves a 16.91% bit rate reduction compared to the state-of-the-art.

(d) Eliminating statistical redundancy across Gaussian attributes, typically through learned entropy models Chen et al. (2024b); Wang et al. (2024); Liu et al. (2024a); Chen et al. (2025) for entropy coding. Despite these advances, the compression performance requires further improvement to support real-time transmission and deployment on storage-constrained devices.

Existing 3DGS compression methods Lee et al. (2024); Niedermayr et al. (2024); Fan et al. (2024); Girish et al. (2024); Liu et al. (2024c); Navaneet et al. (2024); Chen et al. (2024b; 2025); Wang et al. (2024); Liu et al. (2024a) are fundamentally grounded in a post-compression paradigm, where techniques such as quantization and entropy coding are applied after scene reconstruction. A notable limitation is that these post-compression methods have to encode each primitive as a whole, without exploiting the inherent spatial redundancy across them, since each primitive is treated as an independent entity during the reconstruction stage Kerbl et al. (2023); Lu et al. (2024b). In practice, however, 3D scenes exhibit continuous spatial structures, where nearby primitives often possess highly correlated attributes. This implies that, in post-compression methods, shared features across primitives are redundantly stored and encoded. While Scaffold-GS Lu et al. (2024b) proposes using anchors to aggregate Gaussians and thereby reduce spatial redundancy among Gaussians within a local neighborhood, it is important to note that the aggregation remains fine-grained and does not capture large-scale context. As a result, **spatial redundancy at anchor level still persists**.

Prediction techniques have been employed in image compression Minnen et al. (2018); Ma et al. (2019) to effectively reduce spatial redundancy among neighboring pixels. Specifically, rather than encoding every pixel individually, only some pixels are explicitly encoded while the others are predicted from the former. However, predictive strategies remain unexplored in 3DGS compression due to the following challenges: **(a)** The spatial structure of Gaussians or anchors is sparse and irregular, lacking the well-defined neighborhood patterns found in images that are essential for prediction-based techniques. **(b)** Forcibly constructing such neighborhoods introduces significant computational overhead. **(c)** Gaussians or anchors move continuously during the training process, which causes dynamic neighborhood changes, further complicating the prediction process. For the reasons above, it is challenging to build a prediction-driven compression framework for 3DGS.

In this work, we propose a predictive compression framework (**Predict-GS**) for anchor-based Gaussian Lu et al. (2024b), as it offers a more compact structure and serves as the foundation for many representative 3DGS compression methods Chen et al. (2024b); Ren et al. (2024); Wang et al. (2024); Ma et al. (2025); Chen et al. (2025). Specifically, we construct a lightweight Spatial Feature Pool (SFP) to compactly store shared features for representing the 3D scene, enabling each anchor to query the spatial feature at its location for direct Gaussian primitive prediction. However, the predicted Gaussians struggle to accurately capture fine-grained details and high-frequency textures in the scene. To address this issue, we introduce a residual compensation strategy, which adjusts the predicted Gaussians' shape and appearance by adding correction terms. By integrating Gaussian prediction with residual compensation, we achieve spatial redundancy elimination for anchor feature, i.e., instead of encoding the complete feature for every anchor, we only retain their residual. Moreover, we observe that the predictive compression framework suffers from suboptimal reconstruction quality under the conventional training pipeline, primarily due to a mismatch between the limited capacity of the SFP and the complexity of the training views. To address this discrepancy, we design a prediction-tailored progressive training strategy to alleviate optimization difficulties and enhance the performance of our framework. In all, our contributions are summarized as follows:

- We propose a prediction-driven 3DGS compression framework (**Predict-GS**) that leverages a Spatial Feature Pool (SFP) for Gaussian prediction and a residual compensation module for refinement, effectively reducing spatial redundancy while maintaining high rendering quality.
- To fully exploit the potential of our predictive compression framework, we design a prediction-tailored progressive training scheme for improving optimization process.
- Extensive experiments demonstrate the effectiveness of our method, achieving a remarkable size reduction of **over 58×** compared to vanilla 3DGS on Mip-NeRF360 Barron et al. (2022). Even compared with the SOTA compression method, we still achieves a **16.91%** bit rate savings, while preserving or improving rendering quality.

2 Related work

2.1 3D Gaussian Splatting

3D Gaussian Splatting (3DGS) Kerbl et al. (2023) represents 3D scenes using a large number of anisotropic Gaussian ellipsoids. With high-fidelity and real-time rendering, 3DGS has rapidly surpassed NeRF-based methods Mildenhall et al. (2021); Li et al. (2023); Müller et al. (2022) and becomes a popular paradigm in the field of Novel View Synthesis (NVS). Subsequent works improve 3DGS from various perspectives. Lu et al. (2024b); Zhang et al. (2024); Liang et al. (2024); Ye et al. (2024); Kheradmand et al. (2025); Rota Bulò et al. (2024) are devoted to improving the rendering quality, while others focus on accelerating training Fang & Wang (2024b); Mallick et al. (2024) and reducing storage costs Chen et al. (2024b); Fan et al. (2024). For Level-of-Detail (LoD) management, hierarchical and spatial partitioning strategies have been introduced to enable scalable rendering Ren et al. (2024); Kerbl et al. (2024). Moreover, this promising representation has been explored in broader domains such as dynamic scene reconstruction Wu et al. (2025; 2024), 3D content generation Tang et al. (2023), human pose estimation Qian et al. (2024); Zheng et al. (2024), and generalizable Gaussian Splatting Liu et al. (2024b); Charatan et al. (2024). However, 3DGS requires storing a large number of primitives and attributes, leading to significant storage and bandwidth overhead. Efficient 3DGS compression is crucial for the further deployment in real-world applications.

2.2 3DGS compression

Existing 3DGS compression methods can be broadly classified into the following categories: **(a) Compact structure.** Scaffold-GS Lu et al. (2024b) introduces anchors to aggregate local Gaussians, leading to a compact structure and inspiring many follow-up works Chen et al. (2024b; 2025); Wang et al. (2024). Alternative compact representations include structuring Gaussian attributes as compressible images Zhang et al. (2025) or planes Lee et al. (2025), and employing hierarchical structures Ren et al. (2024); Wang et al. (2024); Kerbl et al. (2024). **(b) Gaussian pruning.** Lee et al. (2024) learn a binary mask to prune Gaussians with negligible impact on rendering. Mallick et al. (2024); Fang & Wang (2024a) proposes an importance metric for Gaussian densification and pruning. Alternative approaches focus on pruning color attributes Zhang et al. (2023); Fan et al. (2024). **(c) Quantization.** Vector Quantization (VQ) and Residual Vector Quantization (RVQ) with codebooks are used for compressing Gaussian attributes Lee et al. (2024); Navaneet et al. (2024); Zhang et al. (2023); Papantonakis et al. (2024); Fan et al. (2024); Girish et al. (2024). Besides, Chen et al. (2024b; 2025) employ an adaptive quantization strategy, in which each attribute is assigned a learnable quantization step. **(d) Entropy coding.** HAC Chen et al. (2024b) designs an entropy model to model the probability distribution of anchor attributes, enabling efficient entropy coding. Some works enhance entropy model by incorporating additional prior, including hyper-priors Liu et al. (2024a), spatial Wang et al. (2024) and channel-wise auto-regression Chen et al. (2025). Despite the

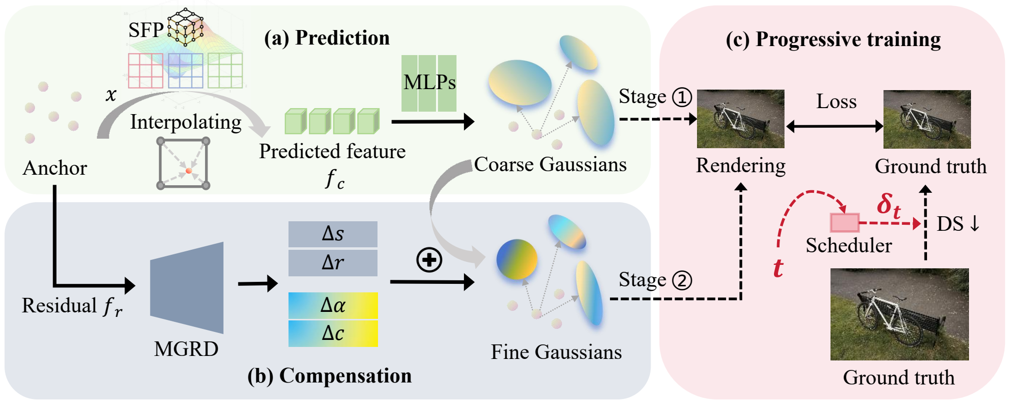


Figure 2: Pipeline of **Predict-GS**. (**a**) Each anchor queries the Spatial Feature Pool (SFP) based on its location to retrieve a predicted feature f_c, which is then used to infer coarse Gaussians following the formulation in Scaffold-GS Lu et al. (2024b). (**b**) To refine the Gaussians, the proposed residual for each anchor is fed into a Multi-head Gaussian Residual Decoder (MGRD), which produces fine-grained corrections to both shape and appearance of predicted Gaussians, obtaining refined, high-fidelity ones. (**c**) The reconstruction process is divided into two stages: the first stage focuses solely on optimizing the Spatial Feature Pool (SFP) for prediction, while the second stage incorporates residual compensation to progressively refine the predicted Gaussians. A cosine-scheduled downsampling (denoted as DS $\downarrow$) is performed to training views, enabling a smooth coarse-to-fine fitting.

progress, a notable limitation persists—existing methods independently encode each primitive in its entirety, overlooking the spatial redundancy inherent among them. Addressing this issue is the core motivation behind our predictive compression framework.

3 Methodology

3.1 Overview

Our overall framework is illustrated in Fig. 2. We first introduce our predictive compression framework in Sec. 3.2, which includes the Spatial Feature Pool (SFP) for prediction and a residual compensation module based on a Multi-head Gaussian Residual Decoder (MGRD). Then, in Sec. 3.3 we present a progressive training strategy tailored-designed for the predictive framework. Finally, Sec. 3.4 details the training loss and the coding pipeline.

3.2 Predictive framework

Preliminary and notation. Anchor-based Gaussian Lu et al. (2024b) adopts anchors as spatial primitives. Each anchor encodes position x, feature f, scale l, and offsets o. A set of associated 3D Gaussians which possesses attributes including position μ, opacity α, color c, scale s and rotation r is derived from each anchor. Specifically, each Gaussian position μ is derived from the anchor's base position x, scale l, and offsets o, while the remaining attributes α, c, s, r are inferred via MLPs that take anchor feature f, relative viewing distance δ_c and viewing direction $\vec{d_c}$ as input. More details are in Appendix A.11.

Prediction. In anchor-based Gaussian Lu et al. (2024b), each anchor is associated with a high-dimensional feature f, which accounts for nearly 60% of the overall model size, as shown in Tab. 3. However, owing to the spatial continuity of the scene, allocating a distinct feature to each anchor leads to significant spatial redundancy. To address the limitation, we propose a Spatial Feature Pool (SFP) for compact storage of shared spatial features, allowing multiple anchors to access a shared feature for Gaussian prediction. Inspired by Instance-NGP Müller et al. (2022) and Compressed NeRF Chen et al. (2024a), our SFP is built on binarized multi-resolution 3D hash grids and 2D hash planes for the following considerations. **(a)** Binarization ensures that the SFP remains lightweight. **(b)** Multi-resolution design allows it to adaptively capture and fuse spatial features at different scales. To balance spatial fidelity and memory efficiency, we leverage low-resolution 3D grids to encode coarse volumetric structures, thereby avoiding the prohibitive memory cost of dense high-resolution 3D volumes. Meanwhile, high-resolution 2D planes are utilized to capture fine-grained, high-frequency details with minimal overhead. **(c)** The use of hash tables not only enables efficient feature querying, but also reduces redundancy among anchors, since anchor coordinates with similar spatial feature may be mapped to the same entry. Specifically, rather than storing a feature f for each anchor in an explicit manner, we query the corresponding spatial feature f_c from SFP using anchor's position x.

$$f_{\text{xyz}} = \mathcal{I}(x, H_{\text{xyz}}), \quad \{f_{\text{xy}}, f_{\text{xz}}, f_{\text{yz}}\} = \{\, \mathcal{I}(x, H_{\text{xy}}), \mathcal{I}(x, H_{\text{xz}}), \mathcal{I}(x, H_{\text{yz}}) \,\}, \tag{1}$$

$$f_c = F(concat(f_{\text{xyz}}, f_{\text{xy}}, f_{\text{xz}}, f_{\text{yz}})), \tag{2}$$

where $\mathcal{I}$ represents the interpolation operation Chen et al. (2024a), H_{xyz} represents the 3D Hash grids and $H_{\text{xy}}, H_{\text{xz}}, H_{\text{yz}}$ denotes the three 2D planes of the decomposition. $F(\cdot)$ is a MLP for multi-scale feature fusion. In replace of the original feature f, the compact shared feature f_c from SFP is used to infer the attributes of k Gaussians via MLP.

$$\{s_i, r_i, \alpha_i, c_i\}_{i=0}^{k-1} = \text{MLP}(f_c, \delta_c, \vec{d_c}). \tag{3}$$

Note that Chen et al. (2024b) employs a similar grid structure as our SFP in its entropy model. In contrast, its grid is used to model the probability of attributes for entropy coding in post-processing, whereas our SFP differs fundamentally by directly predicting features during reconstruction stage.

Residual compensation. However, relying solely on the SFP to predict 3D Gaussians often leads to degraded rendering quality, as the interpolated features f_c are obtained through continuous sampling over spatial coordinates. This interpolation inherently smooths out anchor-specific signals, compromising the ability to capture localized high-frequency details around each anchor. To address this limitation, we introduce a residual feature f_r for each anchor to explicitly compensate for the lost spatial specificity. The residual f_r is decoded by a Multi-head Gaussian Residual Decoder (MGRD) to produce attribute corrections for Gaussian refinement. In details, the shape head MGRD$_s$ models variations for scale and rotation ($\Delta s, \Delta r$) of Gaussians, while the appearance head MGRD$_a$ captures residuals for opacity and color ($\Delta\alpha, \Delta c$). $\delta_c, \vec{d_c}$ provide view-dependent priors.

$$\{\Delta s_i, \Delta r_i\}_{i=0}^{k-1} = \text{MGRD}_s(f_r, \delta_c, \vec{d_c}), \quad \{\Delta\alpha_i, \Delta c_i\}_{i=0}^{k-1} = \text{MGRD}_a(f_r, \delta_c, \vec{d_c}), \tag{4}$$

The predicted Gaussians are refined by incorporating the residual corrections into their respective attributes, resulting in more precise representations. This residual-guided enhancement transforms coarse predicted Gaussians into fine-grained ones, which are subsequently used for splatting and synthesize high-fidelity novel views.

$$\{\hat{s}_i, \hat{r}_i, \hat{\alpha}_i, \hat{c}_i\}_{i=0}^{k-1} = \{s_i, r_i, \alpha_i, c_i\}_{i=0}^{k-1} + \{\Delta s_i, \Delta r_i, \Delta\alpha_i, \Delta c_i\}_{i=0}^{k-1}. \tag{5}$$

where $\{\hat{s}, \hat{r}, \hat{\alpha}, \hat{c}\}$ is the refined attributes of Gaussians.

3.3 Prediction-tailored progressive training

Two-stage training. To better align with our proposed predictive compression framework, we adopt a two-stage training strategy. In the first stage, the residual compensation module is disabled, encouraging the SFP to concentrate on learning precise Gaussian prediction. In the second stage, we activate both the residual attribute and MGRD, enabling the model to learn corrective signals that refine the predictions, and recover high-frequency details in the scene.

Progressive downsampling schedule. As demonstrated in the ablation study (Tab. 4.3), our predictive compression framework is not well supported by the conventional 3DGS training pipeline, which typically optimizes models using high-resolution views throughout the training process. High-resolution views introduce excessive high-frequency details that exceed the representational capacity of our binarized SFP during the initial training stage. As a result, the SFP tends to overfit view-specific details while failing to capture the global structure and underlying geometry.

To address this issue, we propose a progressive training strategy tailored to our predictive compression framework. Specifically, we train the Spatial Feature Pool (SFP) using downsampled views obtained via Bilinear Interpolation $BI(\cdot)$, which offers smoother transitions and better structural preservation compared to pooling-based methods. Downsampling reduces high-frequency signals in the training views and mitigates overfitting to view-specific high-frequency details, encouraging the SFP to focus on the global structure of the scene. To progressively restore the resolution of training views, the downsampling scale is gradually increased during training according to a cosine annealing schedule, which facilitates stable and effective optimization. Ablation studies about the schedule are presented in Appendix A.6. For each training iteration t, the downsampling scale δ_t is derived as follows:

$$\delta_t = \delta_e + 0.5 * (\delta_s - \delta_e) * (1 + cos(\pi * t/T)) \tag{6}$$

where δ_s, δ_e represent the initial and final scale, T denotes total iterations using downsampling. For iteration t, the ground truth $\hat{I}_{gt}$ used for training is downsampled from original image I_{gt}, according to the scale δ_t:

$$\hat{I}_{gt} = BI(I_{gt}, \delta_t). \tag{7}$$

The scale δ_t approaches $\delta_e = 1.0$ at predefined iteration T, after which the model is consistently trained with full-resolution images for fine-tuning. This progressive strategy not only addresses the suboptimal rendering quality but also enhances stability during the transition to high-resolution views for residual learning.

3.4 Training and Coding

Entropy model and training loss. We adopt the Entropy Model EM in HAC Chen et al. (2024b) but modify its output layer adapting to our proposed residual f_r. For each anchor x, the probability distributions and quantization steps of its attributes $a \in \{f_r, l, o\}$ are estimated by EM. Each anchor attribute a, along with its associated probability $p(a)$ and quantization step q_a, is passed to the Arithmetic Encoder (AE) for entropy coding. The detailed derivation of the probability can be found in Appendix A.3. The training objective L remains the Rate-Distortion (R-D) loss as in HAC Chen et al. (2024b), which jointly constrains the rendering quality and the entropy of the attributes. The loss is formulated as:

$$L = L_{scaffold} + \lambda_e L_{entropy} + \lambda_m L_m. \tag{8}$$

$L_{scaffold}$ denotes the loss function definded in Scaffold-GS Lu et al. (2024b) for higher rendering quality, while $L_{entropy}$ and L_m represent the entropy loss and mask loss of HAC Chen et al. (2024b) for lower bit rate. **Note that $L_{entropy}$ includes the entropy loss of our proposed SFP and f_r.** λ_e and λ_m are pre-defined hyper-parameters. More details about the loss can be found in Appendix A.4.

Encoding and decoding. In encoding phase, anchor positions x are used to query Entropy Model EM Chen et al. (2024b), yielding probability distributions and adaptive quantization steps for anchor attributes. These, along with anchor attributes l, o, f_r, are fed into the Arithmetic Encoder (AE) for entropy coding. x are quantized to 16 bits and stored directly. Additionally, Spatial Feature Pool (SFP), Multi-head Gaussian Residual Decoder (MGRD), EM, the mask Chen et al. (2024b), and other MLPs are serialized into bit stream for storage and transmission. In decoding phase, x, SFP, MGRD, EM, mask, and MLPs are first restored. x then query EM to retrieve the corresponding distributions and quantization steps for attributes, which are used by the Arithmetic Decoder (AD) to reconstruct residuals f_r, scales l, and offsets o. Complete encoding and decoding procedures are detailed in Appendix A.7.

Table 1: Quantitative results on Mip-NeRF360 and Tank&Temples.

Dataset	**Mip-NeRF360**				**Tank&Temples**			
Method | Metrics	**PSNR↑**	**SSIM↑**	**LPIPS↓**	**SIZE↓**	**PSNR↑**	**SSIM↑**	**LPIPS↓**	**SIZE↓**
3DGS	27.21	0.815	0.214	734MB	23.14	0.841	0.183	411MB
Scaffold-GS	27.50	0.806	0.252	253.9MB	23.96	0.853	0.177	86.5MB
Compact3DGS	27.03	0.797	0.247	29.1MB	23.32	0.831	0.202	20.9MB
Compressed3D	26.98	0.801	0.238	28.80MB	23.32	0.832	0.194	17.28MB
LightGaussian	27.13	0.806	0.237	45MB	23.44	0.832	0.202	25MB
EAGLES	27.23	0.810	0.240	54MB	23.37	0.840	0.200	29MB
CompGS	27.26	0.800	0.240	16.50MB	23.70	0.840	0.210	9.60MB
Compact3D	27.12	0.806	0.240	19MB	23.44	0.838	0.198	13MB
HAC	27.53	0.807	0.238	15.26MB	24.04	0.846	0.187	8.10MB
Ours	27.53	0.804	0.246	12.68MB	24.18	0.848	0.194	6.62MB
Ours (high rate)	27.78	0.809	0.236	19.28MB	24.20	0.853	0.185	9.32MB

Table 2: Quantitative results on Deep Blending.

Dataset	Deep Blending			
Method | Metrics	PSNR↑	SSIM↑	LPIPS↓	SIZE↓
3DGS	29.41	0.903	0.243	676MB
Scaffold-GS	30.21	0.906	0.254	66MB
Compact3DGS	29.73	0.900	0.258	23.8MB
Compressed3D	29.38	0.898	0.253	25.30MB
LightGaussian	—	—	—	—
EAGLES	29.86	0.910	0.250	52MB
CompGS	29.69	0.900	0.280	8.77MB
Compact3D	—	—	—	—
HAC	29.98	0.902	0.269	4.35MB
Ours	30.17	0.906	0.271	3.75MB
Ours (high rate)	30.49	0.911	0.260	6.35MB

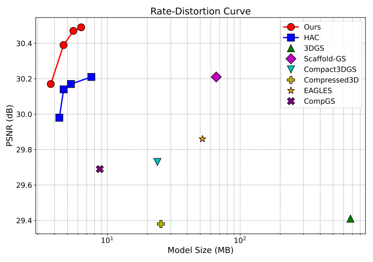


Figure 3: Rate-Distortion (R-D) curve on Deep Blending dataset. A point closer to the top-left corner indicates better compression performance.

Table 3: Bit allocation comparison between HAC and ours on datasets Mip-NeRF360, Tank&Temples and Deep Blending. **Feature** (in MB) denotes the bit rate of anchor features (i.e., f in HAC, and SFP together with f_r in ours). **BSF** indicates the **B**it rate **S**avings of anchor **F**eatures. **Others** (in MB) refers to the bit rate of all remaining components.

Dataset	Mip-NeRF360			Tank&Temples			Deep Blending		
Method | Metrics	Feature	BSF	Others	Feature	BSF	Others	Feature	BSF	Others
HAC	5.92	-0%	9.45	2.95	-0%	5.17	1.47	-0%	3.11
Ours	3.66	**-38.18%**	9.02	1.93	**-34.58%**	4.70	0.78	**-46.94%**	2.97

4 Experiment

4.1 Experimental settings

Implementation details. We build our **Predict-GS** based on the official implementation of HAC Chen et al. (2024b). The SFP is implemented by 3D hash grids with 12 resolution levels and 2D hash planes with 4 resolution levels. Dimension of predicted feature f_c is set as 50 to be consistent with original feature dimension in HAC Chen et al. (2024b). The residual dimension is set to 25, and the corresponding ablation study can be found in Appendix 6. Our MGRD is implemented by a two-layer MLP with ReLU activation. We pretrain the SFP for 5k iterations. The residual branch is activated at iteration 5k and the entropy loss is incorporated at iteration 10k. The total number of iterations is 30k, which is the same as in HAC Chen et al. (2024b). More details can be found in Appendix A.2 and A.4. All experiments are performed on a NVIDIA A100 GPU.

Datasets, metrics and baselines. Evaluation is performed on three public benchmarks: Mip-NeRF360 Barron et al. (2022), Tank&Temples Knapitsch et al. (2017) and Deep Blending Hedman et al. (2018). Quantitative results are reported using PSNR (dB), SSIM, and LPIPS for reconstruction quality, and model size (MB) for compression efficiency. We compare our predictive framework

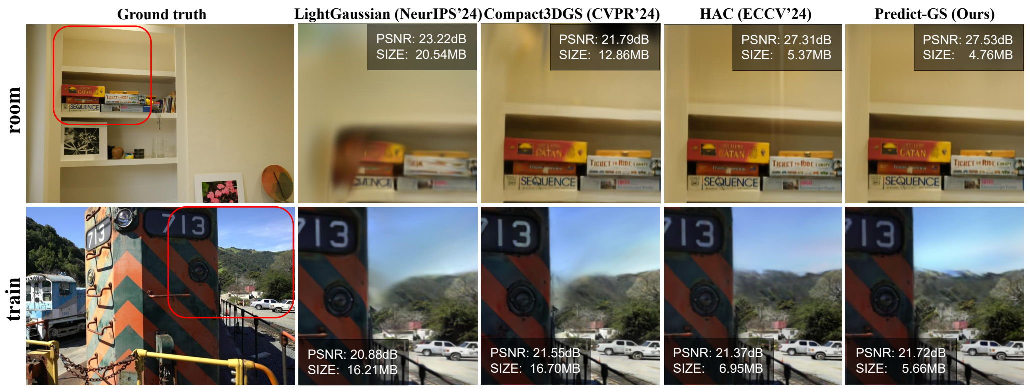

Ground truth
LightGaussian (NeurIPS'24)
Compact3DGS (CVPR'24)
HAC (ECCV'24)
Predict-GS (Ours)
room
PSNR: 23.22dB SIZE: 20.54MB
PSNR: 21.79dB SIZE: 12.86MB
PSNR: 27.31dB SIZE: 5.37MB
PSNR: 27.53dB SIZE: 4.76MB
train
PSNR: 20.88dB SIZE: 16.21MB
PSNR: 21.55dB SIZE: 16.70MB
PSNR: 21.37dB SIZE: 6.95MB
PSNR: 21.72dB SIZE: 5.66MB

Figure 4: Qualitative comparison. In each row, the leftmost image shows the ground truth. The red boxed area is magnified to facilitate more detailed comparison. PSNR and model size are provided.

with 3DGS Kerbl et al. (2023), Scaffold-GS Lu et al. (2024b) and representative 3DGS compression methods (Compact3DGS Lee et al. (2024), Compressed3D Niedermayr et al. (2024), LightGaussian Fan et al. (2024), EAGLES Girish et al. (2024), CompGS Liu et al. (2024c), Compact3D Navaneet et al. (2024), HAC Chen et al. (2024b)), which encompass various compression techniques as discussed in Sec 2.2. It is worth mentioning that HAC Chen et al. (2024b) serves as a strong baseline and is widely acknowledged as the state-of-the-art (SOTA) in 3DGS compression.

4.2 Experimental results

Quantitative and qualitative comparison. A quantitative comparison between our method and representative 3DGS compression approaches is presented in Tab. 1 and Tab. 2, where the top-1, top-2, and top-3 for each metric are highlighted in red, orange, and yellow respectively. In terms of compression performance, our method **(Ours)** achieves significant bit rate savings over existing 3DGS compression methods. Notably, even compared to the SOTA HAC Chen et al. (2024b), our method still yields an average bit rate reduction of 16.91% on Mip-NeRF360 Barron et al. (2022) and 18.27% on Tank&Temples Knapitsch et al. (2017), respectively, which is a significant gain for compression tasks. This gain directly stems from the removal of spatial redundancy among anchor features, validating the effectiveness of our predictive compression framework. For rendering quality, **Ours** outperforms most 3DGS compression baselines and achieves competitive rendering quality with SOTA method HAC Chen et al. (2024b), even with 0.1–0.2 dB improvements in PSNR on Tank&Temples Knapitsch et al. (2017) and Deep Blending Hedman et al. (2018). We attribute these gains to the synergy between our improved progressive training strategy and prediction-driven compression, as presented in Tab. 4. Additionally, a high rate model **Ours (high rate)** is also included to showcase the maximum for reconstruction quality, in a storage-unconstrained setting. We presents a visual comparison of rendering quality between our method and representative compression methods. As shown in Fig. 4, our method achieves comparable rendering quality with HAC Chen et al. (2024b) and outperforms LightGaussian Fan et al. (2024) and Compact3DGS Lee et al. (2024), while maintaining the lowest bit rate among all. More visual examples are provided in Fig. 10 of Appendix A.8.

R-D performance. R-D curve provides a clear visualization of the trade-off between bit rate and reconstruction quality—points closer to the top-left corner indicate better compression performance. We compare our method against several representative compression methods on Deep Blending Hedman et al. (2018) dataset. As shown in Fig. 3, our method achieves superior rate-distortion performance, indicating that under the same bit rate constraints, our model delivers higher reconstruction quality. Conversely, for the same level of rendering quality, our method requires a lower bit rate.

Bit allocation. To further confirm the source of our compression gains and highlight the effectiveness of our method, we report bit rate allocation on each dataset compared with SOTA HAC Chen et al. (2024b). As shown in Tab. 3, the majority of bit rate reduction stems from eliminating spatial redundancy among anchor feature. **The direct gain accounts for up to 40% on average relative to the bit rate of anchor feature.** (We reproduce HAC Chen et al. (2024b) using its official code to obtain the results in Tab. 3.)

4.3 Ablation studies

Effectiveness of each component. Tab. 4 presents quantitative results of our component-wise ablation study. **Baseline** is the SOTA HAC Chen et al. (2024b). **Ours w/ SFP** shows the results using only SFP prediction, **Ours w/ SFP w/** f_r includes the additional residual branch, and **Ours all** further incorporates our progressive training strategy, representing the full method. Compared to **Baseline**, **Ours w/ SFP** achieves a significant reduction in model size because only a lightweight SFP needs to be stored, instead of the complete feature for each anchor. Notably, **Ours w/ SFP** achieves a PSNR of 22.61dB on Tank&Temples Knapitsch et al. (2017) dataset, indicating that SFP possesses the capability to capture spatial features for 3D scene reconstruction and shows strong potential for Gaussian prediction. **Ours w/ SFP w/** f_r highlights the effectiveness of residual compensation for fidelity in local details. While albeit with additional bit rate overhead, **Ours w/ SFP w/** f_r still achieves a lower overall bit rate compared to **Baseline**. **Ours all** further enhances the rendering quality compared to **Ours w/ SFP w/** f_r (0.17dB $\uparrow$ on Tank&Temples Knapitsch et al.

Table 4: Ablation study on dataset Tank&Temples Knapitsch et al. (2017). **w/ SFP** denotes prediction with SFP. **w/ f_r** and **w/ PT** represent the residual compensation and progressive training respectively.

Method	w/ SFP	w/ f_r	w/ PT	PSNR↑	SSIM↑	LPIPS↓	SIZE↓
Baseline	—	—	—	24.04	0.846	0.187	8.10MB
Ours w/ SFP	✓			22.61	0.802	0.261	4.75MB
Ours w/ SFP w/ f_r	✓	✓		24.01	0.841	0.200	6.75MB
Ours all	✓	✓	✓	24.18	0.848	0.194	6.62MB

(2017)), demonstrating the effectiveness of our progressive training in maximizing the potential of our predictive framework.

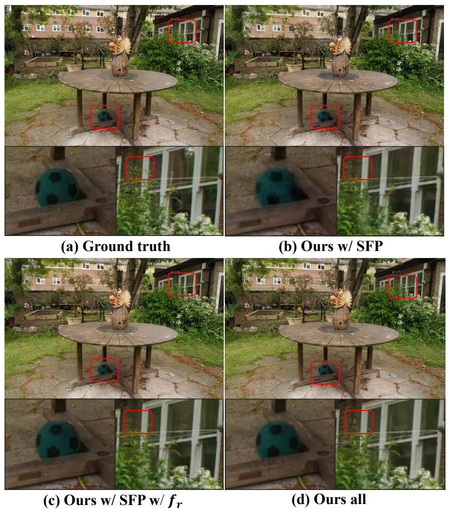


Figure 5: Visualization for ablation study.

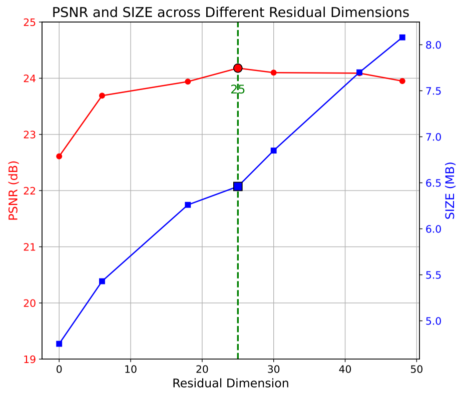


Figure 6: Ablation study on residual dimension. The experiment is conducted on the Tank&Temples Knapitsch et al. (2017) dataset. We report results of average PSNR and model size across the entire dataset under different residual dimensions.

Visualization. We provide visualization in Fig. 5 on the *garden* scene from Mip-NeRF360 Barron et al. (2022). Regions with noticeable differences are highlighted. As shown in Fig. 5, **Ours w/ SFP** indicates that predicted Gaussians can roughly fit the 3D scene context but fail to capture fine details. **Ours w/ SFP w/ f_r** demonstrate that residual compensation can effectively correct the inaccurate predictions (e.g., blurriness and distortions near the football and plants). **Ours all** further shows that the progressive training strategy leads to additional improvements in rendering quality.

The dimension of residual. We set residual dimension as 25 for default. Fig. 6 provides ablation study on Tank&Temples Knapitsch et al. (2017) to validate the effectiveness of our choice. When the dimension exceeds 25, the bit rate increases significantly while rendering quality remains unchanged or even decreased (too much channel redundancy may hinder optimization), indicating that 25-dimension is sufficient to compensate for the bias of the predicted Gaussians. In contrast, using dimension lower than 25 may reduce bit rate, but leads to a notable drop in quality. These results suggest that a 25-dimensional residual strikes a good balance, providing sufficient capacity to recover missing details without introducing excessive channel redundancy.

5 Conclusion

In this work, we introduce prediction and residual compensation into 3DGS compression, and propose a prediction-driven compression framework equipped with a tailored progressive training strategy, effectively mitigating spatial redundancy across anchors. Extensive experiments on public benchmarks confirm the effectiveness of our framework and individual contribution of each component.

REFERENCES

Milena T Bagdasarian, Paul Knoll, Yi-Hsin Li, Florian Barthel, Anna Hilsmann, Peter Eisert, and Wieland Morgenstern. 3dgs. zip: A survey on 3d gaussian splatting compression methods. *arXiv preprint arXiv:2407.09510*, 2024.

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*, pp. 5470–5479, 2022.

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 19457–19467, 2024.

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*, pp. 20321–20330, 2024a.

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*, pp. 422–438. Springer, 2024b.

Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Hac++: Towards 100x compression of 3d gaussian splatting. *arXiv preprint arXiv:2501.12255*, 2025.

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, Zhangyang Wang, et al. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. *Advances in neural information processing systems*, 37:140138–140158, 2024.

Guangchi Fang and Bing Wang. Mini-splatting: Representing scenes with a constrained number of gaussians. In *European Conference on Computer Vision*, pp. 165–181. Springer, 2024a.

Guangchi Fang and Bing Wang. Mini-splatting2: Building 360 scenes within minutes via aggressive gaussian densification. *arXiv preprint arXiv:2411.12788*, 2024b.

Linus Franke, Laura Fink, and Marc Stamminger. Vr-splatting: Foveated radiance field rendering via 3d gaussian splatting and neural points. *arXiv preprint arXiv:2410.17932*, 2024.

Sharath Girish, Kamal Gupta, and Abhinav Shrivastava. Eagles: Efficient accelerated 3d gaussians with lightweight encodings. In *European Conference on Computer Vision*, pp. 54–71. Springer, 2024.

Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5354–5363, 2024.

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.

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 conference papers*, pp. 1–11, 2024.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A hierarchical 3d gaussian representation for real-time rendering of very large datasets. *ACM Transactions on Graphics (TOG)*, 43(4):1–15, 2024.

Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3d gaussian splatting as markov chain monte carlo. *Advances in Neural Information Processing Systems*, 37:80965–80986, 2025.

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.

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*, pp. 21719–21728, 2024.

Soonbin Lee, Fangwen Shu, Yago Sanchez, Thomas Schierl, and Cornelius Hellge. Compression of 3d gaussian splatting with optimized feature planes and standard video codecs. *arXiv preprint arXiv:2501.03399*, 2025.

Jiahao Li, Bin Li, and Yan Lu. Neural video compression with diverse contexts. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 22616–22626, 2023.

Zhihao Liang, Qi Zhang, Wenbo Hu, Lei Zhu, Ying Feng, and Kui Jia. Analytic-splatting: Anti-aliased 3d gaussian splatting via analytic integration. In *European conference on computer vision*, pp. 281–297. Springer, 2024.

Lei Liu, Zhenghao Chen, and Dong Xu. Hemgs: A hybrid entropy model for 3d gaussian splatting data compression. *arXiv preprint arXiv:2411.18473*, 2024a.

Tianqi Liu, Guangcong Wang, Shoukang Hu, Liao Shen, Xinyi Ye, Yuhang Zang, Zhiguo Cao, Wei Li, and Ziwei Liu. Mvsgaussian: Fast generalizable gaussian splatting reconstruction from multi-view stereo. In *European Conference on Computer Vision*, pp. 37–53. Springer, 2024b.

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*, pp. 2936–2944, 2024c.

Tao Lu, Ankit Dhiman, R Srinath, Emre Arslan, Angela Xing, Yuanbo Xiangli, R Venkatesh Babu, and Srinath Sridhar. Turbo-gs: Accelerating 3d gaussian fitting for high-quality radiance fields. *arXiv preprint arXiv:2412.13547*, 2024a.

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*, pp. 20654–20664, 2024b.

Jingui Ma, Yang Hu, Luyang Tang, Jiayu Yang, Yongqi Zhai, and Ronggang Wang. Enhancing 3d gaussian splatting compression via spatial condition-based prediction, 2025. URL `https://arxiv.org/abs/2503.23337`.

Siwei Ma, Xinfeng Zhang, Chuanmin Jia, Zhenghui Zhao, Shiqi Wang, and Shanshe Wang. Image and video compression with neural networks: A review. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(6):1683–1698, 2019.

Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Markus Steinberger, Francisco Vicente Carrasco, and Fernando De La Torre. Taming 3dgs: High-quality radiance fields with limited resources. In *SIGGRAPH Asia 2024 Conference Papers*, pp. 1–11, 2024.

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.

David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. *Advances in neural information processing systems*, 31, 2018.

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.

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*, pp. 330–349. Springer, 2024.

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*, pp. 10349–10358, 2024.

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.

Zhiyin Qian, Shaofei Wang, Marko Mihajlovic, Andreas Geiger, and Siyu Tang. 3dgs-avatar: Animatable avatars via deformable 3d gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5020–5030, 2024.

Kerui Ren, Lihan Jiang, Tao Lu, Mulin Yu, Linning Xu, Zhangkai Ni, and Bo Dai. Octree-gs: Towards consistent real-time rendering with lod-structured 3d gaussians. *arXiv preprint arXiv:2403.17898*, 2024.

Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. Revising densification in gaussian splatting. In *European Conference on Computer Vision*, pp. 347–362. Springer, 2024.

Maximilian Speicher, Brian D Hall, and Michael Nebeling. What is mixed reality? In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pp. 1–15, 2019.

Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023.

Fabio Tosi, Alessio Tonioni, Daniele De Gregorio, and Matteo Poggi. Nerf-supervised deep stereo. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 855–866, 2023.

Yufei Wang, Zhihao Li, Lanqing Guo, Wenhan Yang, Alex Kot, and Bihan Wen. Contextgs: Compact 3d gaussian splatting with anchor level context model. *Advances in neural information processing systems*, 37: 51532–51551, 2024.

Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 20310–20320, 2024.

Jiahao Wu, Rui Peng, Zhiyan Wang, Lu Xiao, Luyang Tang, Jinbo Yan, Kaiqiang Xiong, and Ronggang Wang. Swift4d: Adaptive divide-and-conquer gaussian splatting for compact and efficient reconstruction of dynamic scene. *arXiv preprint arXiv:2503.12307*, 2025.

Jinbo Yan, Rui Peng, Luyang Tang, and Ronggang Wang. 4d gaussian splatting with scale-aware residual field and adaptive optimization for real-time rendering of temporally complex dynamic scenes. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 7871–7880, 2024.

Ziyi Yang, Xinyu Gao, Yang-Tian Sun, Yihua Huang, Xiaoyang Lyu, Wen Zhou, Shaohui Jiao, Xiaojuan Qi, and Xiaogang Jin. Spec-gaussian: Anisotropic view-dependent appearance for 3d gaussian splatting. *Advances in Neural Information Processing Systems*, 37:61192–61216, 2024a.

Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 20331–20341, 2024b.

Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. Absgs: Recovering fine details in 3d gaussian splatting. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 1053–1061, 2024.

Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 19447–19456, 2024a.

Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian opacity fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics (TOG)*, 43(6):1–13, 2024b.

Hongjia Zhai, Xiyu Zhang, Boming Zhao, Hai Li, Yijia He, Zhaopeng Cui, Hujun Bao, and Guofeng Zhang. Splatloc: 3d gaussian splatting-based visual localization for augmented reality. *IEEE Transactions on Visualization and Computer Graphics*, 2025.

Jiahui Zhang, Fangneng Zhan, Ling Shao, and Shijian Lu. Sogs: Second-order anchor for advanced 3d gaussian splatting. *arXiv preprint arXiv:2503.07476*, 2025.

Zhaobin Zhang, Semih Esenlik, Yaojun Wu, Meng Wang, Kai Zhang, and Li Zhang. End-to-end learning-based image compression with a decoupled framework. *IEEE Transactions on Circuits and Systems for Video Technology*, 34(5):3067–3081, 2023.

Zheng Zhang, Wenbo Hu, Yixing Lao, Tong He, and Hengshuang Zhao. Pixel-gs: Density control with pixel-aware gradient for 3d gaussian splatting. In *European Conference on Computer Vision*, pp. 326–342. Springer, 2024.

Shunyuan Zheng, Boyao Zhou, Ruizhi Shao, Boning Liu, Shengping Zhang, Liqiang Nie, and Yebin Liu. Gps-gaussian: Generalizable pixel-wise 3d gaussian splatting for real-time human novel view synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 19680–19690, 2024.

A Appendix

A.1 Overview

This appendix provides supplementary materials and more details to support the main paper. The contents are organized as follows:

A.2 More implementation details

We build our predictive compression framework based on the official implementation of HAC Chen et al. (2024b). In our implementation, we adhere to the same data processing protocols, model optimization strategies, and hyperparameter settings as those employed in HAC Chen et al. (2024b) for a fair comparison.

Prediction and residual compensation. We construct our Spatial Feature Pool (SFP) using binarized multi-resolution 3D hash grids and 2D planes. The resolution list of 3D grids is (18, 24, 33, 44, 59, 80, 108, 148, 201, 275, 376, 514), while resolution list of 2D planes is (130, 258, 514, 1026). Each vertex in the grids or the planes is individually mapped to an entry in hash table. The hash table for the grids has a size of 2^{13}, while the hash table for the planes have a size of 2^{15}. Each entry in the hash tables have a 4-dimension feature. The fusion network following the SFP is a two-layer MLP, with the output dimension set to 50, which is consistent with the original anchor feature dimension used in HAC Chen et al. (2024b). The residual dimension for each anchor is set to 25, which is empirically determined to be optimal through ablation studies. Larger dimensions lead to a rapid increase for bit rate, while smaller ones result in significant quality degradation. The Multi-head Gaussian Residual Decoder (MGRD) is implemented as a two-layer MLP with ReLU activations. The output dimension of the shape head is 7, matches that of the Gaussian attributes scale (3) and rotation (4). The output dimension of the appearance head is 4, matches that of the Gaussian attributes opacity (1) and color (3).

Prediction-tailored progressive training. We adopt a customized progressive training strategy, starting from iteration 0 and continuing for T =9k iterations. The initial downsampling scale δ_s is set to 0.7 and gradually increases to δ_e 1.0 following a cosine schedule. During the first 5k iterations of training, we disable the residual branch and exclusively train the SFP. After 5k iterations, we enable the residual compensation.

A.3 Entropy model

We adopt the Entropy Model EM as in HAC Chen et al. (2024b) but modify its output layer adapting to our proposed residual f_r. For each anchor x, the probability distribution of its attributes a is modeled as a Gaussian distribution $a \sim N(\mu_a, \theta_a)$, with both the mean μ_a and variance θ_a estimated by EM.

$$\{\mu_a, \theta_a, q_a\} = EM(x), \quad a \in \{f_r, o, l\}, \tag{9}$$

where q_a is the corresponding quantization step for a. For each anchor attribute a, its probability $p(a)$ is computed using the probability density function of a Gaussian distribution $\phi_{\mu_a,\theta_a}(\cdot)$.

$$p(a) = \int_{a-\frac{1}{2}q_a}^{a+\frac{1}{2}q_a} \phi_{\mu_a,\theta_a}(x)dx = \Phi_{\mu_a,\theta_a}(a + \frac{1}{2}q_a) - \Phi_{\mu_a,\theta_a}(a - \frac{1}{2}q_a). \tag{10}$$

where $\Phi(\cdot)$ denotes the Cumulative Distribution Function (CDF) of $\phi(\cdot)$. Each anchor attribute a, along with its associated probability $p(a)$ and quantization step q_a, is passed to the Arithmetic Encoder (AE) for entropy coding.

A.4 Training loss and training details

The training objective L in HAC Chen et al. (2024b) consists of three parts, which jointly constrains the rendering quality and the entropy of the attributes. The loss is formulated as:

$$L = L_{scaffold} + \lambda_e L_{entropy} + \lambda_m L_m. \tag{11}$$

$L_{scaffold}$ denotes reconstruction loss definded in Scaffold-GS Lu et al. (2024b), including L1 loss and SSIM loss between ground truth I_{gt} and rendered image I_{render}, as well as a regularization term to encourage smaller scale of Gaussians.

$$L_{scaffold} = L_1(I_{gt}, I_{render}) + \lambda_{SSIM} L_{SSIM}(I_{gt}, I_{render}) + \lambda_{reg} L_{reg}, \tag{12}$$

$L_{entropy}$ and L_m represent the entropy loss and mask loss definded in HAC Chen et al. (2024b). λ_e is set to 0.004 and λ_m is set to 5e-4 for default in our evaluation. Notably, λ_e is set to 0.0005 to attain a high rate model **Ours (high rate)**.

$$L_{entropy} = \sum_{a \in \{f_r, l, o\}} \sum_{i=1}^{N} \sum_{j=1}^{D} \left(-\log_2 p\left(a_{i,j}\right)\right) + L_{hash}, \tag{13}$$

$$L_{hash} = M_+ \times \left(-\log_2\left(h_f\right)\right) + M_- \times \left(-\log_2\left(1 - h_f\right)\right). \tag{14}$$

where N is the number of anchors and $a_{i,j}$ is a_i's j-th dimension value. Minimizing the entropy loss encourages a high probability estimation for $p(a_{i,j})$, guiding the accurate estimation for $\mu_{i,j}, \theta_{i,j}$ of the Entropy Model EM. L_{hash} quantifies the entropy of the binarized embeddings in the hash tables employed by our proposed SFP and EM of Chen et al. (2024b), where M_+ and M_- are total numbers of $+1$ and -1, and h_f is the occurrence frequency of the symbol $+1$. The concept of mask loss L_m was originally proposed in Compact3DGS Lee et al. (2024) to promote the pruning of redundant Gaussians and anchors during training.

The total number of training iterations is 30k. During the first 5k iterations, we only train the SFP module to fit the basic structure of the scene. After 5k iterations, the residual compensation branch is introduced. Additionally, during the first 10k iterations, the entropy loss constraint $L_{entropy}$ is disabled to allow the model to focus on scene reconstruction.

A.5 Efficiency analysis

Training time and FPS. Tab. 5 further quantify the impact on efficiency of our method. Our rendering speed experiences a slight degradation compared with HAC, due to an additional residual branch during inference. Nevertheless, our system still supports real-time rendering for 70+ FPS. Despite the residual branch in forward pass during training, the overall training time remains efficient (even 4% faster than HAC), which stems from two factors: **(a)** The entropy model becomes more lightweight, as it only estimates the probability of residual rather than the complete feature. **(b)** Our progressive training strategy avoids using high-resolution view in early training. By the way, the FPS degradation could be addressed if refined Gaussians are cached at rendering.

Encoding and decoding time. We also report the impact of our method on encoding and decoding time on Mip-NeRF360 dataset. As shown in Tab. 5, our method achieves an average encoding time of 6.73s and decoding time of 18.56s on the Mip-NeRF360 dataset, slightly faster than HAC. It is because our method only needs to encode a short residual, rather than lengthy feature.

Table 5: Efficiency comparison on Mip-NeRF360 (testing on a NVIDIA A100 GPU).

Method	Size (MB)↓	Train time (s)↓	Encode time (s)↓	Decode time (s)↓	Test FPS↑
HAC	15.23	2180	6.96	18.72	87.88
Ours	12.65	2097	6.73	18.56	71.74

Table 6: Results of different annealing schedules. **Linear** denotes that the downsampling scale is annealing linearly during the progressive training process, while **Ours (Cosine)** uses a cosine-based annealing schedule. **Non** refers to the case without progressive annealing, meaning a fixed initial downsampling scale is applied throughout SFP training and no further downsampling is applied once residual training begins.

Dataset		**Mip-NeRF360 Barron et al. (2022)**			
Method	**Metrics**	**PSNR↑**	**SSIM↑**	**LPIPS↓**	**SIZE↓**
Non		26.71	0.779	0.258	12.81MB
Linear		27.43	0.802	0.248	12.71MB
Ours (Cosine)		27.53	0.804	0.246	12.68MB

A.6 Ablation studies on training scheduling

The annealing schedule. To fully harness the potential of our prediction-driven 3DGS compression framework, we adopt a progressive training strategy that applies cosine annealing downsampling for the training views. Tab. 6 shows the impact of various annealing schedules on the final performance of models, demonstrating the effectiveness of the cosine annealing approach we employ. As shown in the first row (**Non**), the absence of a progressive transition leads to a significant drop in quality. This is mainly due to the abrupt increase in the downsampling scale when the residual branch is introduced, which causes instability in the optimization process. The results in the second (**Linear**) and third rows (**Cosine**) demonstrate that cosine annealing outperforms linear annealing, as the smaller slope of the schedule function in the early stages facilitates a more stable training process, allowing the SFP module to effectively capture the geometry context of the scene.

Initial downsampling scale. Furthermore, Tab. 7 reports the results under different initial downsampling scales, further validating the optimality of our chosen hyper-parameters. We observe that an initial downsampling scale in the range of 0.6 to 0.7 yields satisfactory results, while increasing the scale to 0.8 and over leads to a noticeable degradation in rendering quality and bit rate increasing. Considering that an excessively small initial downsampling scale would severely impair image information, while an overly large scale would diminish the benefits of progressive training, we set the initial downsampling scale to 0.7 based on ablation studies. It is worth mentioning that the hyperparameter choice generalizes well across multiple datasets.

A.7 Encoding and decoding pipeline and visualization

A visualization of the compressed bit stream is provided in the Fig. 8, where the encoding and decoding processes are annotated with step-by-step indices.

Fig.9 presents a detailed comparison of anchor feature encoding strategies between our method and other anchor-based Gaussian compression techniques.

A.8 Visually comparison with representative 3DGS compression works

As presented in Fig 10, our method achieves superior rendering quality and lower bit rate compared to representative 3DGS compression works, LightGaussian Fan et al. (2024) and Compact3DGS Lee et al. (2024). Even when compared to the SOTA compression method HAC Chen et al. (2024b), our approach achieves a notable bit rate reduction while maintaining comparable rendering quality.

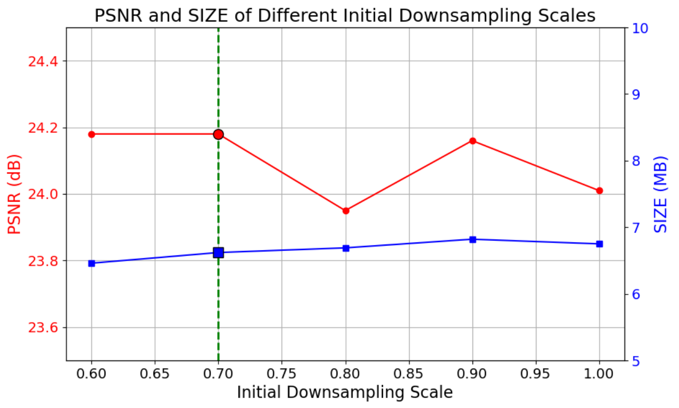


Figure 7: Ablation study on initial scale of downsampling during training. The experiment is conducted on the Tank&Temples Knapitsch et al. (2017) dataset. We report results of average PSNR and model size across the entire dataset under different initial scales of downsampling.

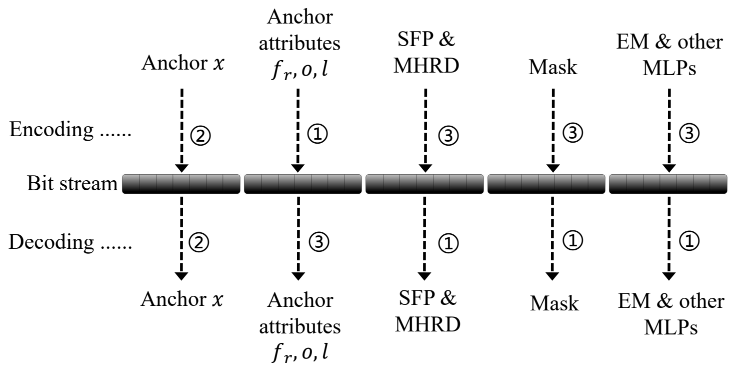


Figure 8: A visualization of the compressed bit stream and the order of encoding and decoding.

A.9 Results for each scene

Tab. 7, Tab. 8 and Tab. 9 present the per-scene results of our method corresponding to the comparative experiments in our paper (Tab. 1 and Tab. 2).

Table 7: Per-scene results on the Mip-NeRF360 Barron et al. (2022) dataset.

Scene	PSNR↑	SSIM↑	LPIPS↓	SIZE↓
bicycle	25.05	0.738	0.274	21.61
garden	27.18	0.837	0.161	17.23
stump	26.78	0.762	0.274	15.83
room	31.75	0.920	0.221	4.76
counter	29.30	0.910	0.201	6.35
kitchen	31.15	0.922	0.135	6.81
bonsai	32.32	0.943	0.193	7.90
flower	21.07	0.562	0.392	16.38
treehill	23.14	0.641	0.362	17.25
Avg	**27.53**	**0.804**	**0.246**	**12.68**

A.10 Limitation and discussion

Limitation. While our predictive compression framework reduces spatial redundancy among anchors and improves compression performance, it also increases the computational cost during training and rendering, mainly due to the additional learning of residual attributes and the residual

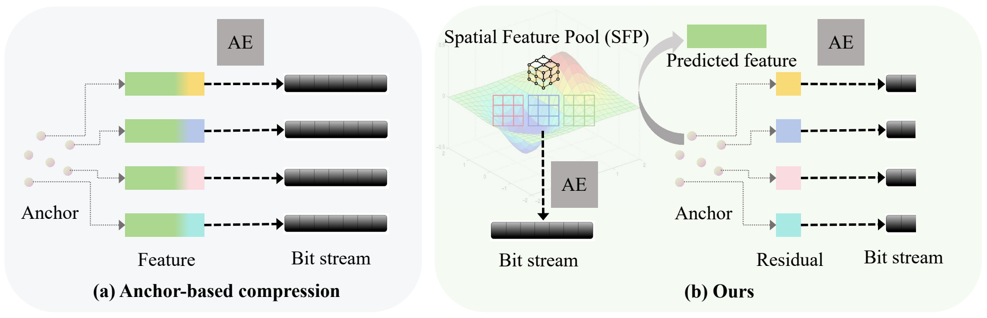


Figure 9: How to encode anchor feature. Existing anchor-based Gaussian compression encodes the complete feature for each anchor, leading to substantial bit rate overhead. Due to high similarity among neighboring anchors, a large portion of the encoded feature is redundant, as shared components are repeatedly stored (**left**). To address this, we propose a compact Spatial Feature Pool (SFP) that captures the shared feature components across anchors. Each anchor then only encodes its residual with respect to the SFP, effectively eliminating spatial redundancy and significantly reducing the bit rate (**right**).

Table 8: Per-scene results on the Tank&Temples Knapitsch et al. (2017) dataset.

Scene	**PSNR↑**	**SSIM↑**	**LPIPS↓**	**SIZE↓**
train	22.47	0.816	0.223	5.66
truck	25.88	0.880	0.165	7.58
Avg	**24.18**	**0.848**	**0.194**	**6.62**

Table 9: Per-scene results on the Deep Blending Hedman et al. (2018) dataset.

Scene	**PSNR↑**	**SSIM↑**	**LPIPS↓**	**SIZE↓**
playroom	29.65	0.904	0.269	4.44
drjohnson	30.69	0.908	0.273	3.06
Avg	**30.17**	**0.906**	**0.271**	**3.75**

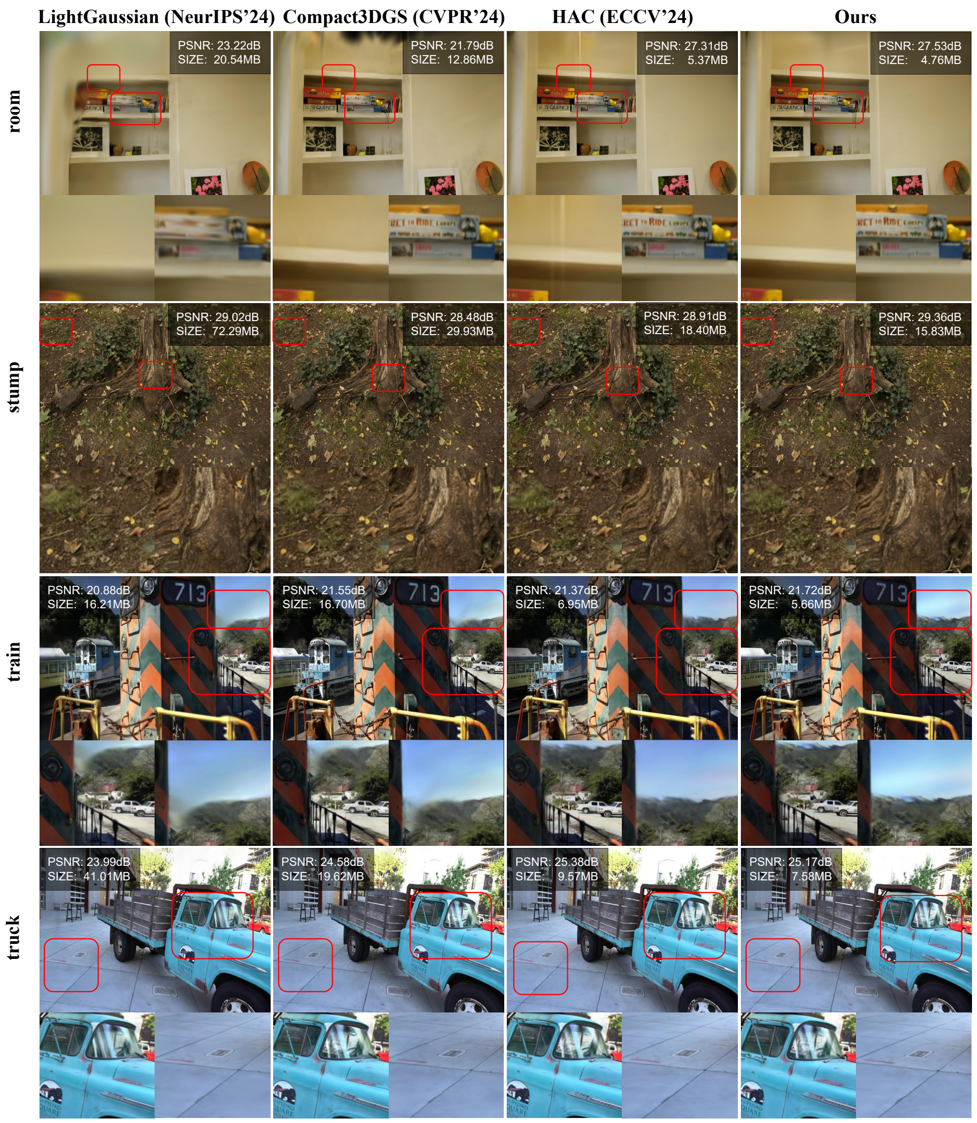

LightGaussian (NeurIPS'24)
Compact3DGS (CVPR'24)
HAC (ECCV'24)
Ours
room
stump
train
truck
PSNR: 23.22dB SIZE: 20.54MB
PSNR: 21.79dB SIZE: 12.86MB
PSNR: 27.31dB SIZE: 5.37MB
PSNR: 27.53dB SIZE: 4.76MB
PSNR: 29.02dB SIZE: 72.29MB
PSNR: 28.48dB SIZE: 29.93MB
PSNR: 28.91dB SIZE: 18.40MB
PSNR: 29.36dB SIZE: 15.83MB
PSNR: 20.88dB SIZE: 16.21MB
PSNR: 21.55dB SIZE: 16.70MB
PSNR: 21.37dB SIZE: 6.95MB
PSNR: 21.72dB SIZE: 5.66MB
PSNR: 23.99dB SIZE: 41.01MB
PSNR: 24.58dB SIZE: 19.62MB
PSNR: 25.38dB SIZE: 9.57MB
PSNR: 25.17dB SIZE: 7.58MB

Figure 10: Visually comparison with representative 3DGS compression works. The regions with noticeable differences have been highlighted with red boxes and zoomed-in views for emphasis. We reproduce the work of LightGaussian Fan et al. (2024), Compact3DGS Lee et al. (2024) and HAC Chen et al. (2024b) to get the visual examples for qualitative visualization.

decoder (MGRD). In future work, we will explore acceleration strategies to address this issue. Further more, our method primarily targets spatial redundancy in feature attributes, while other geometric components (e.g., position, scale, offset) still contribute significantly to the bit stream. This calls for future efforts toward dedicated compression strategies for geometric attributes.

Discussion. We note that concurrent works HAC++ Chen et al. (2025) and HEMGS Liu et al. (2024a), improve upon HAC Chen et al. (2024b) by incorporating richer context into the entropy model, thereby achieving better compression performance. However, in this work, we still adopt HAC (Chen et al. (2024b) as our baseline and implementation framework, primarily due to its wide recognition and established effectiveness. It is worth highlighting that our contribution lies in introducing predictive techniques at the reconstruction stage to eliminate spatial redundancy — a pre-compression design. We do not change the entropy model structure of HAC (Chen et al. (2024b), which is used for post-compression process. Therefore, our method is fully compatible with the aforementioned works (Chen et al. (2025); Liu et al. (2024a), meaning that their improvements to the entropy model can be directly integrated into our predictive compression framework to further enhance overall compression performance. In addition, our work introduces the predictive coding paradigm into 3DGS compression, aligning its overall framework with that of image compression (Ma et al. (2019), and thereby enabling the adaptation of prediction-based advances from image compression to 3DGS context.

A.11 Brief overview of 3DGS and anchor-based variant

3DGS. 3DGS Kerbl et al. (2023) represents a 3D scene by an extensive number of anisotropic ellipsoids equipped with two geometry attributes (location μ, covariance matrix Σ) and two appearance attributes (view-dependent Spherical Harmonic (SH) coefficients h and opacity α), which can be defined as follows:

$$\mathcal{G}(\mathbf{x};\mu,\Sigma) = \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right), \tag{15}$$

where $\mathbf{x}$ is the coordinates of a 3D scene point and the covariance matrix Σ encodes the scale s and rotation r through $\Sigma = rss^Tr^T$. In order to obtain images from a novel perspective, these Gaussians will be splatted to the 2D space and apply α-blending to render the pixel value C:

$$C = \sum_{i\in N} c_i\alpha_i \prod_{j=1}^{i-1}(1-\alpha_j), \tag{16}$$

where N is the number of sorted Gaussians contributing to the rendering, and c is the color calculated by SH coefficients h.

Anchor-based variant. Scaffold-GS Lu et al. (2024b) adopts compact anchors as spatial primitives and utilizes MLPs to infer the attributes of surrounding 3D Gaussians. Each anchor encodes properties including position x, feature f, scale l, and offsets o, from which the associated Gaussians μ are implicitly derived:

$$\{\mu_i\}_{i=0}^{k-1} = x + \{o_i\}_{i=0}^{k-1} * l, \tag{17}$$

The attributes of the k neural Gaussians are inferred via dedicated MLPs that take as input the anchor feature f, the relative viewing distance δ_c, and the viewing direction $\vec{d_c}$. For example, the opacity α of neural Gaussians spawned from an anchor are decoded as follows:

$$\{\alpha_i\}_{i=0}^{k-1} = \mathrm{MLP}_\alpha(f,\delta_c,\vec{d_c}). \tag{18}$$