PCGS: Progressive Compression of 3D Gaussian Splatting

Yihang Chen[1, 2*], **Mengyao Li**[3*], **Qianyi Wu**[2], **Weiyao Lin**[1†], **Mehrtash Harandi**[2], **Jianfei Cai**[2]

[1]Shanghai Jiao Tong University, China
[2]Monash University, Australia
[3]Donghua University, China
{yhchen.ee, wylin}@sjtu.edu.cn, mylee0608@163.com, {qianyi.wu, mehrtash.harandi, jianfei.cai}@monash.edu

Abstract

3D Gaussian Splatting (3DGS) achieves impressive rendering fidelity and speed for novel view synthesis. However, its substantial data size poses a significant challenge for practical applications. While many compression techniques have been proposed, they fail to efficiently utilize existing bitstreams in on-demand applications due to their lack of progressivity, leading to a waste of resource. To address this issue, we propose **PCGS** (**P**rogressive **C**ompression of 3D **G**aussian **S**platting), which adaptively controls **both the quantity and quality** of Gaussians (or anchors) to enable effective progressivity for on-demand applications. For quantity, we introduce a progressive masking strategy that incrementally incorporates new anchors while refining existing ones to enhance fidelity. For quality, we propose a progressive quantization approach that gradually reduces quantization step sizes to achieve finer modeling of Gaussian attributes. Furthermore, to compact the incremental bitstreams, we leverage existing quantization results to refine probability prediction, improving entropy coding efficiency across progressive levels. PCGS achieves progressivity while maintaining compression performance comparable to SoTA non-progressive methods.

Introduction

In the field of novel view synthesis, 3D Gaussian Splatting (3DGS) (Kerbl et al. 2023) has emerged as a leading technology due to its photo-realistic rendering fidelity and real-time rendering speed. By explicitly defining Gaussians with color and geometric attributes, 3DGS provides a flexible and efficient 3D scene representation. However, the large number of Gaussians results in excessively large data sizes, making transmission and storage challenging (Xiangli et al. 2022; Ali et al. 2024; Bagdasarian et al. 2024). To address this, various approaches have been proposed to compress 3DGS (Bagdasarian et al. 2024; Fan et al. 2024; Girish, Gupta, and Shrivastava 2024; Ali et al. 2024; Yang et al. 2024; Zhang et al. 2024b; Morgenstern et al. 2023). Despite these advancements, existing methods exhibit two key limitations: **(1) Single-rate compression**: Each bitstream from these compression models is only for a fixed rate/data size; and

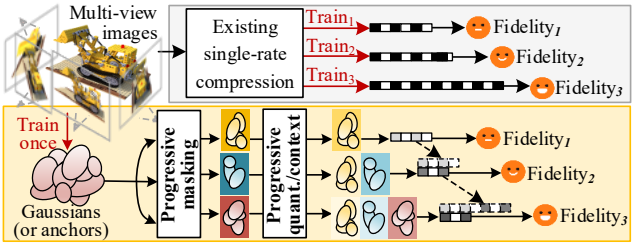


Figure 1: **Comparison of existing approaches (upper) and the proposed progressive compression (lower).** Existing approaches generate multiple independent bitstreams for different rates and fidelity through multiple trainings, while a progressive compression approach (with only one training) can continuously improve the fidelity by adding bitstreams.

(2) Lack of progressivity: Although these models can be retrained to produce bitstreams for different rate targets, the bitstreams are non-reusable, as in the upper part of Fig. 1.

These limitations become more pronounced in on-demand applications. For instance, as transmission bandwidth increases, bitstreams with higher rate budgets can be transmitted for better rendering fidelity. However, due to their single-rate and non-progressive nature of existing 3DGS compression methods, generating a new bitstream requires retraining the model, where existing bitstreams cannot be reused, leading to inefficient resource utilization. These inefficiencies are further exacerbated in large-scale 3DGS scenes (Barron et al. 2022), which often exceed 20 megabytes (MB) even after compression. To overcome these challenges, progressive 3DGS compression presents a promising direction. The key idea is that *existing bitstreams that encode base-quality 3DGS can be refined and augmented with additional bitstreams to achieve higher fidelity*, as illustrated in the lower part of Fig. 1.

Among existing 3DGS compression approaches, some studies have explored progressive 3DGS representations (Shi et al. 2024; Di Sario et al. 2025; Cheng et al. 2024; Zoomers et al. 2024; Shen, Qian, and Zhan 2025; Liu et al. 2025). However, most of these methods vary only the number of Gaussians across progressive levels, without addressing the compression aspect, which is crucial for reducing transmission bandwidth. While GoDe (Di Sario et al. 2025)

*These authors contributed equally.
†Weiyao Lin is the corresponding author.

applies compression techniques to progressive Gaussians, it mainly focuses on increasing the number of Gaussians at each level, rather than improving their quality. Moreover, it fails to exploit the inherent context across levels, which could be leveraged to reuse information from previous levels for enhanced compression efficiency. On the other hand, although context modeling has demonstrated its effectiveness in existing 3DGS compression methods (Chen et al. 2024, 2025b; Zhan et al. 2025; Wang et al. 2024b), these approaches do not support progressive compression. For instance, HAC++ (Chen et al. 2025b, 2024) learns a sparse hash grid to capture contextual relationships among anchors, which are introduced in Scaffold-GS (Lu et al. 2024) to predict nearby 3D Gaussians. By leveraging the context, HAC++ predicts the quantization step and the distributions of anchor attributes to facilitate entropy coding (Ballé et al. 2018) and reduce bitstream size. Despite the advancements, these methods are single-rate, requiring separate trainings and generating independent bitstreams for different rate-distortion (R-D) trade-offs.

To address this, we propose **PCGS**, a novel approach for the **P**rogressive **C**ompression of 3D **G**aussian **S**platting. ***The core idea*** behind PCGS is to jointly and adaptively control both the quantity and quality of Gaussians (or anchors), enabling a progressive bitstream that incrementally enhances fidelity. Specifically, to regulate the quantity of anchors, we introduce a monotonically non-decreasing mask learning strategy, which progressively decodes additional anchors and Gaussians, which are seamlessly integrated with existing ones at the decoder side to improve fidelity. For anchor quality, we progressively refine existing anchors by employing trit-plane division (Lee et al. 2022), which provides more precise quantization results. Building upon this refinement, we further introduce a quantization context model that exploits correlations among quantized values across progressive levels. It leverages a trinomial distribution to more accurately estimate the probability of quantized values, thereby improving entropy coding efficiency and reducing the incremental bitstream size.

Our main contributions are summarized as follows.

- We propose **PCGS**, a novel progressive 3DGS compression framework jointly and progressively enhances anchor quantity and quality, enabling diverse on-demand applications, expanding the applicability of 3DGS.
- We introduce a *progressive masking strategy* that incrementally decodes additional anchors and Gaussians and integrates them with existing ones to enhance fidelity. Furthermore, we propose a *progressive quantization mechanism* that leverages quantization context to refine anchor quality and improve probability estimation.
- Extensive experiments across multiple datasets demonstrate that **PCGS** achieves effective progressivity while maintaining compression performance comparable to SoTA single-rate compression methods.

Related Work

3D Gaussian Splatting. 3DGS represents a scene using a collection of 3D Gaussians (Kerbl et al. 2023), each defined by learnable shape and appearance attributes. Gaussians can be quickly trained and rendered into 2D images given a viewpoint using rasterization (Zwicker et al. 2001). While achieving real-time rendering speed and high fidelity, 3DGS typically demands significant storage (Bagdasarian et al. 2024) and transmission bandwidth due to the large number of Gaussians and their associated attributes.

3DGS Compression. Many efforts (Bagdasarian et al. 2024) have been made to compress 3DGS while maintaining fidelity. Pruning techniques have been extensively explored, typically eliminating unimportant Gaussians through masks (Lee et al. 2024a,b), gradient-informed thresholds (Ali et al. 2024; Ali, Bae, and Tartaglione 2024), view-dependent metrics (Fan et al. 2024), and other importance-evaluation mechanisms (Zhang et al. 2024b; Hanson et al. 2024). (Yang et al. 2024) and (Fang and Wang 2024) propose to combine pruning with structural relations. Instead of pruning entire Gaussian, (Morgenstern et al. 2023; Papantonakis et al. 2024) partially prunes Gaussian attributes, *e.g.*, Sphere Harmonics. Vector quantization (VQ) (Navaneet et al. 2024; Xie et al. 2024a; Lee et al. 2024a; Fan et al. 2024; Niedermayr, Stumpfegger, and Westermann 2024; Dai, Liu, and Zhang 2025) groups similar Gaussians and represents them with a shared approximation. (Wang et al. 2024a) uses entropy-constrained VQ for a more compact representation.

Other methods (Lu et al. 2024; Chen et al. 2024, 2025b; Wang et al. 2024b; Zhan et al. 2025; Shin, Park, and Cho 2025) aim to exploit spatial structure correlations to reduce the size of 3DGS. Scaffold-GS (Lu et al. 2024) employs anchors to cluster region-near Gaussians and predicts the attributes of Gaussians within each cluster using an MLP. Building on Scaffold-GS, HAC (Chen et al. 2024) and HAC++ (Chen et al. 2025b) introduce hash-grid to explore the mutual context information between the attributes of anchors and hash features, facilitating entropy coding for a highly efficient representation. Different from HAC, CAT-3DGS (Zhan et al. 2025) employs multi-scale triplanes to capture spatial correlations. HEMGS (Liu, Chen, and Xu 2024) proposes a hybrid entropy model for 3DGS entropy coding. ContextGS (Wang et al. 2024b) and CompGS (Liu et al. 2024) reduce redundancy among anchors/Gaussians through context-aware designs. GaussianForest (Zhang et al. 2024a) constructs hybrid 3D Gaussians hierarchically, where implicit attributes are shared among sibling Gaussians to save storage. Optimization-free methods (Chen et al. 2025a; Xie et al. 2024b) are also explored to compress existing 3DGS rapidly in a feed-forward pass. However, they all require multiple trainings to achieve various compression levels, which are crucial in practice scenarios of diverse bandwidth and device resource constraints.

Progressive 3DGS. To accommodate on-demand applications, progressive 3DGS has been explored (Shi et al. 2024; Huang et al. 2024; Cheng et al. 2024; Zoomers et al. 2024; Shen, Qian, and Zhan 2025; Liu et al. 2025). These approaches selectively transmit only the necessary Gaussians based on rendering views or fluctuating network conditions. However, they do not consider compression. Given the large size of 3DGS, even partial transmission incurs non-negligible bit consumption. To address this, GoDe (Di Sario

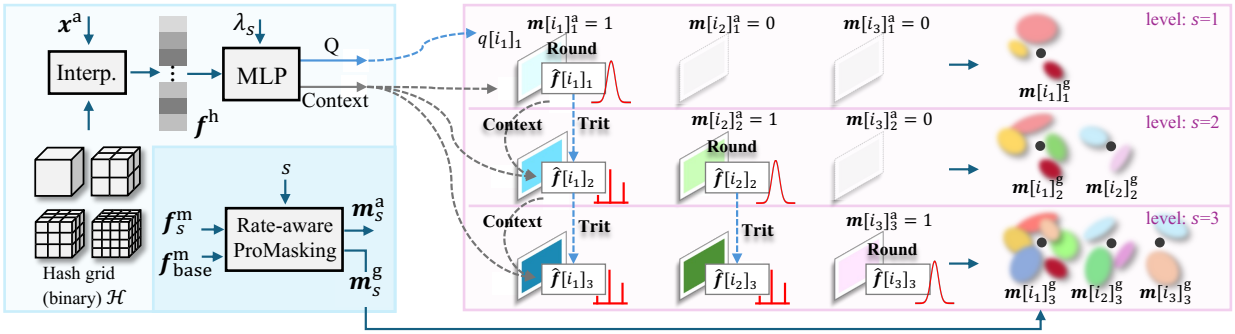


Figure 2: **Overview of the proposed PCGS**, which controls anchors in both quantity and quality in a progressive way, *i.e.*, progressively *decoding new anchors via masking control* and *refining existing anchors with finer quantization steps*. **Left**: Given anchor $\boldsymbol{x}^{\mathrm{a}}$, its position is interpolated within the binary hash grid to obtain the hash feature $\boldsymbol{f}^{\mathrm{h}}$. A rate-aware MLP, conditioned on the level information λ_s, utilizes $\boldsymbol{f}^{\mathrm{h}}$ to determine the quantization steps and provide context information for different progressivity levels. Additionally, the anchor and Gaussian masks $\boldsymbol{m}_s^a$ and $\boldsymbol{m}_s^g$ are obtained via the rate-aware progressive masking strategy from learnable features $\boldsymbol{f}_s^{\mathrm{m}}$ and $\boldsymbol{f}_{base}^{\mathrm{m}}$ (**bottom**). **Right**: At each level s, according to its mask $\boldsymbol{m}[i]_s^{\mathrm{a}}$, the i-th anchor either remains undecoded when $\boldsymbol{m}[i]_s^{\mathrm{a}} = 0$, be newly decoded when $\boldsymbol{m}[i]_s^{\mathrm{a}}$ transits from 0 to 1, or otherwise be refined.

et al. 2025) integrates compression techniques into progressive 3DGS by employing a learned mask to define Level-of-Detail (LoD) representations and compressing them using codebooks. However, it does not refine Gaussians from shallower levels as they propagate to deeper levels, resulting in a quality mismatch when fine details are required. In this work, we propose **PCGS**, which jointly optimizes both the quantity and the quality of anchors. This ensures anchors decoded at shallower levels are progressively refined to meet the fidelity requirements of deeper levels, effectively enabling progressive compression.

Preliminaries

Built on **3DGS** (Kerbl et al. 2023), Scaffold-GS (Lu et al. 2024) introduces anchors to cluster nearby Gaussians and neural predicts attributes from anchors. Each anchor consists of a location $\boldsymbol{x}^{\mathrm{a}} \in \mathbb{R}^3$ and its associated attributions $\left\{\boldsymbol{f}^a \in \mathbb{R}^{D^{\mathrm{a}}}, \boldsymbol{l} \in \mathbb{R}^6, \boldsymbol{o} \in \mathbb{R}^{3\times K}\right\}$, which represent anchor feature, scaling and learnable offsets, respectively. Scaffold-GS employs MLPs and scaling-based regularization to deduce Gaussian attributes for rendering. Based on Scaffold-GS, **HAC++** (Chen et al. 2024, 2025b) leverages a binary hash grid $\mathcal{H}$ to capture correlations among anchors $\boldsymbol{x}^{\mathrm{a}}$. The hash feature $\boldsymbol{f}^{\mathrm{h}}$ is obtained via interpolation and used to predict both the quantization step and the Gaussian distribution of anchor attributes for context-based entropy modeling. HAC++ also employs learnable masks to prune ineffective Gaussians and anchors, further enhancing compression. In this work, we improve HAC++ by introducing progressive compression, further expanding its applicability to varying bandwidth and storage constraints.

Methodology: PCGS

The framework of PCGS is in Fig. 2. Built upon HAC++ (Chen et al. 2025b), it constrains the entropy of 3DGS during training by controlling both the quantity and quality of anchors, thereby enabling progressive compression of 3DGS after training. **For quantity**, we introduce a rate-aware progressive masking strategy, as depicted in the lower-left part of Fig. 2. Specifically, at each progressive level, each anchor learns both an anchor-level mask $\boldsymbol{m}_s^{\mathrm{a}}$ and a Gaussian-level mask $\boldsymbol{m}_s^{\mathrm{g}}$, which are derived from the combination of a level-specific masking feature $\boldsymbol{f}_s^{\mathrm{m}}$ and a base masking feature $\boldsymbol{f}_{base}^{\mathrm{m}}$. Both masking features are explicitly defined as learnable parameters. At each level s, according to the i-th anchor mask $\boldsymbol{m}[i]_s^{\mathrm{a}}$, it will either (1) remain undecoded when $\boldsymbol{m}[i]_s^{\mathrm{a}} = 0$, (2) be newly decoded when $\boldsymbol{m}[i]_s^{\mathrm{a}}$ transits from 0 to 1 at this level, or (3) be refined if it was previously decoded at an earlier level, as illustrated in the right part of Fig. 2. To ensure a smooth progression, the mask values are carefully designed to be monotonically non-decreasing across progressive levels. As the compression levels deepen, more anchors and Gaussians are incorporated, thus improving rendering fidelity. **For quality**, we propose a progressive quantization strategy that continuously refines anchor values as levels progress deeper, enhancing fidelity. As shown in the right part of Fig. 2, when an anchor is first decoded at a level, its attribute is quantized using `Round`. At subsequent levels, the previously quantized value is refined through trit-plane quantization (Lee et al. 2022). For entropy modeling, we use a Gaussian distribution when an anchor is first decoded, as it undergoes `Round` quantization. For subsequent levels, we adopt a trinomial distribution to align with trit-plane quantization and effectively leverage level-wise context. By integrating these two strategies, each progressive level improves fidelity by both introducing new anchors and refining existing ones, striking an effective balance for progressive compression. For simplicity, we omit the anchor index $[i]$ in the following sections. *Unless otherwise specified, all operations are performed per anchor.*

Progressive Masking for Quantity Increase

Masking strategies have proven effective in 3DGS compression by explicitly reducing the number of parameters (*e.g.*, anchors or Gaussians) (Chen et al. 2025b; Lee et al. 2024a).

By adjusting the masking ratio, a balance between size and fidelity can be achieved. To enable progressive compression, we ensure that the valid ratio of anchors (or Gaussians) is monotonically non-decreasing as the levels progress. This allows newly decoded anchors to be integrated with existing ones, thereby improving rendering fidelity.

However, due to the unstructured nature of anchors, manually defining their importance and consequently determining the order of progressive masking are challenging. To address this issue, we introduce learnable masks that adaptively determine the progressive level s at which an anchor (or Gaussian) is initially decoded. Specifically, the masks $\boldsymbol{m}_s^{\mathrm{g}} \in \mathbb{R}^K$ are first defined at the Gaussian level as:

$$\boldsymbol{m}_s^{\mathrm{g}} = \mathbb{1}[\mathtt{Sig}(\boldsymbol{f}_{\mathrm{base}}^{\mathrm{m}} + \sum_{l=1}^{s} \mathtt{Sfp}(\boldsymbol{f}_l^{\mathrm{m}})) > \epsilon^{\mathrm{m}}], \tag{1}$$

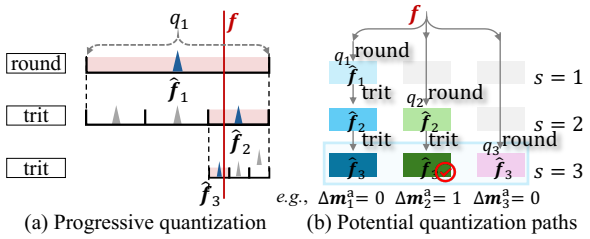
where $\mathtt{Sig}$ and $\mathtt{Sfp}$ denote the sigmoid and softplus activation functions, respectively, and ϵ^{m} is a constant threshold. Here, $\boldsymbol{f}_{\mathrm{base}}^{\mathrm{m}} \in \mathbb{R}^K$ and $\boldsymbol{f}_s^{\mathrm{m}} \in \mathbb{R}^K$ represent the base masking feature and the masking feature at level s, which are explicitly defined as learnable parameters. The gradient of $\boldsymbol{m}_s^{\mathrm{g}}$ is maintained by STE (Lee et al. 2024a; Bengio, Léonard, and Courville 2013). In Eq. 1, we apply a softplus activation to $\boldsymbol{f}_s^{\mathrm{m}}$ to ensure a positive output. Summing it with outputs from previous levels, we enforce a monotonically non-decreasing mask with respect to levels s. This guarantees once a Gaussian is decoded at any level s (*i.e.*, $\boldsymbol{m}_s^{\mathrm{g}} = 1$), it remains valid in all subsequent levels, ensuring reusability. Building on the Gaussian-level mask, we derive an anchor-level mask $\boldsymbol{m}_s^{\mathrm{a}} \in \mathbb{R}$ by determining whether at least one valid Gaussian exists within an anchor, following (Chen et al. 2025b). $\boldsymbol{m}_s^{\mathrm{a}}$ inherits the monotonically non-decreasing property of $\boldsymbol{m}_s^{\mathrm{g}}$. This monotonicity plays a crucial role in progressive compression. (1) it encourages anchors from previous levels to collaborate with newly introduced anchors at the current level. (2) it is fundamental to the progressive quantization strategy, as it guarantees that an anchor undergoing refinement can always access its coarse value from the previous level for a finer quantization and context modeling. By comparing the masks across levels, we can determine whether an anchor is newly decoded or refined from an existing one as:

$$\Delta\boldsymbol{m}_s^{\mathrm{a}} = \boldsymbol{m}_s^{\mathrm{a}} - \boldsymbol{m}_{s-1}^{\mathrm{a}} \tag{2}$$

where $\boldsymbol{m}_0^{\mathrm{a}}$ is set to 0. If $\Delta\boldsymbol{m}_s^{\mathrm{a}} = 1$, then the corresponding anchor is newly decoded at level s; otherwise, it is either previously decoded ($\boldsymbol{m}_{s-1}^{\mathrm{a}} = 1$) or invalid ($\boldsymbol{m}_s^{\mathrm{a}} = 0$). Note that $\Delta\boldsymbol{m}_s^{\mathrm{g}}$ can be calculated in the same way. However, merely increasing the number of anchors or Gaussians is insufficient, as the anchors decoded in shallower levels may lack the finer details required for the improved model fidelity at deeper levels. We further refine the quantization accuracy of existing anchors through progressive quantization.

Progressive Quantization and Context for Quality Enhancement

A finer quantization of anchor attributes maintains precision, enhancing fidelity, but also increases storage size due to the complexity of predicting fine-grained values. In our approach, a value is first decoded at an initial level and then progressively refined at subsequent levels. Upon initial decoding, the value is quantized using $\mathtt{Round}$, with its entropy modeled by a Gaussian distribution. In later levels, the value undergoes refinement by leveraging strong prior information from the previously quantized value. This exploitation of prior information is referred to as **context model**. The context model takes the prior information as context for input and outputs a set of distribution parameters for entropy modeling of the target value. A more accurate predicted distribution leads to lower entropy and hence smaller compressed size. To effectively exploit this context, we adopt trit-plane quantization and model entropy by a trinomial distribution, shown in Fig. 3 (a).



(a) Progressive quantization (b) Potential quantization paths

Figure 3: (a): Progressive quantization. The real value $\boldsymbol{f}$ is initially quantized at level $s = 1$ using $\mathtt{Round}$ with the step size q_1. It is then progressively refined using trit-plane quantization at levels $s = 2$, and 3, gradually approaching $\boldsymbol{f}$. (b): Three possible paths to reach the quantized value $\hat{\boldsymbol{f}}_3$ at the current level ($s = 3$) in training with $\Delta\boldsymbol{m}_2^{\mathrm{a}} = 1$ indicating that the middle path is selected by mask weighting.

For an anchor at its initial decoding, no prior quantization information is available. Thus, the value $\boldsymbol{f}$ is quantized by $\mathtt{Round}$, *i.e.*$\hat{\boldsymbol{f}}_s = \mathtt{Round}(\boldsymbol{f}, q_s)$, where the quantization step q_s is determined as,

$$q_s, \boldsymbol{\mu}_s, \boldsymbol{\sigma}_s = \mathrm{MLP}\left(\boldsymbol{f}^{\mathrm{h}}, \lambda_s\right) \tag{3}$$

where $\boldsymbol{f}^h$ is the hash feature obtained by querying anchor location $\boldsymbol{x}^{\mathrm{a}}$ in the hash grid $\mathcal{H}$. The parameter λ_s controls the R-D loss trade-off (detailed in Eq. 8). $\boldsymbol{\mu}_s$ and $\boldsymbol{\sigma}_s$ are the estimated Gaussian distribution parameters for entropy modeling. At initial decoding, the quantized values lack prior information and can be estimated as any value within the real number space. Thus, we adopt a Gaussian distribution ϕ, which imposes no preconditions on the value range:

$$p^{\mathrm{G}}(\hat{\boldsymbol{f}}_s) = \int_{\hat{\boldsymbol{f}}_s - \frac{q_s}{2}}^{\hat{\boldsymbol{f}}_s + \frac{q_s}{2}} \phi\left(x \mid \boldsymbol{\mu}_s, \boldsymbol{\sigma}_s\right) dx, \tag{4}$$

where $\hat{\boldsymbol{f}}_s$ is obtained by adding noise to $\boldsymbol{f}$ during training to keep gradient, and $\mathtt{Round}$ in testing (Chen et al. 2025b).

For an anchor that has been previously decoded, it is refined at the current level. Given the previously decoded value $\hat{\boldsymbol{f}}_{s-1}$ and its quantization step size q_{s-1}, the real value $\boldsymbol{f}$ is known to lie within range of $[\hat{\boldsymbol{f}}_{s-1} - \frac{q_{s-1}}{2}, \hat{\boldsymbol{f}}_{s-1} + \frac{q_{s-1}}{2})$. To

fully utilize this prior, we adopt trit-plane quantization (Lee et al. 2022), as illustrated in Fig. 3 (a). Specifically, trit-plane quantization refines the value by subdividing the range into three equal sub-intervals with midpoints: $\{\hat{\boldsymbol{f}}_{s^c}\}_{c=1,2,3} = \{\hat{\boldsymbol{f}}_{s-1} - \frac{q_{s-1}}{3}, \hat{\boldsymbol{f}}_{s-1}, \hat{\boldsymbol{f}}_{s-1} + \frac{q_{s-1}}{3}\}$ (*i.e.*, the three triangles in the second row in Fig. 3 (a)). The quantized value $\hat{\boldsymbol{f}}_s$ is assigned to the midpoint closest to $\boldsymbol{f}$:

$$\hat{\boldsymbol{f}}_s = \hat{\boldsymbol{f}}_{s^u}, \quad \text{where} \quad u = \arg\min_c |\boldsymbol{f} - \hat{\boldsymbol{f}}_{s^c}|. \tag{5}$$

If $\boldsymbol{f}$ lies exactly at a boundary between two sub-intervals, the value on the right is chosen. The quantization step is then updated as $q_s = \frac{q_{s-1}}{3}$ for recursive use of tirt-plane quantization in subsequent levels. For entropy modeling, we employ a trinomial distribution, which aligns naturally with the three candidate values in trit-plane quantization.

$$p^{\mathrm{T}}(\hat{\boldsymbol{f}}_s) = \boldsymbol{p}_{s^u}, \text{where } \{\boldsymbol{p}_{s^c}\}_{c=1,2,3} = \mathrm{MLP}(\hat{\boldsymbol{f}}_{s-1}, \boldsymbol{f}^h, \lambda_s). \tag{6}$$

The context leverages the strong prior from the previous level $s-1$, enabling more accurate probability estimation. Note that although anchor values in subsequent levels could also be quantized using `Round` and their entropy modeled by a Gaussian distribution, this does not fully exploit the prior information of previous levels, leading to suboptimal performance. Please refer to ablation study for evaluations.

An anchor may be initially decoded at different levels. Trit-plane quantization is a causal process that relies on the anchor's value at its first decoded level. However, due to the progressive masking strategy, during training an anchor may be initially decoded at different levels with varying quantization steps, potentially leading to different trit-plane quantized results. This is illustrated in Fig. 3 (b) as a lower triangular matrix, where different columns represent the anchor is first decoded at a different level (*i.e.*, $s = 1, 2$, or 3) and it leads to three different $\hat{\boldsymbol{f}}_3$ in training.

This is not a problem during testing, where the anchor's initial decoding level can be directly obtained using the trained $\Delta\boldsymbol{m}^{\mathrm{a}}$, and hence the desired $\hat{\boldsymbol{f}}_s$ can be decoded progressively. However, during training, since $\Delta\boldsymbol{m}^{\mathrm{a}}$ is still learnable, all possible $\hat{\boldsymbol{f}}_s$ must be enumerated and weighted by the mask $\Delta\boldsymbol{m}^{\mathrm{a}}$ to enable the joint optimization of both the mask and the quantized value. Here, enumerating all possible quantization values $\hat{\boldsymbol{f}}_s$ is complex and increases the learning difficulty. To address this issue, for the quantization step when an anchor is firstly decoded (*i.e.*, the `Round` steps at the diagonal of Fig. 3 (b)), we define it as $q_s = \frac{q_1}{3^{s-1}}$. Given the candidate values in the trit-plane as $\{\hat{\boldsymbol{f}}_{s^c}\}_{c=1,2,3} = \{\hat{\boldsymbol{f}}_{s-1} - \frac{q_{s-1}}{3}, \hat{\boldsymbol{f}}_{s-1}, \hat{\boldsymbol{f}}_{s-1} + \frac{q_{s-1}}{3}\}$, this formulation guarantees that the quantized value at the current level s remains identical, regardless of the level at which the anchor is initially decoded. This approach unifies the different paths illustrated in Fig. 3 (b) into a single path: during training, we directly apply `Round` to $\boldsymbol{f}$ using q_s at level s to obtain $\hat{\boldsymbol{f}}_s$, making it independent of $\Delta\boldsymbol{m}^{\mathrm{a}}_s$.

Training Losses for Progressive Compression

As levels progress deeper, new anchors and Gaussians are decoded, while existing anchors are refined, progressively enhancing the model's fidelity. At each training iteration, we randomly sample a level s, and compute the entropy loss based on the *incremental* bit at this level. For the first level ($s = 1$), the loss follows HAC++. For levels $s \geq 2$, entropy constraint L_{entropy} is from an *incremental* perspective:

$$\begin{aligned} L_{\text{entropy}} &= \sum_i^N b[i]_s, \quad \text{where for each anchor } i, \\ b_s &= \boldsymbol{m}^{\mathrm{a}}_s \Delta\boldsymbol{m}^{\mathrm{a}}_s \sum_{\boldsymbol{f}\in\{\boldsymbol{f}^a, \boldsymbol{l}\}} \sum_{j=1}^{D^f} \Big(-\log_2 p^{\mathrm{G}}(\hat{f}_{s,j})\Big) + \\ &\quad \boldsymbol{m}^{\mathrm{a}}_s (1 - \Delta\boldsymbol{m}^{\mathrm{a}}_s) \sum_{\boldsymbol{f}\in\{\boldsymbol{f}^a, \boldsymbol{l}\}} \sum_{j=1}^{D^f} \Big(-\log_2 p^{\mathrm{T}}(\hat{f}_{s,j})\Big) + \\ &\quad \sum_{k=1}^{K} \Delta\boldsymbol{m}^{\mathrm{g}}_{s,k} \sum_{j=1}^{3} \Big(-\log_2 p^{\mathrm{G}}(\hat{o}_{s,3k+j})\Big) \end{aligned} \tag{7}$$

where D^f is the dimension of f, and o and K are defined in preliminaries. The first and second terms respectively account for the bits needed for a newly decoded anchor at level-s (with $\Delta\boldsymbol{m}^{\mathrm{a}}_s = 1$), modeled as a Gaussian distribution in Eq. 4, and the bits needed for a refined anchor (with $\Delta\boldsymbol{m}^{\mathrm{a}}_s = 0$), modeled as a trinomial distribution in Eq. 6. The third term is the bits needed for any newly added Gaussians (with $\Delta\boldsymbol{m}^{\mathrm{g}}_{s,k} = 1$) associated with the anchor. For Gaussian offsets, only the quantity increases while previously decoded values remain unchanged. The final loss at level s is:

$$Loss = L_{\text{Scaffold}} + \lambda_s \frac{1}{N(D^a + 6 + 3K)} (L_{\text{entropy}} + L_{\text{hash}}). \tag{8}$$

where L_{Scaffold} and L_{hash} are the same as those defined in HAC++ (Chen et al. 2025b) and the trade-off λ_s varies with s to balance the R-D performance at different levels.

Experiments

Implementation. PCGS is built upon HAC++ (Chen et al. 2025b). The hyperparameters remain consistent with HAC++, except that we employ a set of λ_s to train the model progressively within a single training process. At each iteration, we randomly sample a level s and its corresponding λ_s from this set for training. In addition to the results trained by $30k$ iterations for a fair comparison with the standard protocol, we also report results trained with $40k$ to enable comparison with methods trained with more iterations. Experiments are conducted using an NVIDIA L40S GPU.

Baselines. Progressive compression of 3DGS still lacks exploration. Specifically, GoDe (Di Sario et al. 2025) organizes Gaussians into layers and applies codebooks for progressive compression. Other compression methods generate only a single rate per training. By adjusting training parameters and retraining the scene multiple times, we obtain the R-D curves for HAC (Chen et al. 2024), HAC++ (Chen et al. 2025b), ContextGS (Wang et al. 2024b), HEMGS (Liu, Chen, and Xu 2024), and CAT-3DGS (Zhan et al. 2025). We present the curves of these approaches in Fig. 4, as they exhibit the best compression performance among existing methods. For **all metrics** (*i.e.*, PSNR, SSIM (Wang

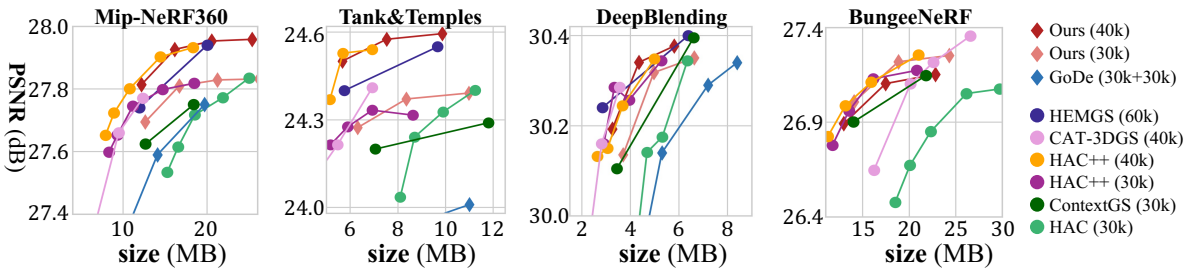


Figure 4: **R-D curve comparisons.** GoDe applies an additional finetune stage of $30k$ iterations to the Scaffold-GS which is originally trained for $30k$. Diamond ♢ represents **progressive compression** methods. Circle ◯ denotes **single-rate** methods.

et al. 2004), LPIPS (Zhang et al. 2018), size, training time, and encoding/decoding time) and results of more comparison methods (including the baselines 3DGS (Kerbl et al. 2023) and Scaffold-GS (Lu et al. 2024), and various compression methods (Wang et al. 2024a; Morgenstern et al. 2023; Navaneet et al. 2024; Fan et al. 2024; Girish, Gupta, and Shrivastava 2024; Liu et al. 2024; Lee et al. 2024a; Niedermayr, Stumpfegger, and Westermann 2024; Papantonakis et al. 2024; Xie et al. 2024b)), please refer to the Appendix.

Dataset. Real-world large-scale datasets are utilized, including Mip-NeRF360 (Barron et al. 2022), DeepBlending (Hedman et al. 2018), Tanks&Temples (Knapitsch et al. 2017) and BungeeNeRF (Xiangli et al. 2022).

Results. As shown in Fig. 4, although PCGS is trained only once to produce progressive representations, it achieves performance comparable to state-of-the-art single-rate methods. It supports compression into multiple progressive bitstreams, efficiently adapting to varying network and storage constraints, and outperforms the progressive method GoDe, which achieves progressivity merely by increasing the number of anchors. PCGS jointly optimizes both the quantity and quality of anchors, and effectively leverages contextual correlations across levels for more accurate entropy modeling, resulting in more compact incremental bitstreams. Notably, additional iterations do not improve performance on the BungeeNeRF dataset due to the scale discrepancy between training and testing views, which leads to overfitting. Compared to single-rate methods, PCGS outperforms HEMGS on most datasets, even with fewer iterations. Importantly, we enable the progressivity of the base model HAC++, while still achieving comparable performance.

Decoding Order. We introduce the decoding order of different components across levels. Background knowledge on the codec, including how bitstreams are encoded and decoded, is provided in the Appendix. PCGS enables a progressive decoding process by reusing existing bitstreams. Initially, shared components such as MLPs, anchor locations $\boldsymbol{x}^{\mathrm{a}}$, the hash grid $\mathcal{H}$, and masks $\boldsymbol{m}^{\mathrm{g}}$ (from which $\boldsymbol{m}^{\mathrm{a}}$ is derived) are decoded from the header information. Note that since anchor locations are encoded using GPCC (Chen et al. 2023), which relies on mutual information among anchors to reduce entropy, we encode all valid anchor locations together in the header, rather than encoding the incremental anchor locations at each level, to reduce the total bit consumption.

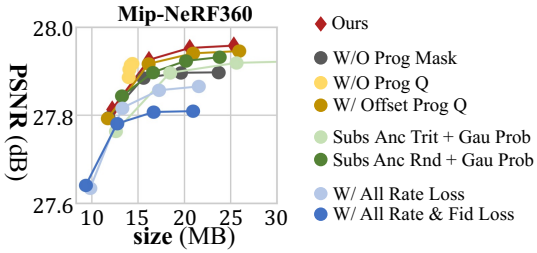


Figure 5: R-D curve of ablation study on Mip-NeRF360.

Based on the mask information, the model determines which anchors to be newly decoded or refined at each level. Specifically, at the first level ($s = 1$), attributes of valid anchors and Gaussian offsets are decoded using Gaussian distributions. At the next level ($s = 2$), two operations occur: (1) already decoded anchors are refined using the progressive quantization mechanism, with a trinomial distribution for probability estimation, and (2) newly introduced anchors and Gaussian offsets are decoded via a Gaussian distribution. This process continues consistently across all progressive levels, ensuring efficient and structured decoding.

Ablation Study

We conduct ablation studies on the Mip-NeRF360 dataset (Barron et al. 2022). Results are shown in Fig. 5. The ablations are analyzed from three perspectives: **(1) Effectiveness of Progressivity.** Disabling progressive masking (W/O Prog Mask, *i.e.*, keeping the mask identical across levels) prevents the model from improving fidelity in the high-rate segments due to the lack of additional anchors or Gaussians. When the progressive quantization mechanism is removed (W/O Prog Q), progressivity relies solely on the mask, resulting in a narrow rate range with limited fidelity improvements, as the quality of existing anchors remains fixed and fails to adapt to varying levels of details. Applying progressive quantization to offsets (W Offset Prog Q) shifts the model's focus toward optimizing offset sizes, which may increase overall rate consumption and impact fidelity. **(2) Quantization and Entropy Modeling of Subsequent Anchors:** For anchors that have been previously decoded, replacing the trinomial distribution with a Gaussian distribu-

	Size(MB)	PSNR(dB)	Train(s)	Enc(s)	Dec(s)	FPS
Ours	12.19	27.81	4044	13.0	20.5	145
	25.34	27.96		(13.0)+7.7	(20.5)+8.8	141
HAC++	10.81	27.80	3269	11.8	19.4	146
	18.32	27.93	3294	18.0	30.4	130

Table 1: **Complexity comparison** between PCGS and HAC++, both trained with $40k$ iterations on Mip-NeRF360.

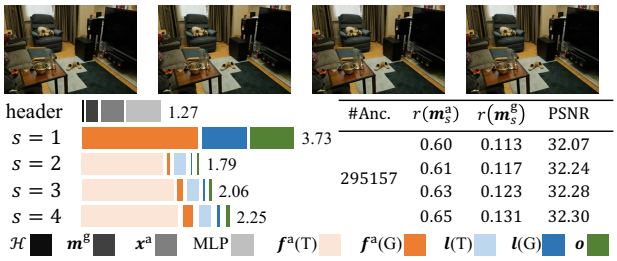


Figure 6: **Sizes of different components.** Total size (in MB) of header or each level is at the right of each bin. **Upper**: Qualitative results of scene *room* across levels. **Lower**: incremental size and mask information of levels. $r(\boldsymbol{m}_s^a)$ and $r(\boldsymbol{m}_s^g)$ are the *valid* mask ratios for anchors and Gaussians. Color meanings are in the **bottom**. (T) and (G) denote the attribute is refined or newly decoded, respectively.

tion (Subs Anc Trit + Gau Prob) leads to inaccurate probability estimation, as it fails to sufficiently exploit the quantization context across levels. Similarly, if progressive quantization of these subsequent-level anchors is replaced with direct Round and entropy modeling using a Gaussian distribution (Subs Anc Rnd + Gau Prob), the performance degrades due to ineffective adaptation to progressive refinement. **(3) Training Loss Design:** Using the accumulated entropy from all previous and current levels as loss (W/ All Rate Loss) causes the model to degrade in fidelity, as it does not explicitly consider the quality of reconstructions at earlier levels when optimizing the current level. However, if fidelity at previous levels is incorporated into the optimization (W/ All Rate & Fid Loss), the performance becomes worse, as this introduces excessive complexity and hinders the model from effectively balancing the R-D trade-off across levels.

Complexity Analysis

As shown in Tab. 1, **for training:** PCGS incurs longer time due to its additional components. Although computing the progressive masking is fast, all possible masking conditions must remain differentiable during training, which increases complexity. **For coding:** PCGS achieves progressive compression, resulting in significantly reduced time for coding of incremental bitstreams, *e.g.*, given bitstreams from the previous rate point, decoding the next level requires only an additional 8.8s, whereas HAC++ takes 30.4s to decode the second rate point. Importantly, PCGS requires **a single training run** to get multiple progressive rates, making it more efficient. *This one-time training to support multiple progressive RD tradeoffs is our key novelty.*

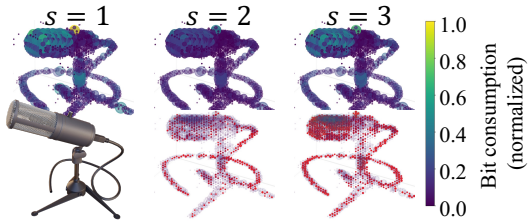


Figure 7: **Visualization of bit allocation across progressive levels in the *mic* scene**. Following HAC++, we voxelize the scene, with each voxel represented by a ball. **Upper**: The size and color of balls indicate number of anchors the amount of *incremental* bits within it. **Lower-right**: Red balls are newly decoded anchors at the current level.

Progressive Bit Allocation and Visualization

Allocation: The statistical information on bit allocations and mask ratios across levels is in Fig. 6. More samples are in the Appendix. We first encode and store the header information, including MLPs, anchor locations, the hash grid and masks. Then, we present the relative bit consumption at each level. For anchors at levels $s \geq 2$, the bit consumption is from two parts: trit-plane refinement, and newly decoded anchors and Gaussian offsets. While the bits from newly decoded anchors and Gaussian offsets share a small portion, they play a crucial role in improving fidelity and enhancing the progressivity of the model. Moreover, both $r(\boldsymbol{m}_s^a)$ and $r(\boldsymbol{m}_s^g)$ increase as the level deepens, indicating more new anchors and Gaussians are decoded, which highlights the effectiveness of the progressive masking strategy.
Visualization: We visualize the bit allocation of *mic* scene, shown in Fig.7 (**upper**). At the first level (*i.e.*, $s = 1$), the highest number of bits is consumed, as it provides basic information for subsequent levels. Gaussian distribution used at this level lacks prior information, making probability estimation more challenging. At higher levels (*i.e.*, $s = 2$ and 3), they consume fewer bits, which demonstrates the effectiveness of the trinomial distribution. Moreover, as shown in Fig. 7 (**lower-right**), new anchors are decoded at subsequent levels. Complex areas with higher bit consumptions activate more new anchors for further refinement of fidelity.

Generalizability of the Progressive Masking

We have validated the generalizability of our progressive masking strategy on the vanilla Scaffold-GS (Lu et al. 2024) method. Detailed results are provided in the Appendix.

Conclusion

We propose PCGS, a novel framework for progressive compression of 3DGS. Controlling **both the quantity and quality** of anchors, PCGS achieves performance comparable to SoTA single-rate methods. The progressive nature of PCGS makes it well-suited for on-demand applications, where dynamic bandwidth and diversion storage conditions often arise, significantly broadening the applicability of 3DGS.

Acknowledgements

The paper is supported in part by the National Natural Science Foundation of China (No. 62325109, U21B2013), and in part by the Shanghai 'The Belt and Road' Young Scholar Exchange Grant (24510742000). MH is supported by funding from The Australian Research Council Discovery Program DP230101176.

References

Ali, M. S.; Bae, S.-H.; and Tartaglione, E. 2024. ELMGS: Enhancing memory and computation scaLability through coMpression for 3D Gaussian Splatting. arXiv:2410.23213.

Ali, M. S.; Qamar, M.; Bae, S.-H.; and Tartaglione, E. 2024. Trimming the Fat: Efficient Compression of 3D Gaussian Splats through Pruning. *arXiv preprint arXiv:2406.18214*.

Bagdasarian, M. T.; Knoll, P.; Li, Y.-H.; Barthel, F.; Hilsmann, A.; Eisert, P.; and Morgenstern, W. 2024. 3dgs. zip: A survey on 3d gaussian splatting compression methods. *arXiv preprint arXiv:2407.09510*.

Ballé, J.; Minnen, D.; Singh, S.; Hwang, S. J.; and Johnston, N. 2018. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*.

Barron, J. T.; Mildenhall, B.; Verbin, D.; Srinivasan, P. P.; and Hedman, P. 2022. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5470–5479.

Bengio, Y.; Léonard, N.; and Courville, A. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Chen, A.; Mao, S.; Li, Z.; Xu, M.; Zhang, H.; Niyato, D.; and Han, Z. 2023. An introduction to point cloud compression standards. *GetMobile: Mobile Computing and Communications*, 27(1): 11–17.

Chen, Y.; Wu, Q.; Li, M.; Lin, W.; Harandi, M.; and Cai, J. 2025a. Fast Feedforward 3D Gaussian Splatting Compression. In *The Thirteenth International Conference on Learning Representations*.

Chen, Y.; Wu, Q.; Lin, W.; Harandi, M.; and Cai, J. 2024. HAC: Hash-grid Assisted Context for 3D Gaussian Splatting Compression. In *European Conference on Computer Vision*.

Chen, Y.; Wu, Q.; Lin, W.; Harandi, M.; and Cai, J. 2025b. HAC++: Towards 100X Compression of 3D Gaussian Splatting. *arXiv preprint arXiv:2501.12255*.

Cheng, K.; Long, X.; Yang, K.; Yao, Y.; Yin, W.; Ma, Y.; Wang, W.; and Chen, X. 2024. Gaussianpro: 3d gaussian splatting with progressive propagation. In *Forty-first International Conference on Machine Learning*.

Dai, Z.; Liu, T.; and Zhang, Y. 2025. Efficient Decoupled Feature 3D Gaussian Splatting via Hierarchical Compression. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 11156–11166.

Di Sario, F.; Renzulli, R.; Grangetto, M.; Sugimoto, A.; and Tartaglione, E. 2025. GoDe: Gaussians on Demand for Progressive Level of Detail and Scalable Compression. *arXiv preprint arXiv:2501.13558*.

Fan, Z.; Wang, K.; Wen, K.; Zhu, Z.; Xu, D.; and Wang, Z. 2024. LightGaussian: Unbounded 3D Gaussian Compression with 15x Reduction and 200+ FPS. *Advances in neural information processing systems*.

Fang, G.; and Wang, B. 2024. Mini-Splatting: Representing Scenes with a Constrained Number of Gaussians. In *European Conference on Computer Vision*.

Girish, S.; Gupta, K.; and Shrivastava, A. 2024. Eagles: Efficient accelerated 3d gaussians with lightweight encodings. In *European Conference on Computer Vision*.

Hanson, A.; Tu, A.; Singla, V.; Jayawardhana, M.; Zwicker, M.; and Goldstein, T. 2024. PUP 3D-GS: Principled Uncertainty Pruning for 3D Gaussian Splatting. arXiv:2406.10219.

Hedman, P.; Philip, J.; Price, T.; Frahm, J.-M.; Drettakis, G.; and Brostow, G. 2018. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (ToG)*, 37(6): 1–15.

Huang, H.; Huang, W.; Yang, Q.; Xu, Y.; and li, Z. 2024. A Hierarchical Compression Technique for 3D Gaussian Splatting Compression. arXiv:2411.06976.

Kerbl, B.; Kopanas, G.; Leimkühler, T.; and Drettakis, G. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 42(4).

Knapitsch, A.; Park, J.; Zhou, Q.-Y.; and Koltun, V. 2017. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4): 1–13.

Lee, J. C.; Rho, D.; Sun, X.; Ko, J. H.; and Park, E. 2024a. Compact 3D Gaussian Representation for Radiance Field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Lee, J. C.; Rho, D.; Sun, X.; Ko, J. H.; and Park, E. 2024b. Compact 3D Gaussian Splatting for Static and Dynamic Radiance Fields. arXiv:2408.03822.

Lee, J.-H.; Jeon, S.; Choi, K. P.; Park, Y.; and Kim, C.-S. 2022. DPICT: Deep progressive image compression using trit-planes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 16113–16122.

Liu, H.; Wang, Y.; Li, C.; Cai, R.; Wang, K.; Li, W.; Molchanov, P.; Wang, P.; and Wang, Z. 2025. FlexGS: Train Once, Deploy Everywhere with Many-in-One Flexible 3D Gaussian Splatting. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 16336–16345.

Liu, L.; Chen, Z.; and Xu, D. 2024. HEMGS: A Hybrid Entropy Model for 3D Gaussian Splatting Data Compression. arXiv:2411.18473.

Liu, X.; Wu, X.; Zhang, P.; Wang, S.; Li, Z.; and Kwong, S. 2024. Compgs: Efficient 3d scene representation via compressed gaussian splatting. In *Proceedings of the 32nd ACM International Conference on Multimedia*, 2936–2944.

Lu, T.; Yu, M.; Xu, L.; Xiangli, Y.; Wang, L.; Lin, D.; and Dai, B. 2024. Scaffold-GS: Structured 3D Gaussians for View-Adaptive Rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Morgenstern, W.; Barthel, F.; Hilsmann, A.; and Eisert, P. 2023. Compact 3D Scene Representation via Self-Organizing Gaussian Grids. *arXiv preprint arXiv:2312.13299*.

Navaneet, K.; Meibodi, K. P.; Koohpayegani, S. A.; and Pirsiavash, H. 2024. Compact3d: Compressing gaussian splat radiance field models with vector quantization. In *European Conference on Computer Vision*.

Niedermayr, S.; Stumpfegger, J.; and Westermann, R. 2024. Compressed 3d gaussian splatting for accelerated novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10349–10358.

Papantonakis, P.; Kopanas, G.; Kerbl, B.; Lanvin, A.; and Drettakis, G. 2024. Reducing the memory footprint of 3d gaussian splatting. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 7(1): 1–17.

Shen, J.; Qian, Y.; and Zhan, X. 2025. LOD-GS: Achieving Levels of Detail using Scalable Gaussian Soup. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 671–680.

Shi, Y.; Gasparini, S.; Morin, G.; and Ooi, W. T. 2024. LapisGS: Layered Progressive 3D Gaussian Splatting for Adaptive Streaming. *arXiv preprint arXiv:2408.14823*.

Shin, S.; Park, J.; and Cho, S. 2025. Locality-aware Gaussian Compression for Fast and High-quality Rendering. arXiv:2501.05757.

Wang, H.; Zhu, H.; He, T.; Feng, R.; Deng, J.; Bian, J.; and Chen, Z. 2024a. End-to-End Rate-Distortion Optimized 3D Gaussian Representation. In *European Conference on Computer Vision*.

Wang, Y.; Li, Z.; Guo, L.; Yang, W.; Kot, A. C.; and Wen, B. 2024b. ContextGS: Compact 3D Gaussian Splatting with Anchor Level Context Model. *Advances in neural information processing systems*.

Wang, Z.; Bovik, A. C.; Sheikh, H. R.; and Simoncelli, E. P. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4): 600–612.

Xiangli, Y.; Xu, L.; Pan, X.; Zhao, N.; Rao, A.; Theobalt, C.; Dai, B.; and Lin, D. 2022. Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering. In *European conference on computer vision*, 106–122.

Xie, S.; Liu, J.; Zhang, W.; Ge, S.; Pan, S.; Tang, C.; Bai, Y.; and Wang, Z. 2024a. SizeGS: Size-aware Compression of 3D Gaussians with Hierarchical Mixed Precision Quantization. arXiv:2412.05808.

Xie, S.; Zhang, W.; Tang, C.; Bai, Y.; Lu, R.; Ge, S.; and Wang, Z. 2024b. MesonGS: Post-training Compression of 3D Gaussians via Efficient Attribute Transformation. arXiv:2409.09756.

Yang, R.; Zhu, Z.; Jiang, Z.; Ye, B.; Chen, X.; Zhang, Y.; Chen, Y.; Zhao, J.; and Zhao, H. 2024. Spectrally Pruned Gaussian Fields with Neural Compensation. *arXiv preprint arXiv:2405.00676*.

Zhan, Y.-T.; Ho, C.-Y.; Yang, H.; Chen, Y.-H.; Chiang, J. C.; Liu, Y.-L.; and Peng, W.-H. 2025. CAT-3DGS: A Context-Adaptive Triplane Approach to Rate-Distortion-Optimized 3DGS Compression. In *The Thirteenth International Conference on Learning Representations*.

Zhang, F.; Luo, Y.; Zhang, T.; Zhang, L.; and Huang, Z. 2024a. GaussianForest: Hierarchical-Hybrid 3D Gaussian Splatting for Compressed Scene Modeling. arXiv:2406.08759.

Zhang, R.; Isola, P.; Efros, A. A.; Shechtman, E.; and Wang, O. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 586–595.

Zhang, Z.; Song, T.; Lee, Y.; Yang, L.; Peng, C.; Chellappa, R.; and Fan, D. 2024b. LP-3DGS: Learning to Prune 3D Gaussian Splatting. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Zoomers, B.; Wijnants, M.; Molenaers, I.; Vanherck, J.; Put, J.; Jorissen, L.; and Michiels, N. 2024. PRoGS: Progressive Rendering of Gaussian Splats. *arXiv preprint arXiv:2409.01761*.

Zwicker, M.; Pfister, H.; van Baar, J.; and Gross, M. 2001. EWA volume splatting. In *Proceedings Visualization, 2001. VIS '01.*, 29–538.