NEURAL REPARAMETERIZATION FOR 3D GAUSSIAN SPLATTING COMPRESSION

Anonymous authors
Paper under double-blind review

ABSTRACT

While effective in diverse applications, including 3D/4D novel-view synthesis, Gaussian Splatting typically requires substantial storage to represent high-quality scenes. Many existing compression methods partially formulate this problem as point-based coding, treating Gaussians as a set of primitives with positions (or anchor locations) and rich appearance and shape attributes. These positions must therefore be encoded at high precision, as the storage cost scales with the number of primitives. However, many different sets of Gaussians can render the same scene equally well, so it is their spatial distribution that carries visual information and needs to be transmitted rather than any particular set. In this work, we propose **GSRC**, a Gaussian Splatting compression framework that achieves efficient representation and compression through neural reparameterization, which represents the Gaussians implicitly as the output of two networks instead of storing them. The first network maps a set of deterministic pseudo-random samples to a coarse point cloud representing the scene geometry, while the second one predicts the fine attributes of the Gaussians at these points using multi-resolution hash tables. To enable compact storage, we apply rate-aware training and entropy-code the resulting network parameters. Our results demonstrate that point cloud compression is not strictly necessary to achieve high-quality Gaussian Splatting representations, and that GSRC yields strong compression performance while preserving rendering quality, through implicit representation learning and model compression alone. Before entropy coding, the reparameterized model is 4× smaller than 3D-GS on Mip-NeRF 360 at 0.5 dB higher PSNR, which we attribute to the prior of the network. After coding, GSRC is about 100× smaller than 3D-GS at 0.25 dB higher PSNR, outperforms recent codecs with context models in most cases, especially in SSIM and LPIPS, and requires 49% fewer bytes than HAC++ at equal PSNR while decoding 24× faster. The implementation will be released.

1 INTRODUCTION

Recent advances in Gaussian Splatting enable high-quality scene representation, but at a higher storage cost than implicit representations such as Neural Radiance Fields (NeRF) (Mildenhall et al., 2021) and its variants: vanilla 3D-GS (Kerbl et al., 2023) can cost on the order of gigabytes to represent a large-scale scene. To reduce the storage requirements of Gaussian Splatting for practical deployment, many existing compression methods cast this problem as point-based coding (Graziosi et al., 2020; Schwarz et al., 2019), treating Gaussians as a set of primitives with positions (or anchor locations) and rich appearance and shape attributes. However, point-based coding remains challenging: rendering quality is highly sensitive to geometric precision and the attributes are coded against a point set that is already fixed, so

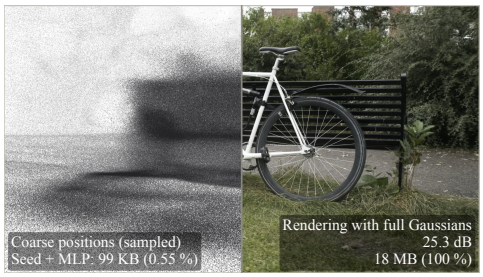


Figure 1: A view rendered by GSRC, which stores no Gaussians, positions or anchors. **Left:** the coarse point cloud generated by a lightweight MLP from pseudo-random samples drawn from a seed. **Right:** the Gaussians predicted at those points by the fine MLP and rasterized.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

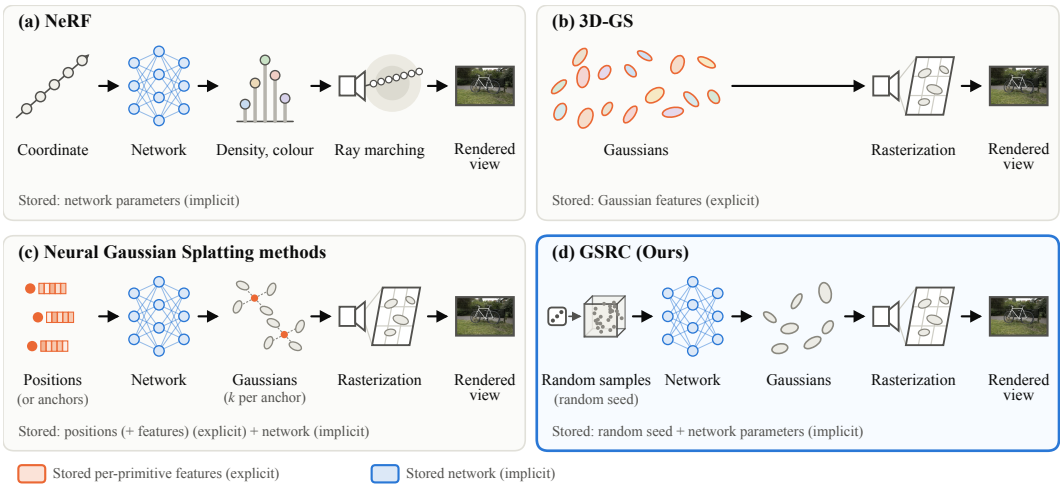


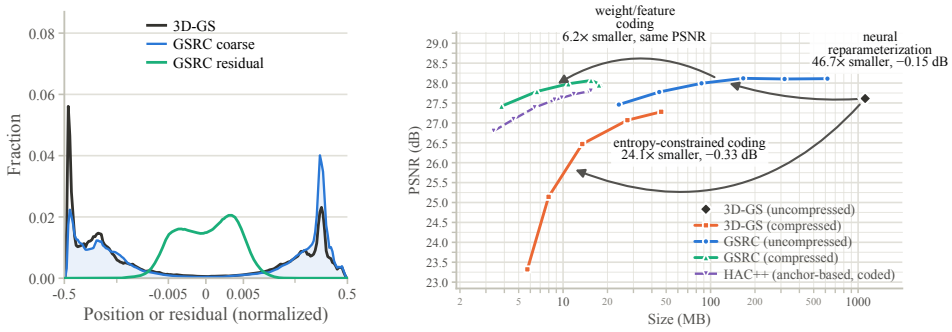
Figure 2: Comparison of NeRF, 3D-GS, existing neural Gaussian Splatting methods, and the proposed GSRC. **NeRF (a)** is fully implicit, but the network is evaluated along every ray, so rendering is costly. **3D-GS (b)** rasterizes in real time, but its storage scales linearly with the number of primitives. **Existing neural Gaussian Splatting methods (c)** decode the Gaussians from a network, whereas the positions or anchors at which it is queried are stored explicitly and must be coded at high precision. **GSRC (d)** represents the Gaussians implicitly by neural reparameterization. The network maps deterministic pseudo-random samples to a coarse point cloud and then to the fine attributes of the Gaussians, which are rasterized as in 3D-GS. Only the network parameters are coded, so no point cloud is stored, and the storage is nearly independent of the number of Gaussians.

the geometry must be carried at high precision and commonly accounts for 15–45% of the bitstream (Lee et al., 2024; Wang et al., 2024a; Chen et al., 2024; Wang et al., 2024b; Xie et al., 2024; Papantonakis et al., 2024; Zhan et al., 2025a).

Furthermore, the storage of the remaining attributes, such as the color and shape of each Gaussian (Niedermayr et al., 2024; Lee et al., 2024), or the latent features representing them (Lu et al., 2024; Chen et al., 2024; Tang et al., 2025; Wu & Tuytelaars, 2024), scales linearly with the number of points, and coding them relies on advanced entropy models (Chen et al., 2025c; Wang et al., 2024b; Liu et al., 2024a) that predict the distribution by exploiting redundancy. Figure 2b–c shows the two cases, where either every Gaussian is stored, or a point set with its features is stored together with a network that decodes them. In contrast, recent neural codecs use an overfitted representation to code images and videos (Dupont et al., 2021; 2022; Chen et al., 2021; Ladune et al., 2023; Kwan et al., 2023; 2024; 2026), where the network not only serves as a decoder for the positional encoding or the latent, but also captures redundancy across space and time. Figure 3b contrasts the two pathways: compressing an explicit Gaussian set, or reparameterizing the Gaussians and compressing the network weights.

The exact positions of the Gaussians need not be transmitted at all. A Gaussian Splatting scene is not identified by one particular set of primitives (Kheradmand et al., 2024). Many different sets render the same scene equally well, so what carries visual information is the spatial distribution of the Gaussians rather than the identity of individual points. This suggests that the Gaussians can be coded efficiently by coding a distribution of positions from which sets can be sampled, rather than coding one such set explicitly. Figure 3a shows the distinction on *Bicycle*. Positions generated from a seed follow the same distribution as those of a trained 3D-GS model, and the coarse positions cost 0.19 bits per point, against 45 bits to code them explicitly, and the fine positions add 6.1 bits per point, spent on a residual of lower entropy. Figure 1 shows a view rendered from the decoded bitstream beside the coarse positions it is generated from, none of which are stored.

In this work, we propose a novel framework, Gaussian Splatting Reparameterization Compression (GSRC), which reparameterizes the Gaussians that represent the scene as the output of a neural network (Ulyanov et al., 2018; Hoyer et al., 2019). Instead of storing an explicit point cloud or per-primitive features, GSRC stores a small network that generates the entire Gaussian set, at a cost that does not scale with the number of Gaussians. Both before and after compression, the



(a) Position distributions on *Bicycle*.

(b) Two pathways to compression.

Figure 3: **Why the positions need not be coded, and what this buys.** (a) The coarse positions follow the distribution of a trained 3D-GS model, and the residual is narrow and centered at zero (asinh axis). Per point, coding the positions costs 45 bits, whereas GSRC spends 0.19 on the coarse positions and 6.1 on the fine ones. (b) Mip-NeRF 360 results (average of all scenes). With neural reparameterization and weight/feature coding, and no context model, GSRC outperforms both 3D-GS compressed directly with the same rate-aware fine-tuning and HAC++, which uses a context model.

reparameterized Gaussians match or exceed the quality of strong explicit and anchor-based models, and surpass 3D-GS trained from the same initialization in PSNR, which we attribute to the prior of the network, as for deep image priors (Ulyanov et al., 2018). Although the representation is fully neural, it is decoded only once: a single forward pass produces an explicit set of Gaussians, which the the standard 3D-GS rasterizer renders at the speed of ordinary Gaussian Splatting, and. Unlike NeRF-style implicit models, no network is evaluated per ray sample(Figure 2d).

Specifically, GSRC employs one coarse MLP and one fine MLP. The coarse MLP maps pseudo-random samples, drawn from a seed shared by the encoder and the decoder, to coarse positions, and the fine MLP, augmented with hash tables, predicts the full Gaussian parameters at these positions with high quality. Since the Gaussians are generated, their number N sets the rendering cost, while the table size T sets the rate. The network weights are fine-tuned with a rate–distortion objective and entropy-coded without a context model, enabling fast, parallel encoding and decoding of the bitstream in chunks.

Our contributions are as follows:

- To the best of our knowledge, GSRC is the **first scene-specific 3D-GS compression framework that transmits no explicit Gaussian or anchor set**, representing a scene as a compact neural representation, whose coarse positions are generated from a shared seed and never stored.
- We **reparameterize the Gaussians with a coarse-to-fine network**: a coarse MLP maps random samples to a coarse point set, and a fine MLP predicts the full Gaussian parameters, so that the bitstream consists of network parameters alone.
- Since the primitives are generated rather than stored, **the number of Gaussians and the rate are nearly decoupled and can be configured separately**, and GSRC **does not use a context model**, enabling parallel entropy coding.

Our experiments on novel-view synthesis datasets show that, **both before and after compression**, GSRC matches or exceeds the quality of strong Gaussian Splatting models. Before coding, the reparameterized model is 4× smaller than 3D-GS on Mip-NeRF 360 at 0.5 dB higher PSNR, and ahead of 3D-GS trained from the same initialization in PSNR on every dataset. After coding, it is 99× smaller than 3D-GS at 0.25 dB higher PSNR and **outperforms the context-model codecs in most cases**, especially in SSIM and LPIPS: on Mip-NeRF 360 it needs 49%, 59% and 72% fewer bytes than HAC++ at equal PSNR, SSIM and LPIPS, while decoding 24× faster than HAC++ and 18–79× faster than the other context-model codecs.

2 Related Work

Gaussian Splatting Compaction. 3D Gaussian Splatting (Kerbl et al., 2023) enables high-fidelity, real-time rendering of 3D scenes via explicit splatting. However, this performance comes at a high storage cost due to the large number of unorganized primitives. To address this, early work focused on reducing the number of Gaussians through different pruning strategies, removing primitives with low opacity or low accumulated contribution to the training views (Fan et al., 2024; Fang & Wang, 2024; Lee et al., 2024). Later work estimates the contribution of each primitive more carefully, through rendering sensitivity (Hanson et al., 2025b;a) or sparsity-constrained optimization (Zhang et al., 2025), and pushes the count down to a few hundred thousand Gaussians with limited loss in quality (Lee et al., 2025). Beyond pruning, recent work reorganizes Gaussians into more efficient structures. Anchor points, for example, were introduced by Lu et al. (2024) to hierarchically manage local Gaussians, using lightweight MLPs to predict attributes and reduce redundancy. To facilitate Level-of-Detail management for large-scale or dynamic scenes, tree-based or hierarchical structures have been adopted (Kerbl et al., 2024; Ren et al., 2024), and self-organizing grids have been explored to normalize primitive distributions (Morgenstern et al., 2024). Efforts to bridge implicit neural fields with explicit splatting (Wu & Tuytelaars, 2024; Shin et al., 2025; Tang et al., 2025) offer alternative compact representations, but still require storing per-primitive positions or attributes. Despite the effective reduction in primitive counts, these approaches typically retain an explicit list-based representation, whose storage grows with the number of primitives.

Gaussian Splatting Compression. Complementary to reducing the number of primitives, another stream of research focuses on compressing the attributes (e.g., color, covariance, rotation) of each Gaussian using advanced coding techniques. For instance, sensitivity-aware quantization combined with entropy coding has been employed to reduce bit precision (Niedermayr et al., 2024). To address repetitive patterns in attributes, vector quantization schemes (Navaneet et al., 2024; Girish et al., 2024; Morgenstern et al., 2026) cluster Gaussian parameters into codebooks to reduce storage. To further exploit spatial redundancy, recent approaches incorporate rate–distortion optimization directly into the training pipeline (Wang et al., 2024a). More structurally, hash-grid assisted context models (Chen et al., 2024; 2025c) and tri-plane representations (Zhan et al., 2025a;b) have been proposed to query anchor attributes by leveraging the mutual information between spatial features and Gaussian attributes. Other strategies include anchor-level context modeling (Wang et al., 2024b), predicting coupled primitives from anchor embeddings (Liu et al., 2024b; 2025), hybrid entropy models (Yang et al., 2025), and progressive compression schemes (Di Sario et al., 2025; Chen et al., 2025a). Feed-forward codecs (Chen et al., 2025b) compress an existing Gaussian set in a single pass without per-scene optimization. Like the methods above, they keep an explicit primitive set as the coded representation.

3 Method

GSRC reparameterizes the Gaussians that represent a scene as the output of a network (Figure 4). As in 3D-GS (Kerbl et al., 2023), each Gaussian $\mathbf{g}_i = (\mathbf{x}_i, \mathbf{o}_i, \mathbf{f}_i, \mathbf{s}_i, \mathbf{q}_i)$ holds a position, an opacity, appearance coefficients, a scale and a rotation. In GSRC, none of them is stored: the Gaussians $\mathcal{G}_\theta(s)$ are generated from a seed s by the weights $\theta = (\theta_{\text{coarse}}, \theta_{\text{fine}})$ of the coarse and the fine MLP, which are optimized and stored instead. A coarse MLP maps N pseudo-random samples, drawn from a seed shared by the encoder and the decoder, to coarse positions (§3.1), and a fine MLP with hash tables whose size is set by T predicts all Gaussian parameters at those positions (§3.2). The number of Gaussians N therefore sets the rendering cost, and T decides the rate. All positions live in a normalized frame, where PCA alignment and contraction map the scene into the unit cube (Barron et al., 2022). The model is trained through the rasterizer (Kerbl et al., 2023), fine-tuned for the rate–distortion trade-off, and its weights are entropy-coded into a bitstream (§3.3, §3.4).

3.1 Distribution Modeling of the Point Cloud

Existing codecs store the positions of the Gaussians or anchors at high precision, since rendering is sensitive to them. A specific point set need not be transmitted, since different sets of Gaussians can render the same scene equally well. Specifically, any set of points sampled from the spatial distribution can represent the scene, once their corresponding features are learned. In GSRC, the coarse MLP therefore reparameterizes the point set as a generator of that distribution, which can

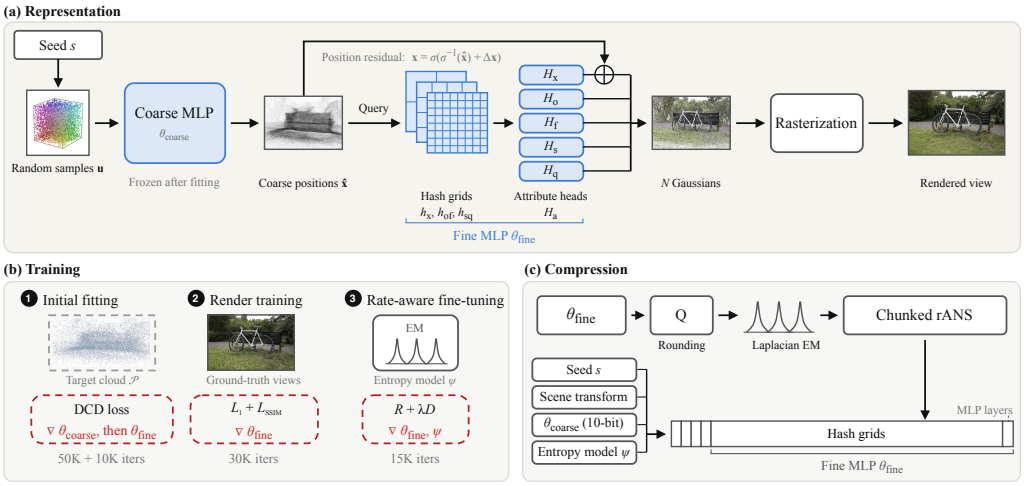


Figure 4: **Overview of GSRC.** (a) A seed generates N pseudo-random samples, which the coarse MLP (θ_{coarse}) maps to a coarse point cloud; the fine MLP (θ_{fine}), made of hash grids and attribute heads, then predicts the Gaussians at these positions, which are rasterized as in 3D-GS. (b) The networks are first fitted to a target point cloud, then trained on the training views, and finally fine-tuned with the rate term. (c) Only the network weights are coded, so none of the Gaussians is stored.

produce any number of points from a given seed. This decomposes the geometry into a coarse model, which can be coded at a very low rate, and a residual position offset. Since the coarse model already carries the spatial structure (Figure 1), the remaining offset is small and carries only fine-scale detail. The fine MLP then predicts it like any other attribute, from the multi-resolution hash grids that capture its local correlation, so the full positions need neither a separate geometry coder nor a context model. As Figure 3a shows, the coarse positions follow the distribution of a trained 3D-GS model, and the residual is narrow and zero-centered. Even where the learned distribution misses complex content such as sharp geometry, the attributes and the position offset (§3.2) compensate for the rendering error.

Fitting objective. Given a target point cloud $\mathcal{P}$ that holds the scene geometry, such as an SfM reconstruction or the output of a Gaussian initialization method (Appendix A.3), GSRC views that set as samples of an underlying spatial distribution, and trains the coarse MLP to map uniform samples $\mathbf{u}_i \sim \mathcal{U}([0,1]^3)$ to coarse positions $\hat{\mathbf{x}}_i$ that follow the same distribution, by minimizing the loss

$$\mathcal{L}_{\text{fit}} = D_{\text{fit}}\big(\{\hat{\mathbf{x}}_i\}_{i=1}^{N},\ \mathcal{P}\big). \tag{1}$$

Here, D_{fit} matches the two sets as distributions, and the density-aware Chamfer distance (Wu et al., 2021) is used. Fresh random samples are drawn at each training step, so that the coarse MLP is fitted to the distribution itself. By minimizing $\mathcal{L}_{\text{fit}}$, the coarse MLP becomes a generator, and the whole point set can be reproduced from a fixed seed. The target point cloud is used for fitting only and never enters the bitstream.

Coarse network. The coarse MLP maps the uniform random samples to coarse positions:

$$\hat{\mathbf{x}} = \sigma\big(\sigma^{-1}(\alpha\,\mathbf{u} + \beta) + F(\gamma(\mathbf{u}))\big), \tag{2}$$

where σ is the logistic function, γ is the sinusoidal positional encoding (Mildenhall et al., 2021; Tancik et al., 2020), and F the layers of the coarse MLP, with weights θ_{coarse}. The first term is the sample scaled by α and shifted by β, which avoids the saturating ends of σ, and the displacement is applied in logit space, which keeps the generated positions in $[0,1]^3$ by construction.

Freezing and caching. Once the distribution is fitted, the coarse MLP is frozen, and its weights are rounded to a fixed bit width at that point. The rounding happens before the coarse positions are generated, so that the fine MLP is trained on exactly the positions that the decoder will reproduce from the stored weights. A fixed set of N pseudo-random points, drawn from the seed, is then mapped through the frozen MLP once, and the resulting coarse positions are cached. They serve two purposes. Each gives a Gaussian its initial position, which the fine MLP refines through rasterization. More importantly, each gives the hash grid (Müller et al., 2022) of §3.2 a stable query coordinate,

since a hash lookup is only meaningful if the coordinate that produces it does not move. Since the samples cost no rate, N sets only the rendering cost, while the table size T sets the rate. Empirically, a lightweight MLP of about 0.1 MB is sufficient to model the distribution and generates millions of points at good quality (Figure 1).

3.2 Gaussian Parameter Prediction

With the coarse positions sampled from the learned distribution (§3.1), the fine MLP predicts the Gaussian parameters in a single forward pass.

Feature encoding. Since fine detail is hard to model with a coordinate-based network alone, even with positional encoding (Mildenhall et al., 2021; Tancik et al., 2020), the fine MLP first queries a multi-resolution hash grid (Müller et al., 2022) at the coarse position, and its layers read the resulting feature rather than the coordinate itself.

The fine MLP reads three multi-resolution hash tables, one per attribute group (position; opacity with appearance; scale with rotation), whose per-level sizes sum to 2^T entries (Appendix A.2). Since the tables hold the majority of the parameters, T governs the rate.

Attribute heads. We model each attribute (the position offset $\Delta\mathbf{x}$, opacity $\mathbf{o}$, appearance $\mathbf{f}$, scale $\mathbf{s}$ and rotation $\mathbf{q}$) with its own MLP layers, a head H_a.

$$\begin{aligned} \Delta\mathbf{x} &= H_x\big(h_x(\hat{\mathbf{x}})\big), & \mathbf{x} &= \sigma\big(\sigma^{-1}(\hat{\mathbf{x}}) + \Delta\mathbf{x}\big), \\ \mathbf{o} &= H_o\big(h_{of}(\hat{\mathbf{x}})\big), & \mathbf{f} &= H_f\big(h_{of}(\hat{\mathbf{x}})\big), \\ \mathbf{s} &= H_s\big(h_{sq}(\hat{\mathbf{x}})\big), & \mathbf{q} &= H_q\big(h_{sq}(\hat{\mathbf{x}})\big), \end{aligned} \tag{3}$$

where h_x, h_{of} and h_{sq} are the hash encodings queried at the coarse position. The tables and the heads form the fine MLP, with weights θ_{fine}. The position offset is applied in the same logit space as Eq. equation 2, so $\mathbf{x}$ stays inside the normalized frame by construction.

Decoded parameters. The appearance $\mathbf{f}$ holds the spherical harmonic coefficients up to degree three, and $\mathbf{o}$, $\mathbf{s}$ and $\mathbf{q}$ follow the standard definitions, with the layers emitting pre-activation values to which the usual sigmoid, exponential and normalization are applied.

View decoder. Inspired by Scaffold-GS (Lu et al., 2024), we also introduce an optional view-dependent decoder, and denote the variant that uses it GSRC-V. When it is enabled, the fine MLP additionally predicts a latent $\mathbf{z}$ for each Gaussian, and the appearance $\mathbf{f}$ is reduced to the base color, without higher-order spherical harmonics. A small view decoder V takes $\mathbf{z}$, the view direction $\mathbf{d}$ and the log distance δ from the camera center to the Gaussian center, and predicts residuals $(\Delta\mathbf{f}, \Delta\mathbf{o}, \Delta\mathbf{s}, \Delta\mathbf{q}) = V(\mathbf{z}, \mathbf{d}, \delta)$ that refine the color, opacity, scale and rotation.

Initialization. Before training through the renderer, the fine MLP is fitted to the same target cloud $\mathcal{P}$. The fine MLP is regressed onto the target attributes at the coarse positions, combining the density-aware Chamfer loss (Wu et al., 2021) of §3.1 on positions with a small MSE term on the remaining attributes.

3.3 Representation Compression

Entropy model and quantization parameters. We use the same entropy model for the hash grids and for the MLP weights, which is a Laplacian with a learned mean and scale, together with a learned quantization step size. The step size and the scale are parameterized in the log domain, which keeps them positive. We use per-table and per-level parameters for the grids, and per-weight-tensor ones for the MLPs. These parameters incur a small overhead, and they let the model spend precision where a given scene needs it. The coarse MLP instead uses a fixed bit width: it is rounded at freeze (§3.1) and stored without entropy coding.

Unlike most existing work, our entropy model is context-free, yet it remains competitive with the context-model codecs. The reason is that most of the redundancy is reduced by the representation itself, before the weights reach the entropy coder. First, the coarse MLP has already absorbed the global spatial structure of the scene (Figure 3a). Second, the multi-resolution hash grid captures the

remaining spatial redundancy across its levels. Under this decomposition, little correlation remains for a context model to exploit, and a static entropy model already codes the representation well.

Bitstream coding. The bitstream holds the seed, the scene transform, the coarse MLP, the entropy-model parameters, and the fine MLP (the hash tables and layers). The frame parameters are the PCA rotation, the center and the bounding box of the normalized frame (§3), which the decoder needs to map the generated positions from the unit cube back to world coordinates. Only the fine MLP is entropy coded, while the seed is a header field, the scene transform is written as floating point, the coarse MLP ships as a quantized scale-int stream, and the entropy-model parameters are stored raw. The tables, which hold almost all of the coded parameters, are written with a range asymmetric numeral systems (rANS) coder in chunks. The chunks are coded independently and in parallel, since there is no context model and no symbol depends on another, so the decode time is flat in the rate.

3.4 Training and Rate-Aware Fine-Tuning

With the coarse MLP frozen and the fine MLP initialized, GSRC is trained by minimizing the distortion D of the rendered views through the differentiable rasterizer, as in 3D-GS (Kerbl et al., 2023). In each training step, the MLPs generate the Gaussians for rendering. Since the forward pass of the MLPs is shared by all views, we optimize GSRC on multiple views in each step to amortize its cost. We do not apply densification or pruning to control the number of Gaussians, which are generated by the forward pass. After training, we further fine-tune the representation with the rate–distortion objective

$$\min_{\theta_{\text{fine}},\,\psi} \; R(\hat{\theta}_{\text{fine}};\psi) + \lambda\, D\big(\mathcal{G}_{\theta_{\text{coarse}},\,\hat{\theta}_{\text{fine}}}(s)\big), \tag{4}$$

where $\hat{\theta}_{\text{fine}}$ are the quantized fine-MLP weights and ψ the parameters of the entropy model, including the quantization step sizes (§3.3), which are written to the bitstream directly. Here R is the coded size of these weights, estimated by their entropy (Ballé et al., 2017), and λ weights the distortion against it, so that a larger λ increases quality but also the rate.

Since R and D are not differentiable under quantization, we approximate rounding with soft-rounding and Kumaraswamy noise during the fine-tune (Agustsson & Theis, 2020; Kim et al., 2024), and round hard at coding time.

4 Experiments

Setup. We evaluate on NeRF-Synthetic (Mildenhall et al., 2021), Mip-NeRF 360 (Barron et al., 2022), Tanks&Temples (T&T) (Knapitsch et al., 2017) and Deep Blending (Hedman et al., 2018) (pooled into one mean), and BungeeNeRF (Xiangli et al., 2022), and report PSNR, SSIM and LPIPS on the test views and the size in MB (10^6 bytes): every parameter at 32 bits for uncompressed models, and the coded file for compressed ones. The uncompressed references are 3D-GS (Kerbl et al., 2023), FastGS (Ren et al., 2025), 3DGS-MCMC (Kheradmand et al., 2024), EDGS (Kotovenko et al., 2026), Scaffold-GS (Lu et al., 2024) and Octree-GS (Ren et al., 2024). The compressed baselines are RDO-Gaussian (Wang et al., 2024a) and KISS-GS (Morgenstern et al., 2026), and HAC++ (Chen et al., 2025c), ContextGS (Wang et al., 2024b), CAT-3DGS (Zhan et al., 2025a) and PCGS (Chen et al., 2025a). Each baseline is either reproduced from its public code or taken from its paper, with more details in Appendix A.4. For GSRC and GSRC-V, we sweep T and N with a fixed $\lambda = 1024$ for the compressed setting (Appendix A.3), and summarize each comparison by the BD-rate (Bjøntegaard, 2001) over the two coded frontiers. In Table 1, each codec is shown at its published point nearest in size to our largest compressed model, and our rows are named by T and N; ‡ marks a cell obtained at another T or N than its row, † a result under a different protocol, and § the Octree-GS variant built on Scaffold-GS.

Reparameterization. The uncompressed rows of Table 1 measure the reparameterization itself, before any coding: GSRC is already 2.6–10× smaller than 3D-GS at equal or higher PSNR. With the same initialization, GSRC is ahead of EDGS in PSNR on every dataset at a fraction of its size, for instance 28.02 dB at 166 MB against 27.82 dB at 487 MB on Mip-NeRF 360. On the first three datasets it also exceeds 3D-GS, FastGS and Scaffold-GS in PSNR, and on Mip-NeRF 360 it is level with 3DGS-MCMC and Octree-GS. Since GSRC uses neither densification nor relocation, we attribute this quality to the prior of the network (Ulyanov et al., 2018).

Table 1: **Uncompressed and compressed results.** Quality (PSNR, SSIM, LPIPS) and size (MB) of uncompressed representations and of codecs on four datasets. Bold and underline: best and second best within each half; symbols are explained in §4.

		NeRF-Synthetic				Mip-NeRF 360				Tanks. + Deep.				BungeeNeRF			
	Method	PSNR	SSIM	LPIPS	MB	PSNR	SSIM	LPIPS	MB	PSNR	SSIM	LPIPS	MB	PSNR	SSIM	LPIPS	MB
Uncompressed	*3D-GS-based*																
	3D-GS	33.86	0.970	0.031	62.6	27.54	0.814	0.220	655	26.82	0.881	0.204	502	25.29	0.853	0.194	1655
	FastGS-Big	32.94	0.965	0.038	10.3	27.96	0.820	0.216	288	27.26	0.884	0.207	148	24.98	0.827	0.244	340
	3DGS-MCMC	**34.03**	**0.971**	0.030	69.8	28.03	**0.834**	0.190	757	27.20	**0.890**	0.191	595	26.03	**0.875**	**0.168**	1655
	EDGS	33.21	0.966	0.031	187	27.82	0.831	**0.178**	487	27.08	**0.890**	**0.168**	459	24.58	0.819	0.245	290
	Anchor-based																
	Scaffold-GS	33.51	0.967	0.034	17.7	27.72	0.811	0.228	179	27.24	0.882	0.211	68.9	**26.64**	0.867	0.196	194
	Octree-GS§	33.05	0.964	0.037	20.8	28.05	0.819	0.214	150	**27.58**	0.889	0.197	84.0	24.04	0.841	0.207	323
	GSRC (Ours)																
	T=18, N=2^{19}	33.58‡	0.969	0.033	7.1	27.46	0.806	0.239	44.9	27.16	0.877	0.222	44.9	23.64	0.760	0.324	44.9
	T=20, N=2^{21}	33.94‡	0.969	**0.029**	23.9	28.02	0.826	0.191	166	27.38	0.887	0.189	166	25.83	0.851	0.218	166
	GSRC-V (Ours)																
	T=18, N=2^{19}	33.83‡	0.968	0.032	8.8	27.54	0.807	0.236	56.1	26.84	0.868	0.231	56.1	24.05	0.774	0.320	56.1
	T=20, N=2^{21}	34.01‡	0.968	**0.029**	29.9	**28.17**	0.827	0.189	209	26.97	0.874	0.203	209	26.40	0.867	0.206	209
Compressed	*3D-GS-based*																
	RDO-Gaussian	33.12	0.967	0.035	2.3	26.87	0.796	0.248	15.4	26.43	0.866	0.229	9.5	23.37†	0.762	0.286	41.0
	KISS-GS	33.54†	0.969	0.030	3.8	27.94	0.823	0.208	13.4	27.28	**0.889**	**0.190**	13.4	–	–	–	–
	Anchor-based																
	ContextGS	33.51	0.968	0.035	1.6	27.62	0.808	0.237	13.3	27.34	0.883	0.217	9.7	26.90	0.866	0.222	14.7
	HAC++	33.75	0.969	0.033	1.9	27.80	0.810	0.234	15.4	27.33	0.882	0.216	7.3	**27.13**	0.873	0.210	16.9
	CAT-3DGS	32.46	0.962	0.046	0.8	27.77†	0.810	0.241	12.9	27.35†	0.881	0.229	5.5	26.65†	0.859	0.226	17.0
	PCGS	33.77	0.969	0.034	2.4	27.93	0.810	0.238	16.9	**27.49**	0.883	0.220	8.2	27.10	0.870	0.217	18.3
	GSRC (Ours)																
	T=17, N=2^{22}	33.05‡	0.966	0.037	0.8	27.42	0.800	0.240	3.8	27.21‡	0.877	0.226	3.7	24.86	0.798	0.271	3.8
	T=18, N=2^{22}	33.52‡	0.968	0.033	1.1	27.79	0.815	0.216	6.6	26.76‡	0.859	0.247	6.2	25.58	0.830	0.238	6.7
	T=19, N=2^{22}	33.81‡	0.969	0.031	1.8	27.98	0.823	0.201	10.9	27.31‡	0.885	0.210	8.9	26.06	0.851	0.215	11.5
	T=20, N=2^{22}	33.94‡	0.969	0.030	2.8	28.07	**0.827**	0.194	15.5	27.34‡	0.887	0.206	10.9	26.22	0.859	0.205	17.5
	GSRC-V (Ours)																
	T=17, N=2^{22}	33.52‡	0.967	0.034	0.9	27.58	0.803	0.234	4.7	26.45	0.842	0.270	4.6	25.57	0.822	0.253	4.7
	T=18, N=2^{22}	33.90‡	0.969	0.031	1.4	27.90	0.816	0.212	8.1	26.26	0.822	0.278	7.2	26.28	0.851	0.220	8.2
	T=19, N=2^{22}	34.12‡	0.969	0.029	2.2	**28.10**	0.823	0.199	12.5	27.00	0.864	0.227	10.0	26.70	0.868	0.200	12.7
	T=20, N=2^{22}	**34.20**‡	**0.970**	**0.028**	3.4	28.09	0.826	**0.193**	16.7	25.73	0.848	0.253	12.4	26.81	**0.874**	**0.192**	18.0

Compression. With the rate-aware fine-tuning and entropy coding, and without the context model that most recent codecs rely on, GSRC is far ahead of RDO-Gaussian on every dataset and outperforms the context-model codecs in most cases, especially in SSIM and LPIPS (Table 1, Figure 5). At equal PSNR, SSIM and LPIPS, it needs 49%, 59% and 72% fewer bytes than HAC++ on Mip-NeRF 360. On T&T and Deep Blending, it saves 6% and 13% at equal SSIM and LPIPS but needs 25% more bytes at equal PSNR. KISS-GS, the strongest codec without a context model, matches GSRC on Mip-NeRF 360 at a larger size and leads on T&T and Deep Blending in SSIM and LPIPS. BungeeNeRF, whose views range from city to street scale, is the exception in PSNR: GSRC trails the context-model codecs there, and GSRC-V narrows the gap to HAC++ to 0.26 dB at equal size and needs 43% fewer bytes at equal LPIPS. Figure 6 shows one view at a matched size.

Number of Gaussians and table size. On Mip-NeRF 360 at $T = 20$, GSRC raises PSNR from 27.18 to 28.07 dB when N increases from 2^{18} to 2^{22}, while the coded size grows only from 13.1 to 15.5 MB and the uncompressed size does not change. At $N = 2^{21}$, increasing T from 17 to 22 moves the coded size from 3.8 to 17.0 MB. N thus sets the rendering cost and T decides the rate. Figure 13 in the appendix shows the full $N \times T$ grid on every dataset, before and after compression.

Decoding speed. On one NVIDIA GH200 GPU, GSRC decodes a Mip-NeRF 360 bitstream in 0.85 s and encodes it in 0.65 s, 24× faster decoding than HAC++ and 18–79× faster than the other context-model codecs (Table 2), since its entropy models are static and the tables are decoded in independent chunks, and that the decoding time barely changes with the rate. The decoded Gaussians are then rendered by the unmodified 3D-GS rasterizer.

Table 2: Training, encoding and decoding time on Mip-NeRF 360, on one NVIDIA GH200 GPU.

Method	Train (min)↓	Enc. (s)↓	Dec. (s)↓
ContextGS	67	17.40	15.50
HAC++	**54**	11.40	20.80
CAT-3DGS	76	76.70	67.40
PCGS	73	30.60	45.20
GSRC	124	**0.65**	**0.85**
GSRC-V	138	0.72	1.06

Ablations. Appendix A.1 ablates the design on Mip-NeRF 360. The rate-aware fine-tuning is the main compression lever: quantizing the trained model without it costs 0.57 dB at 73% more bytes.

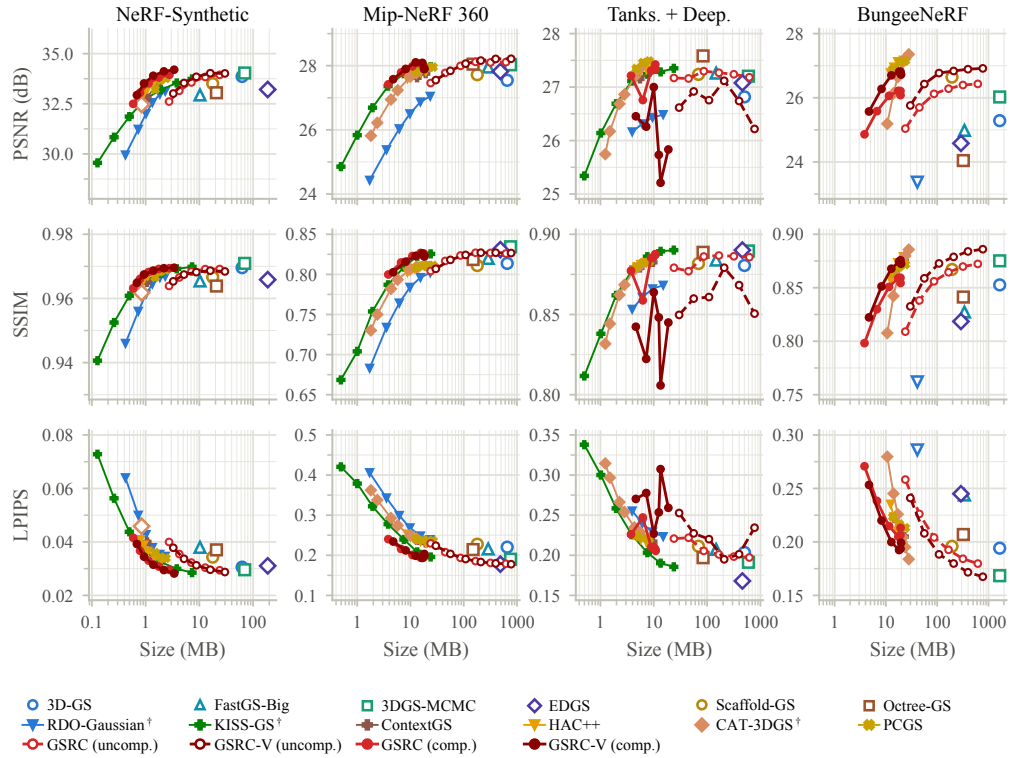


Figure 5: **Rate–distortion on four datasets.** PSNR, SSIM and LPIPS against size (MB) for GSRC, GSRC-V and the uncompressed and compressed baselines.



Figure 6: Visual comparison at a matched size on *SCENE* (Mip-NeRF 360).

EDGS is the best of the initializations we tried, and the grouped tables are within 0.05 dB of both per-attribute tables and a single shared table.

5 Conclusion

This paper presents GSRC, which reparameterizes the Gaussians of a scene as the output of neural networks, so that no Gaussian primitives, positions or anchors need to be stored and only the network parameters are coded. The design is motivated by a key observation: a scene is not uniquely identified by a particular set of Gaussians, but rather by their distribution. A coarse MLP that generates the point set from a shared seed therefore replaces the geometry bitstream, while the remaining attributes are predicted from hash tables and coded using a static entropy model, without requiring a context model. Before any coding, the reparameterized model is already 4× smaller than 3D-GS on Mip-NeRF 360 at higher PSNR. After coding, GSRC outperforms the context-model codecs in most cases, particularly in SSIM and LPIPS, while decoding 24× faster than HAC++. The limitations fo this work are its encoding time, which is more than twice the training time of HAC++ (Table 2). Its use of a fixed number of Gaussians rather than a scene-adaptive point budget, and its relatively low PSNR on scenes with large scale variation, such as BungeeNeRF, a gap that is narrowed by GSRC-V. Adaptive point budgets and priors shared across scenes provide natural directions for future work.

REFERENCES

Eirikur Agustsson and Lucas Theis. Universally quantized neural compression. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 12367–12376, 2020.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end optimized image compression. In *International Conference on Learning Representations (ICLR)*, 2017.

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5470–5479, 2022.

Gisle Bjøntegaard. Calculation of average PSNR differences between RD-curves. Technical Report VCEG-M33, ITU-T SG16/Q6 (VCEG), 2001.

Hao Chen, Bo He, Hanyu Wang, Yixuan Ren, Ser-Nam Lim, and Abhinav Shrivastava. NeRV: Neural representations for videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

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, 2024.

Yihang Chen, Mengyao Li, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Pcgs: Progressive compression of 3d gaussian splatting. *arXiv preprint arXiv:2503.08511*, 2025a.

Yihang Chen, Qianyi Wu, Mengyao Li, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Fast feed-forward 3d gaussian splatting compression. In *International Conference on Learning Representations (ICLR)*, 2025b.

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

Francesco Di Sario, Riccardo Renzulli, Marco Grangetto, Akihiro Sugimoto, and Enzo Tartaglione. Gode: Gaussians on demand for progressive level of detail and scalable compression. *arXiv preprint arXiv:2501.13558*, 2025.

Emilien Dupont, Adam Golinski, Milad Alizadeh, Yee Whye Teh, and Arnaud Doucet. COIN: COmpression with implicit neural representations. *arXiv preprint arXiv:2103.03123*, 2021.

Emilien Dupont, Hrushikesh Loya, Milad Alizadeh, Adam Goliński, Yee Whye Teh, and Arnaud Doucet. COIN++: Neural compression across modalities. *arXiv preprint arXiv:2201.12904*, 2022.

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, 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.

D. Graziosi, O. Nakagami, S. Kuma, A. Zaghetto, T. Suzuki, and A. Tabatabai. An overview of ongoing point cloud compression standardization activities: video-based (V-PCC) and geometry-based (G-PCC). *APSIPA Transactions on Signal and Information Processing*, 9(1), 2020.

Alex Hanson, Allen Tu, Geng Lin, Vasu Singla, Matthias Zwicker, and Tom Goldstein. Speedy-splat: Fast 3d gaussian splatting with sparse pixels and sparse primitives. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21537–21546, 2025a.

Alex Hanson, Allen Tu, Vasu Singla, Mayuka Jayawardhana, Matthias Zwicker, and Tom Goldstein. PUP 3d-gs: Principled uncertainty pruning for 3d gaussian splatting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5949–5958, 2025b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

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.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

Stephan Hoyer, Jascha Sohl-Dickstein, and Sam Greydanus. Neural reparameterization improves structural optimization. *arXiv preprint arXiv:1909.04240*, 2019.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 80965–80986, 2024.

Hyunjik Kim, Matthias Bauer, Lucas Theis, Jonathan Richard Schwarz, and Emilien Dupont. C3: High-performance and low-complexity neural compression from a single image or video. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

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.

Dmytro Kotovenko, Olga Grebenkova, and Björn Ommer. EDGS: Eliminating densification for efficient convergence of 3DGS. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 41065–41076, 2026.

Ho Man Kwan, Ge Gao, Fan Zhang, Andrew Gower, and David Bull. HiNeRV: Video compression with hierarchical encoding-based neural representation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Ho Man Kwan, Ge Gao, Fan Zhang, Andrew Gower, and David Bull. NVRC: Neural video representation compression. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Ho Man Kwan, Tianhao Peng, Fan Zhang, Mike Nilsson, Andrew Gower, and David Bull. Enhanced neural video representation compression across extreme complexity and quality scales. In *European Conference on Computer Vision (ECCV)*, 2026.

Théo Ladune, Pierrick Philippe, Félix Henry, Gordon Clare, and Thomas Leguay. COOL-CHIC: Coordinate-based low complexity hierarchical image codec. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

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.

Joo Chan Lee, Jong Hwan Ko, and Eunbyung Park. Optimized minimal 3d gaussian splatting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

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

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, 2024b.

Xiangrui Liu, Xinju Wu, Shiqi Wang, Zhu Li, and Sam Kwong. Compgs++: Compressed gaussian splatting for static and dynamic scene representation. *arXiv preprint arXiv:2504.13022*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

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, 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.

Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. Compact 3d scene representation via self-organizing gaussian grids. In *European Conference on Computer Vision*, pp. 18–34. Springer, 2024.

Wieland Morgenstern, Friedrich Elias Branschke, Florian Fleischmann, Adrian Szatmari, Paul Schlack, Florian Barthel, Peter Eisert, and Anna Hilsmann. KISS-GS: 3d gaussian splatting compression kept simple. In *European Conference on Computer Vision*, pp. 677–694. Springer, 2026.

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):102:1–102:15, 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.

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.

Shiwei Ren, Tianci Wen, Yongchun Fang, and Biao Lu. FastGS: Training 3d gaussian splatting in 100 seconds. *arXiv preprint arXiv:2511.04283*, 2025.

Sebastian Schwarz, Marius Preda, Vittorio Baroncini, Madhukar Budagavi, Pablo Cesar, Philip A. Chou, Robert A. Cohen, Maja Krivokuca, Sebastien Lasserre, Zhu Li, Joan Llach, Khaled Mammou, Rufael Mekuria, Ohji Nakagami, Ernestasia Siahaan, Ali Tabatabai, Alexis M. Tourapis, and Vladyslav Zakharchenko. Emerging MPEG standards for point cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):133–148, 2019.

Seungjoo Shin, Jaesik Park, and Sunghyun Cho. Locality-aware gaussian compression for fast and high-quality rendering. In *International Conference on Learning Representations (ICLR)*, 2025.

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 7537–7547, 2020.

Zhenyu Tang, Chaoran Feng, Xinhua Cheng, Wangbo Yu, Junwu Zhang, Yuan Liu, Xiaoxiao Long, Wenping Wang, and Li Yuan. Neuralgs: Bridging neural fields and 3d gaussian splatting for compact 3d representations. *arXiv preprint arXiv:2503.23162*, 2025.

Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9446–9454, 2018.

Henan Wang, Hanxin Zhu, Tianyu He, Runsen Feng, Jiajun Deng, Jiang Bian, and Zhibo Chen. End-to-end rate-distortion optimized 3d gaussian representation. In *European Conference on Computer Vision*, pp. 76–92. Springer, 2024a.

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, 2024b.

Minye Wu and Tinne Tuytelaars. Implicit gaussian splatting with efficient multi-level tri-plane representation. *arXiv preprint arXiv:2408.10041*, 2024.

Tong Wu, Liang Pan, Junzhe Zhang, Tai Wang, Ziwei Liu, and Dahua Lin. Density-aware Chamfer distance as a comprehensive metric for point cloud completion. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 29088–29100, 2021.

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

Shuzhao Xie, Weixiang Zhang, Chen Tang, Yunpeng Bai, Rongwei Lu, Shijia Ge, and Zhi Wang. Mesongs: Post-training compression of 3d gaussians via efficient attribute transformation. In *European Conference on Computer Vision*, pp. 434–452. Springer, 2024.

Qi Yang, Le Yang, Geert Van Der Auwera, and Zhu Li. Hybridgs: High-efficiency gaussian splatting data compression using dual-channel sparse representation and point cloud encoder. *arXiv preprint arXiv:2505.01938*, 2025.

Yu-Ting Zhan, Cheng-Yuan Ho, Hebi Yang, Yi-Hsin Chen, Jui Chiu Chiang, Yu-Lun Liu, and Wen-Hsiao Peng. Cat-3dgs: A context-adaptive triplane approach to rate-distortion-optimized 3dgs compression. *arXiv preprint arXiv:2503.00357*, 2025a.

Yu-Ting Zhan, He-bi Yang, Cheng-Yuan Ho, Jui-Chiu Chiang, and Wen-Hsiao Peng. Cat-3dgs pro: A new benchmark for efficient 3dgs compression. *arXiv preprint arXiv:2503.12862*, 2025b.

Yangming Zhang, Wenqi Jia, Wei Niu, and Miao Yin. Gaussianspa: An "optimizing-sparsifying" simplification framework for compact and high-quality 3d gaussian splatting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26673–26682, 2025.

Statement on the Use of Generative AI

Large language models were used as an assistant for programming, for managing, scheduling and monitoring the experiments, and for editing the text for clarity and grammar. The research method was designed by the authors, and the core code was fully validated by the authors. Every number in the paper was generated from the recorded results and checked by the authors against the experiment records, and every reference was checked against its source. The authors take full responsibility for the content.

Reproducibility Statement

The architecture, the training and coding pipeline and the evaluation protocol are specified in §3, §4 and the Appendix. The implementation, the configurations of every reported model, and the results will be released.

A Appendix

A.1 Ablation Study

Table 3: **Ablations on Mip-NeRF 360.** Each row changes one component of GSRC ($N = 2^{20}$) and reports the average change from the default model over the nine scenes and $T = 18$–21. The network and the initialization are compared before compression (default 27.71–27.99 dB at 45–318 MB), and the compression settings after rate-aware fine-tuning and coding (default 27.69–27.91 dB at 6.7–17.6 MB). The size change is the average ratio to the size of the default model, taken as a geometric mean.

Ablation	ΔPSNR (dB)	ΔSSIM	ΔLPIPS	ΔSize (%)
Network (uncompressed)				
Tables (default: three grouped tables)				
Five per-attribute tables	−0.05	−0.003	+0.004	−9.5
One shared table	−0.05	−0.001	−0.001	−4.4
Architecture (default: layer normalization, residual blocks, GELU)				
No layer normalization	−0.16	−0.003	+0.004	±0.0
No residual blocks	−0.11	−0.011	+0.015	−0.4
ReLU instead of GELU	−0.04	−0.001	+0.001	±0.0
MLP width (default: 128)				
Width 64	−0.05	±0.000	±0.000	−0.6
Width 96	−0.02	±0.000	−0.001	−0.4
Width 192	+0.03	±0.000	±0.000	+1.0
Appearance (default: spherical harmonics of degree 3)				
Degree 0	−0.85	−0.010	+0.007	±0.0
Initialization (uncompressed)				
Target cloud (default: untrained EDGS)				
SfM points	−0.49	−0.026	+0.039	±0.0
Trained 3D-GS	−0.13	−0.018	+0.041	±0.0
Trained FastGS	−0.04	−0.009	+0.023	±0.0
Compression (compressed)				
Coarse MLP (default: 10-bit)				
8-bit	−0.03	−0.001	+0.001	−0.7
32-bit floating point	−0.01	±0.000	±0.000	+1.2
Quantization (default: rate-aware fine-tuning)				
No fine-tuning	−0.57	−0.020	+0.002	+52.8
Fixed 8-bit, no fine-tuning	−0.86	−0.034	+0.007	+125.0
Entropy model (default: per table level and per weight tensor)				
One for all tables	+0.04	+0.001	−0.002	+24.7
One for all MLP layers	±0.00	−0.001	+0.003	−2.0
Training length (default: 30K training and 15K fine-tuning iterations)				
Half the training (15K)	−0.26	−0.007	+0.010	+18.3
Half the fine-tuning (7.5K)	−0.03	−0.001	+0.001	+5.2

Table 3 provides the results of the ablation study on the Mip-NeRF 360 dataset. We use GSRC with $N = 2^{20}$ as the default model, replace one of its settings at a time for each variant, and average the results over $T = 18$ to 21. We conduct the experiments in three groups. Overall, the default is the best, or within noise of the best, on every axis, where the spread between seeds of a nine-scene mean is 0.06–0.14 dB.

Network architecture. In this group, we vary the network and evaluate the uncompressed models. The grouped tables perform on par with five per-attribute tables and with a single shared table, while requiring fewer table lookups than the former. Layer normalization (Ba et al., 2016) and the residual blocks (He et al., 2016) contribute 0.16 and 0.11 dB, whereas GSRC is robust to the size of its decoder, as the width of the MLP layers has little effect between 64 and 192. The spherical harmonic degree has the largest impact. Reducing it to zero costs 0.85 dB, with the largest drops on the indoor scenes.

Initialization. In this group, we replace the target cloud of the fitting and evaluate the uncompressed models. SfM points lose 0.49 dB, and trained 3D-GS (Kerbl et al., 2023) so we use the EDGS (Kotovenko et al., 2026) initialization by default.

Compression. In this group, we vary the quantization, the entropy model and the training length, and evaluate the models after rate-aware fine-tuning and coding. The rate-aware fine-tuning is the main compression level. Without it, the quantized model loses 0.57 dB and is 53% larger, and a fixed 8-bit quantization loses 0.86 dB at more than twice the size. One entropy model for all tables makes the bitstream 25% larger than per-table and per-level models, while one model for all MLP layers makes little difference, as these layers are a small part of the bitstream. The coarse MLP is

insensitive to its precision between 8 and 32 bits, so 10 bits cost nothing. Halving the training costs 0.26 dB, while halving the fine-tuning costs only 0.03 dB at 5% more bits, which offers a way to reduce the encoding time.

A.2 Network Architecture

Figure 7 shows the full network.

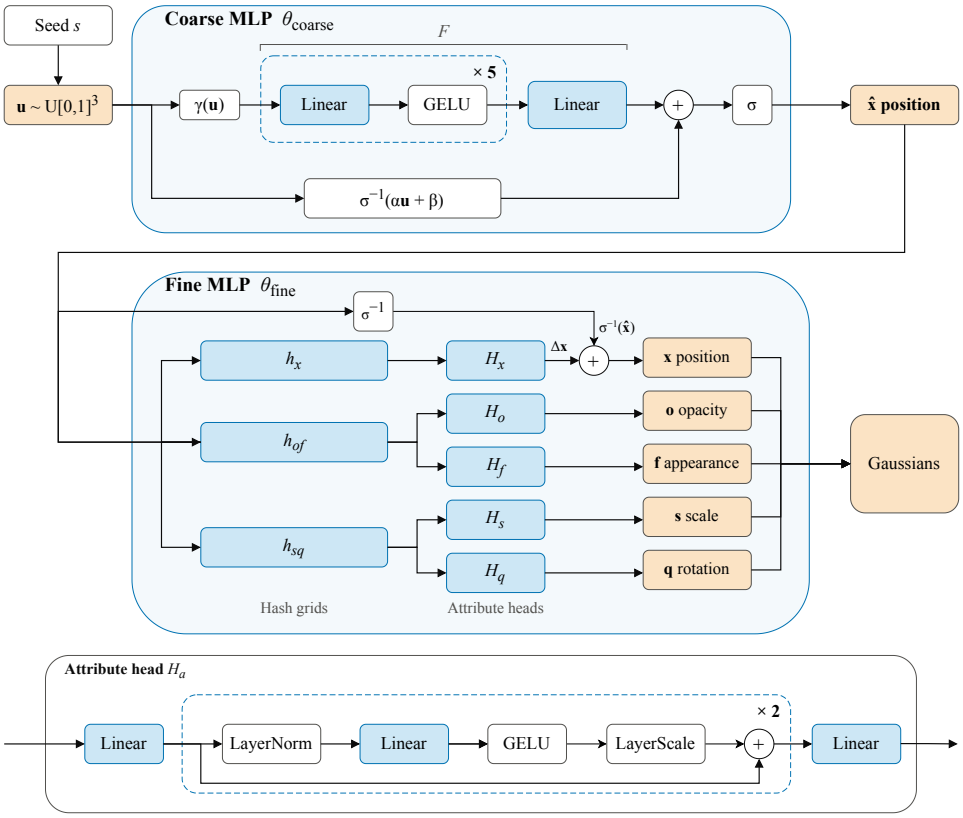


Figure 7: **Network architecture of GSRC.** The coarse MLP maps the seeded samples $\mathbf{u}$ to the coarse positions $\hat{\mathbf{x}}$. The fine MLP queries three hash tables at $\hat{\mathbf{x}}$ and decodes each attribute with its own MLP layers (bottom); the position is a residual on $\sigma^{-1}(\hat{\mathbf{x}})$. Blue: learned and stored; orange: tensors; white: fixed operations.

Scene frame. The scene is rotated into the principal axes of its SfM point cloud and centered on its centroid (Zhan et al., 2025a). Along each axis, the interval between the 0.2 and 0.8 quantiles of the points is mapped to $[-1, 1]$, and coordinates outside the resulting box are contracted as in mip-NeRF 360 (Barron et al., 2022), $\mathbf{y} \mapsto (2t - t^2/\|\mathbf{y}\|)\,\mathbf{y}/\|\mathbf{y}\|$ for $\|\mathbf{y}\| > t$, with $t = 1.5$. The contracted coordinates are finally rescaled to the unit cube, in which the samples, the coarse positions and all hash queries live.

Coarse MLP. The N samples $\mathbf{u}$ are drawn uniformly in $[0, 1]^3$ by a pseudo-random generator with a fixed seed. The positional encoding γ concatenates $\mathbf{u}$ with the sine and cosine of $2^k \pi \mathbf{u}$ for $k = 0, \ldots, 15$ (99 dimensions). F has five hidden layers of width 128 with GELU activations (Hendrycks & Gimpel, 2016) and a linear output of dimension 3, for 79,235 parameters in total. In Eq. equation 2, $\alpha = 0.5$ and $\beta = 0.25$. After fitting, its weights are rounded to 10-bit integers with one scale per tensor and stored without entropy coding.

Hash tables. Each table is a multiresolution hash encoding (Müller et al., 2022) with 12 levels of 4 features, a base resolution of 16 and a growth factor of 2 between levels. The number of entries per level is capped at 2^{T-1} for the position table and at 2^{T-2} for the opacity–appearance and the scale–rotation tables, so that the caps of the three tables sum to 2^T; the coarsest levels, which have

fewer cells than the cap, are stored densely. At $T = 20$, the three tables hold 1.03×10^7 entries, or 4.1×10^7 parameters. Grouping the attributes in this way reduces the number of table lookups and parameters, with little effect on quality.

MLP layers. The layers of each attribute read the 48-dimensional feature of its group's table. A linear layer projects it to width 128, followed by two pre-norm residual blocks $\mathbf{h} \leftarrow \mathbf{h} + \boldsymbol{\alpha} \odot \mathrm{GELU}(\mathrm{Linear}(\,\mathrm{LN}(\mathbf{h})))$, where LN is layer normalization (Ba et al., 2016), GELU is the Gaussian Error Linear Unit (Hendrycks & Gimpel, 2016), and $\boldsymbol{\alpha}$ a learned per-channel scale initialized to 0.1, and a linear output layer of dimension 3 for $\Delta\mathbf{x}$, 1 for $\mathbf{o}$, 48 for $\mathbf{f}$ (spherical harmonics of degree three), 3 for $\mathbf{s}$ and 4 for $\mathbf{q}$. The layers of the five attributes have 207,931 parameters in total. Opacity, scale and rotation are activated by a sigmoid, an exponential and a normalization (Kerbl et al., 2023). The log-scale is bounded softly by $s \mapsto b - \mathrm{softplus}(\beta(b - s))/\beta$, with $b = 0.69$ and $\beta = 6/b$ in world units, and with the same function at $b = 0$ and $\beta = 6$ on the scale measured in the contracted frame.

GSRC-V. GSRC-V adds a fourth table of the same form with 2^{T-2} entries per level and MLP layers that predict the latent $\mathbf{z} \in \mathbb{R}^{24}$, and its appearance layers predict only the base color (dimension 3). The view decoder V maps the 28-dimensional input $[\mathbf{z}, \mathbf{d}, \delta]$ through one hidden layer of width 64 with a GELU activation to 11 outputs, the pre-activation residuals of opacity (1), color (3), scale (3) and rotation (4), for 2,571 parameters. Here δ is the log distance from the camera center to the Gaussian, minus its median over the training cameras and the target points. Each residual is bounded as $b\tanh(\Delta/b)$, with $b = 2, 1, 1, 0.5$ for opacity, color, scale and rotation, and the last layer of V is initialized to zero, so that training starts from the view-independent model. At test time, δ is clamped per Gaussian to the range observed over the training cameras.

A.3 Training Details

Initialization. On every dataset, the target cloud $\mathcal{P}$ is the EDGS (Kotovenko et al., 2026) initialization, obtained from the training views before any 3D-GS optimization. The coarse MLP is fitted for 50K iterations with the density-aware Chamfer distance (Wu et al., 2021), drawing $65{,}536$ fresh uniform samples and $65{,}536$ target points at every iteration. The fine MLP is then fitted for 10K iterations, with the same set loss on the positions and a mean squared error of weight 0.01 on the attributes of the nearest target points. Both fits use AdamW (Loshchilov & Hutter, 2019). with a learning rate of 10^{-3}, a weight decay of 10^{-4}, a warm-up over the first 10% of the iterations, and step decays by 0.4 and 0.2 at 50% and 75%.

Training. GSRC is trained for 30K iterations on 4 views per iteration (2 on BungeeNeRF), with the 3D-GS loss $0.8\,\mathcal{L}_1 + 0.2\,\mathcal{L}_{\text{SSIM}}$ (Kerbl et al., 2023), where $\mathcal{L}_{\text{SSIM}} = 1 - \mathrm{SSIM}$. All parameters of the fine MLP are optimized by AdamW with $\epsilon = 10^{-15}$, a linear warm-up to a learning rate of 10^{-3} over 7,500 iterations and a cosine decay to 10^{-5}. The hash tables use five times this learning rate and no weight decay; the MLP layers use a weight decay of 10^{-4} and learning-rate multipliers of 1 for opacity, appearance and the latent, 0.5 for scale and rotation, and 0.2 for the position offset, further scaled by the scene extent as in 3D-GS. The spherical harmonic degree of GSRC is raised by one every 1,000 iterations (7,500 on BungeeNeRF, where the higher-degree coefficients are also regularized). We use $N = 2^{18}$ to 2^{22} and $T = 17$ to 22 (NeRF-Synthetic: $N = 2^{17}$ to 2^{21}, $T = 13$ to 17; BungeeNeRF: $N \geq 2^{19}$). All runs use a single NVIDIA GH200 GPU for training with mixed precision (FP16).

Rate-aware fine-tuning. Starting from the trained weights, with a new optimizer, we fine-tune all parameters of the fine MLP and the entropy models for 15K iterations with $R + \lambda D$ and $\lambda = 1024$, where R is in megabytes; the operating points are obtained by varying T and N rather than λ. The learning rate follows a 500-iteration warm-up and a cosine decay from 10^{-3} to 10^{-5}, ten times larger for the entropy-model parameters. Following the alternating rate and distortion steps of NVRC (Kwan et al., 2024), the rate term is evaluated every fourth iteration at four times its weight, while the distortion is evaluated at every iteration. The total rate gradient is therefore unchanged, and the rate is evaluated less often. Quantization uses soft rounding with Kumaraswamy noise (Agustsson & Theis, 2020; Kim et al., 2024), whose temperature is annealed linearly from 0.5 to 0.1 for the tables and to 0.3 for the MLP layers, and whose noise strength from 2 to 1. The step sizes are initialized to 0.1 times the standard deviation of each table and to 0.005 for the MLP layers. At coding time, the parameters are rounded. The tables are coded with static rANS in independent chunks of 2^{20}

Table 4: **Tanks&Temples and Deep Blending separately.** The two datasets that Table 1 pools, with the same rows and symbols. #G: Gaussians in millions, equal to N for ours. Compressed GSRC uses $N = 2^{19}$ on Tanks&Temples (‡). Bold and underline: best and second best within each half.

		Tanks&Temples					Deep Blending				
	Method	PSNR↑	SSIM↑	LPIPS↓	MB↓	#G	PSNR↑	SSIM↑	LPIPS↓	MB↓	#G
Uncompressed	*3D-GS-based*										
	3D-GS	23.85	0.854	0.169	390	–	29.78	0.908	0.238	614	–
	FastGS-Big	24.36	0.857	0.174	134	–	30.16	**0.911**	0.240	161	–
	3DGS-MCMC	24.46	<u>0.870</u>	0.148	459	–	29.95	<u>0.910</u>	0.235	732	–
	EDGS	24.14	**0.873**	**0.120**	436	–	30.01	0.907	**0.215**	482	–
	Anchor-based										
	Scaffold-GS	24.05	0.853	0.172	79.7	–	<u>30.43</u>	<u>0.910</u>	0.250	58.2	–
	Octree-GS§	<u>24.67</u>	0.866	0.153	92.9	–	**30.49**	**0.911**	0.241	75.2	–
	GSRC (Ours)										
	T=18, N=2^{19}	24.35	0.848	0.187	44.9	0.52	29.98	0.906	0.256	44.9	0.52
	T=20, N=2^{21}	**24.72**	0.866	<u>0.146</u>	166	2.10	30.04	0.909	<u>0.231</u>	166	2.10
	GSRC-V (Ours)										
	T=18, N=2^{19}	23.77	0.831	0.208	56.1	0.52	29.91	0.905	0.254	56.1	0.52
	T=20, N=2^{21}	24.09	0.840	0.175	209	2.10	29.84	0.908	<u>0.231</u>	209	2.10
Compressed	*3D-GS-based*										
	RDO-Gaussian	23.34	0.835	0.195	12.0	–	29.48	0.899	0.265	6.7	–
	KISS-GS	**24.86**	**0.874**	**0.134**	13.3	–	29.76	0.904	0.255	7.7	–
	Anchor-based										
	ContextGS	24.29	0.855	0.175	12.4	–	**30.39**	0.909	0.259	6.9	–
	HAC++	24.31	0.854	0.178	9.0	–	30.34	0.911	0.254	5.5	–
	CAT-3DGS	24.41†	0.853	0.189	7.3	–	30.29†	0.909	0.269	3.7	–
	PCGS	24.59	0.856	0.181	10.3	–	<u>30.38</u>	0.910	0.259	6.1	–
	GSRC (Ours)										
	T=17, N=2^{22}	24.41‡	0.849	0.189	3.9	0.52	30.04	0.909	0.250	3.5	4.19
	T=18, N=2^{22}	23.54‡	0.811	0.236	7.0	0.52	30.21	0.911	0.241	5.7	4.19
	T=19, N=2^{22}	24.57‡	0.862	0.166	11.2	0.52	30.16	<u>0.912</u>	<u>0.237</u>	7.5	4.19
	T=20, N=2^{22}	<u>24.66</u>‡	<u>0.867</u>	<u>0.157</u>	14.6	0.52	30.28	**0.913**	<u>0.237</u>	7.7	4.19
	GSRC-V (Ours)										
	T=17, N=2^{22}	22.95	0.776	0.290	5.0	4.19	29.96	0.909	0.251	4.2	4.19
	T=18, N=2^{22}	22.28	0.733	0.315	8.3	4.19	30.24	<u>0.912</u>	0.240	6.1	4.19
	T=19, N=2^{22}	23.77	0.814	0.218	12.5	4.19	30.23	**0.913**	**0.236**	7.5	4.19
	T=20, N=2^{22}	23.29	0.806	0.231	16.4	4.19	28.18	0.890	0.276	8.5	4.19

symbols, and the MLP layers as one stream. Table entries that no coarse position reads are skipped: the decoder reproduces the coarse positions from the seed, so both sides know which entries are used.

A.4 Evaluation Setup

During evaluation, images wider than 1,600 pixels are downsampled to a width of 1,600 pixels, as in 3D-GS (Kerbl et al., 2023), and NeRF-Synthetic is rendered on a black background. Tanks&Temples and Deep Blending are reported as one mean over their four scenes in the main text, and separately in Appendix A.5. 3D-GS, FastGS ren2025fastgs, 3DGS-MCMC (Kheradmand et al., 2024) and EDGS (Kotovenko et al., 2026) are reproduced with their public code under the same protocol, Scaffold-GS (Lu et al., 2024), Octree-GS (Ren et al., 2024) and all compressed baselines are the values published in their papers, except RDO-Gaussian (Wang et al., 2024a) on BungeeNeRF (Xiangli et al., 2022), which is taken as reported by HAC++ (Chen et al., 2025c). Published results obtained at another image resolution are marked with †. For timing, encoding is the coding pass on the trained model, decoding is time for the bitstream to the reconstructed Gaussians.

A.5 Rate–distortion results per dataset

Figures 8–12 repeat Figure 5 for each dataset on its own, Tanks&Temples and Deep Blending apart, and Table 4 gives the two datasets that Table 1 pools.

Figure 13 shows every model we trained: each table size T at each anchor count N, before and after compression.

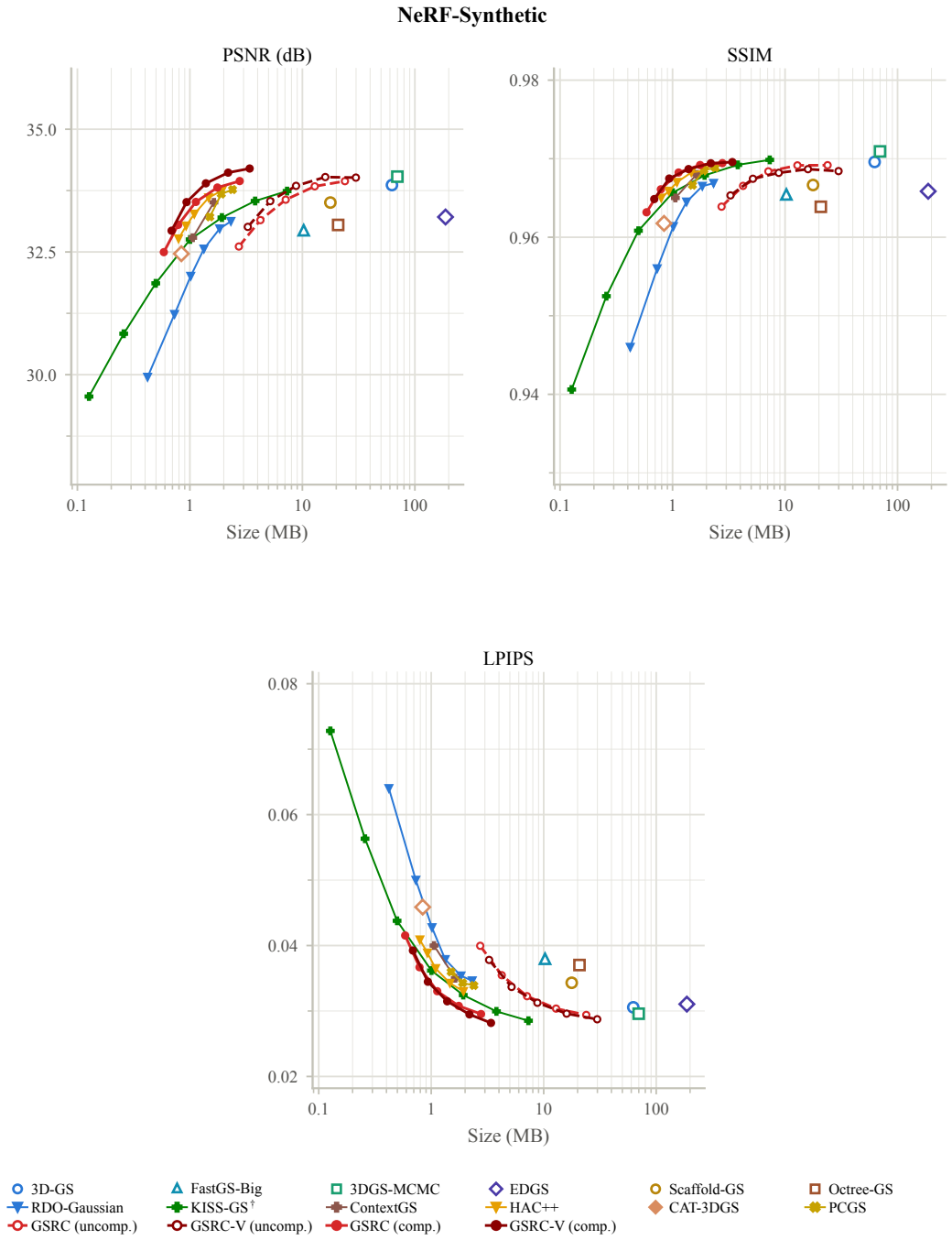


Figure 8: **Rate–distortion on NeRF-Synthetic.** PSNR, SSIM and LPIPS against size (MB $= 10^6$ bytes, log scale). Solid: compressed. Dashed or hollow: uncompressed. GSRC (red) and GSRC-V (dark red) sweep the table size $T = 13$–17 at $\lambda = 1024$ and $N = 2^{21}$. †: protocol differs from ours.

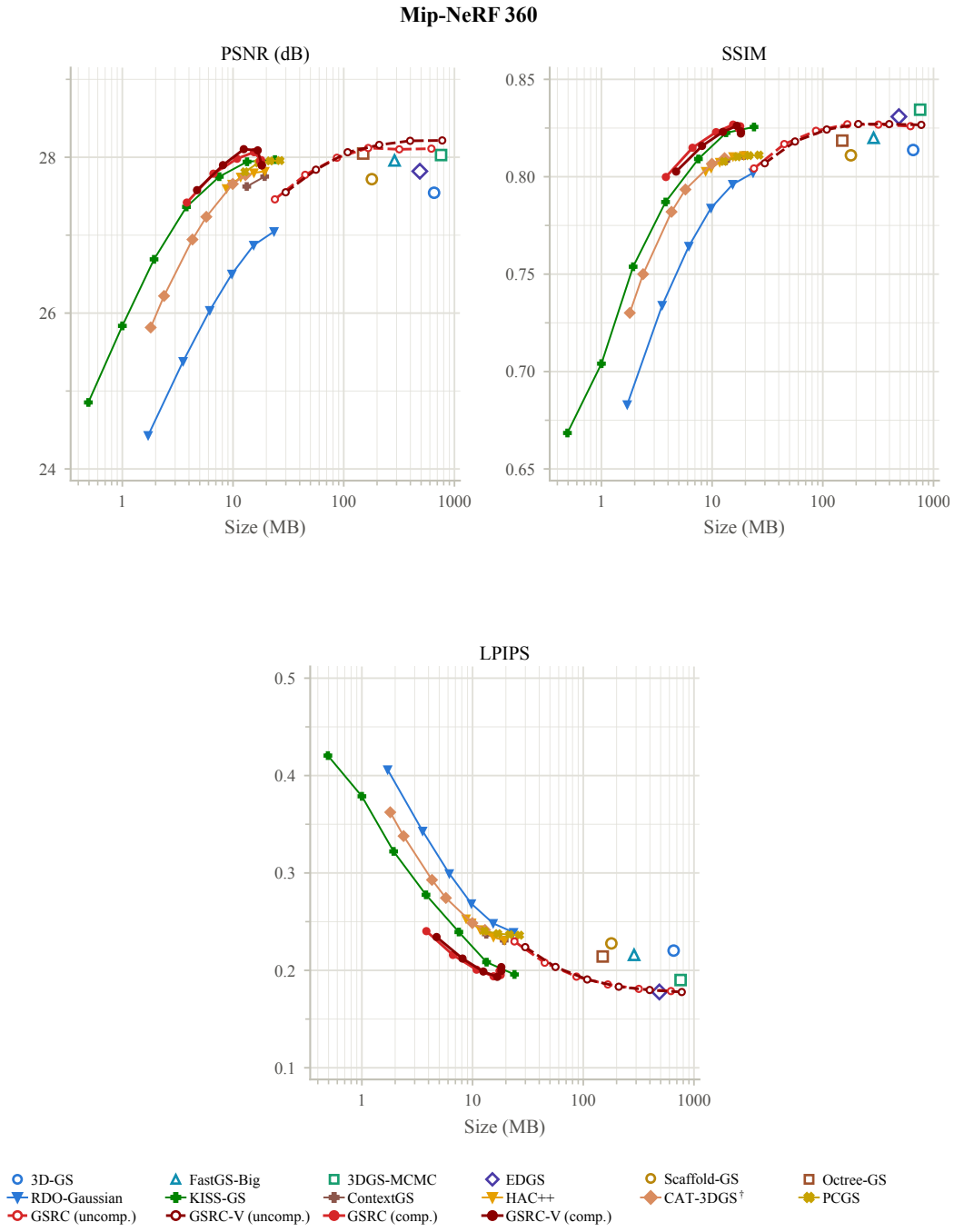


Figure 9: **Rate–distortion on Mip-NeRF 360.**

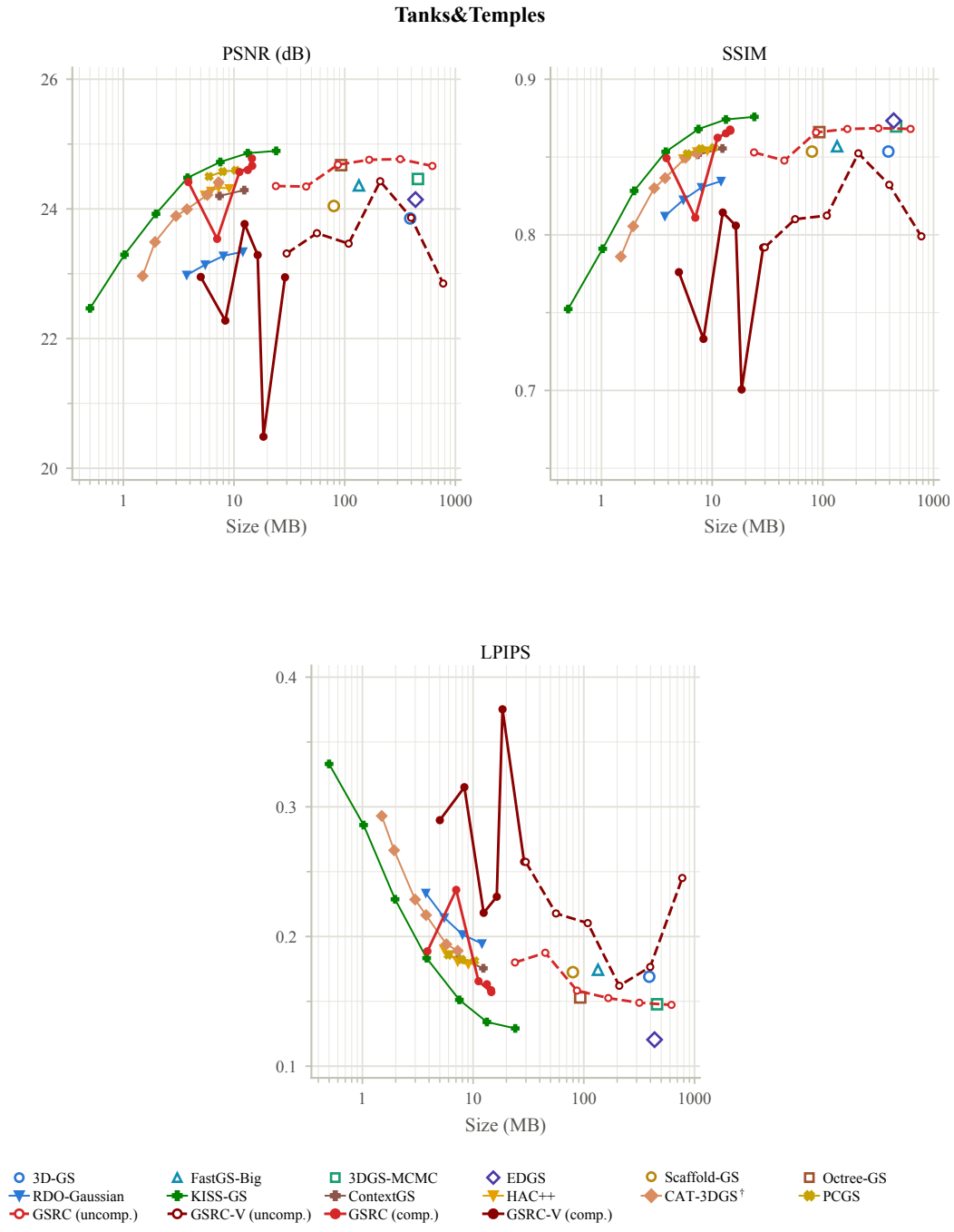


Figure 10: **Rate–distortion on Tanks&Temples.**

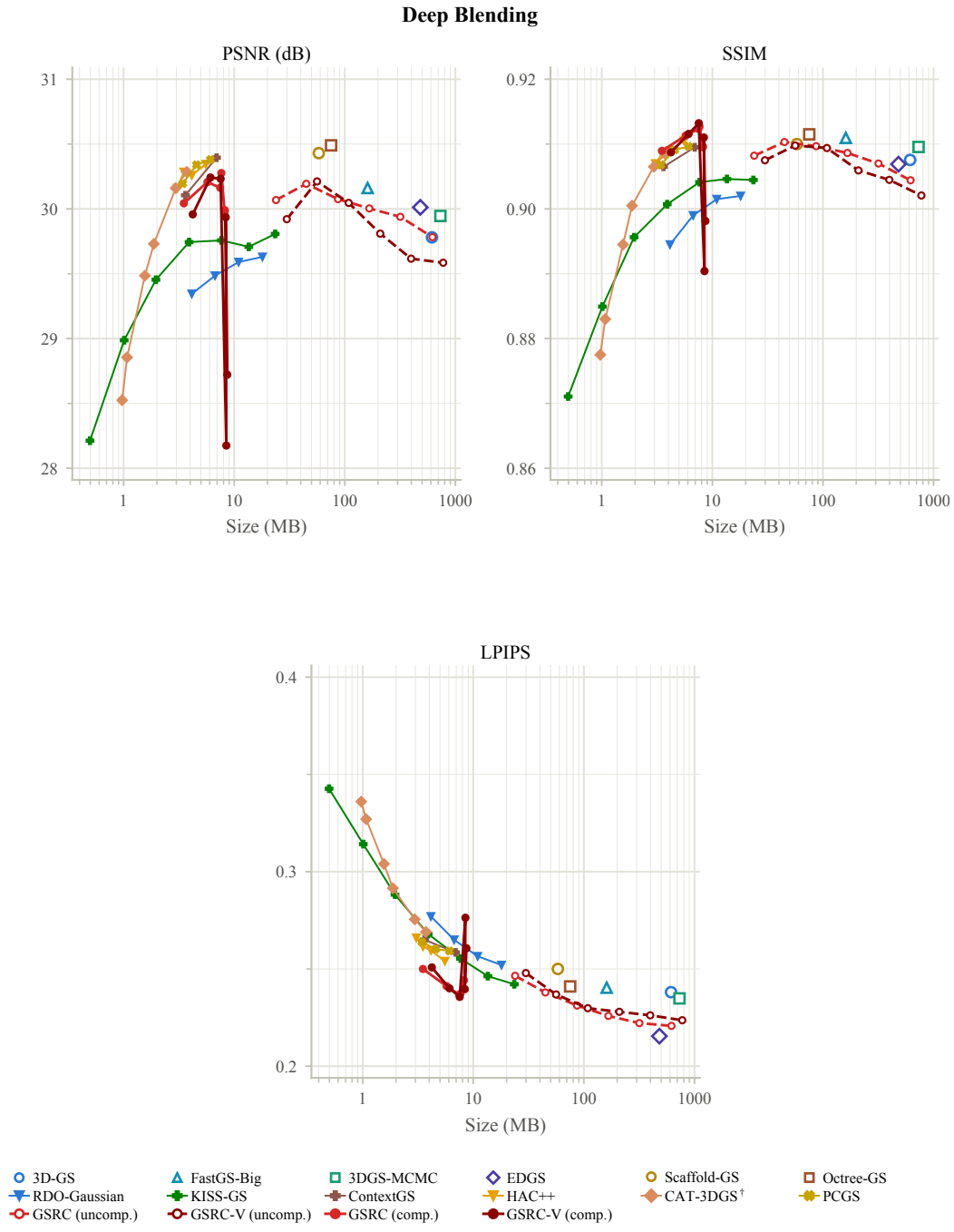


Figure 11: **Rate–distortion on Deep Blending.**

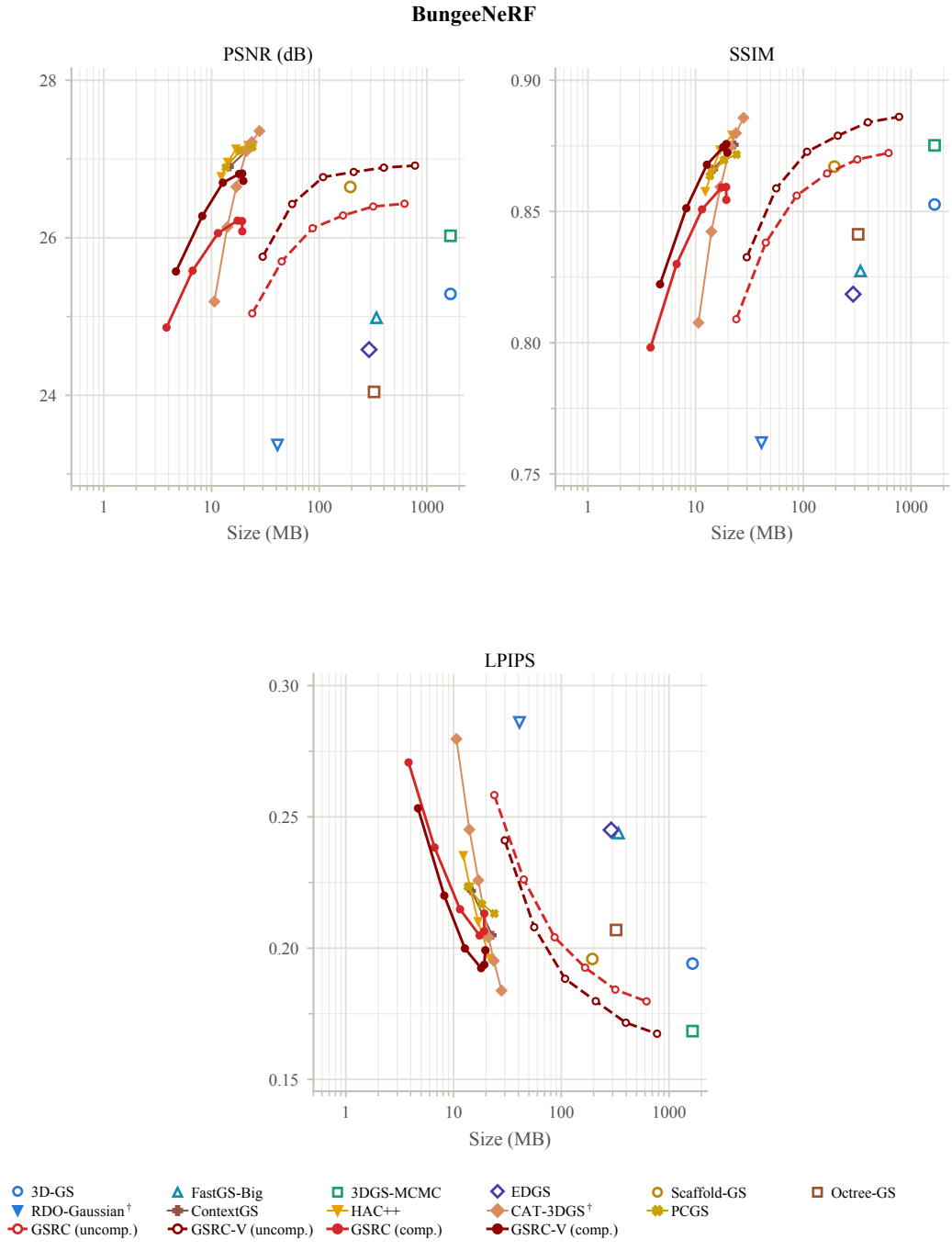


Figure 12: **Rate–distortion on BungeeNeRF.**

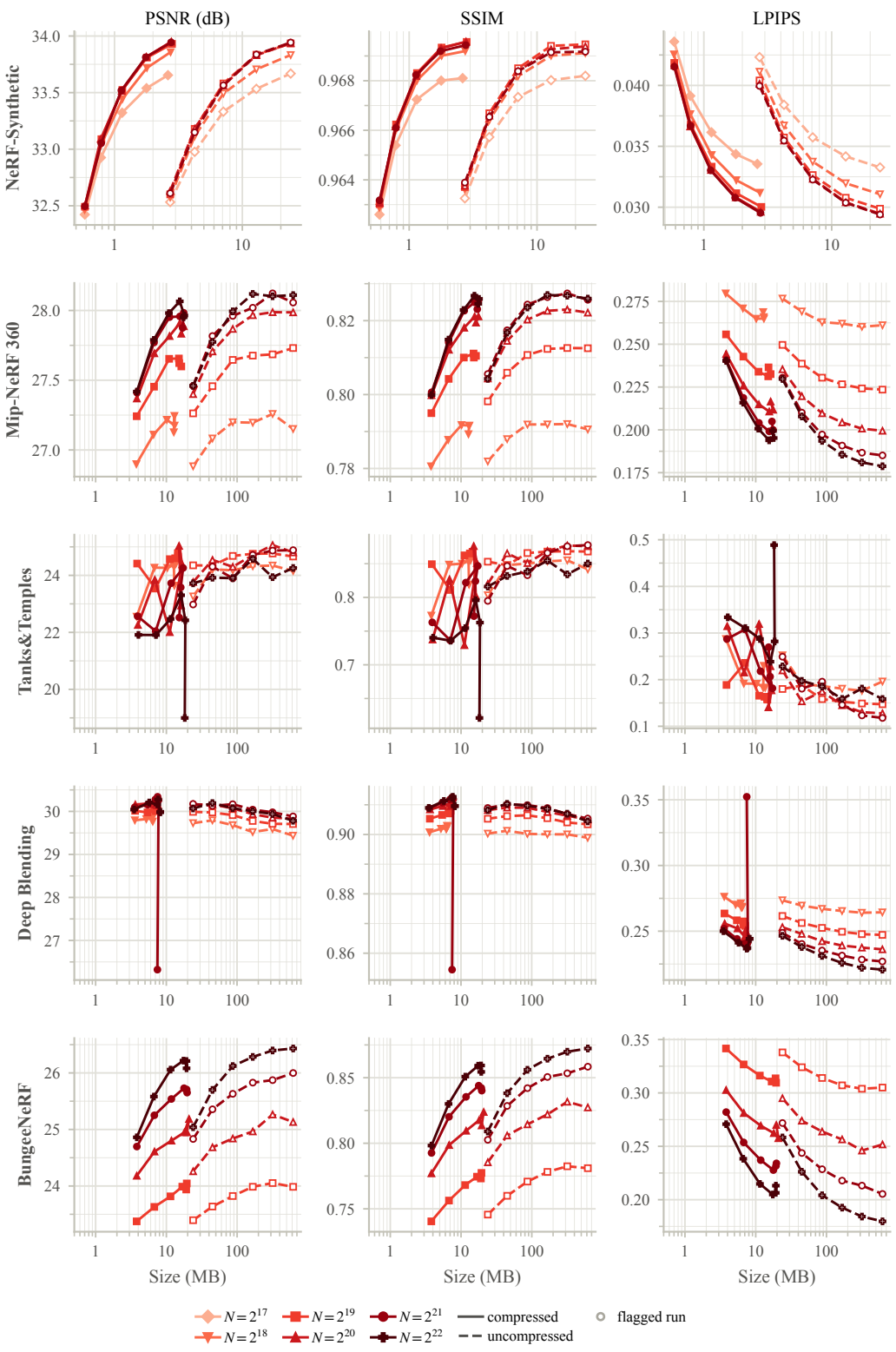


Figure 13: **The full $N \times T$ grid of GSRC.** One curve per number of Gaussians N, its points in order of the table size T (13–17 on NeRF-Synthetic, 17–22 elsewhere), compressed at $\lambda = 1024$ (solid) and uncompressed (dashed). Size in MB (10^6 bytes), log scale. Larger N raises quality at almost the same size, while T moves along the curve. Hollow markers: runs that collapsed during the rate-aware fine-tuning, kept on their curves.

A.6 QUALITATIVE RESULTS

Figures 14 and 15 compare crops of test views from GSRC or GSRC-V and the context-model codecs at similar sizes.

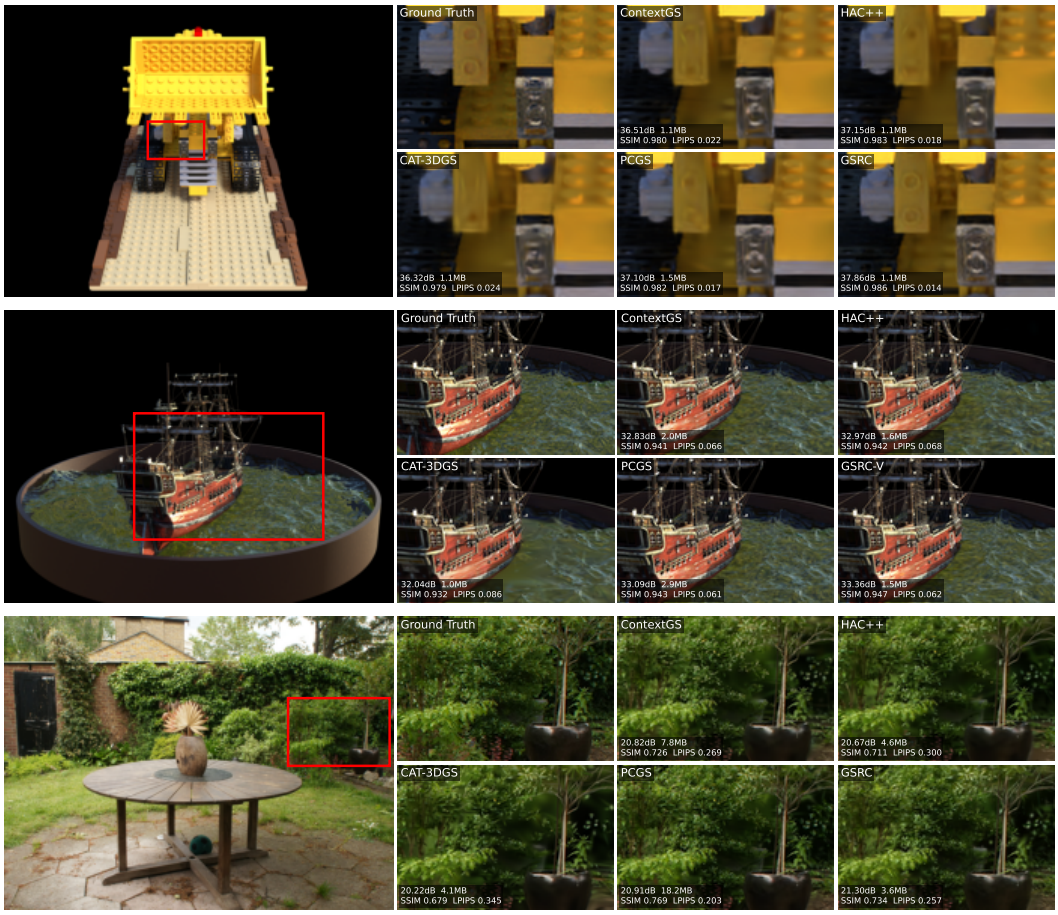


Figure 14: **Qualitative comparison (1/2).** From top to bottom, lego and ship from NeRF-Synthetic, and garden from Mip-NeRF 360. The left column shows the full test view with the cropped region in red, and each crop is labelled with the PSNR, size, SSIM and LPIPS of its method.

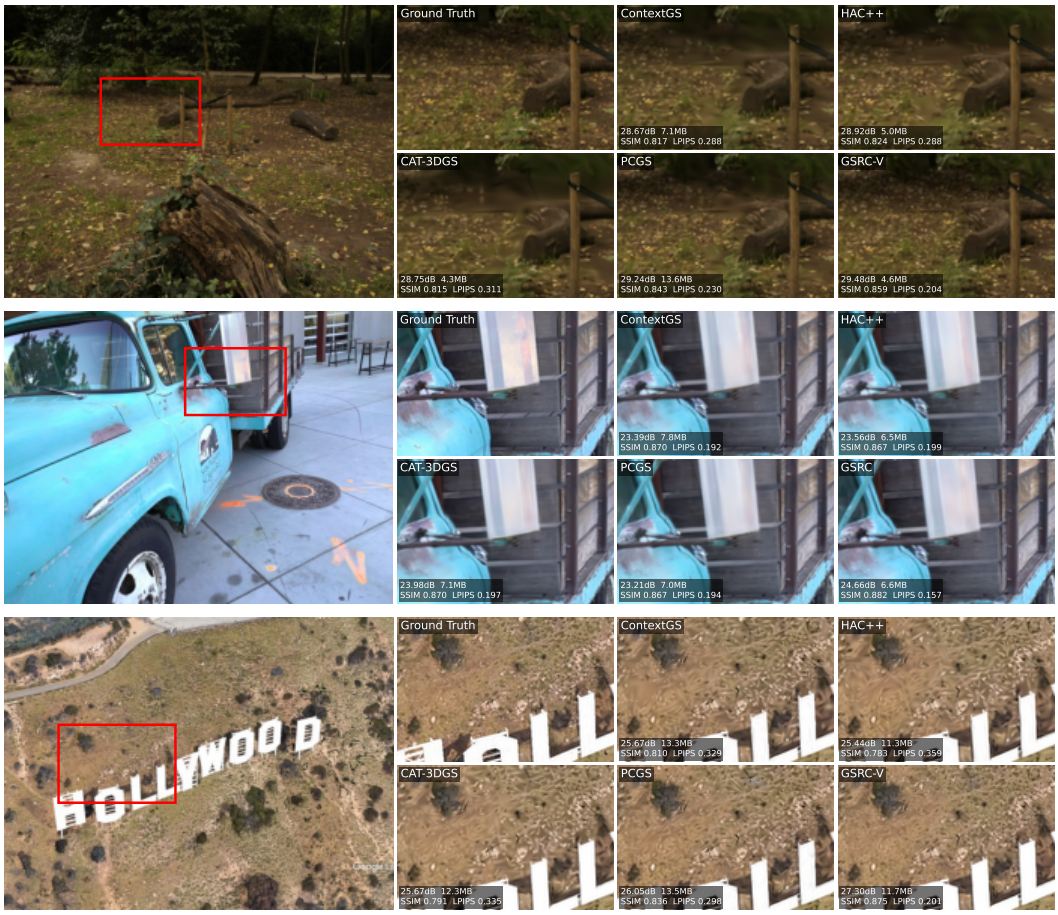


Figure 15: **Qualitative comparison (2/2).** From top to bottom, stump from Mip-NeRF 360, truck from Tanks&Temples and hollywood from BungeeNeRF, shown as in Figure 14.