ccSplat for Compact and Compressed Feed-Forward 3D Gaussian Splatting

Anonymous authors
Paper under double-blind review

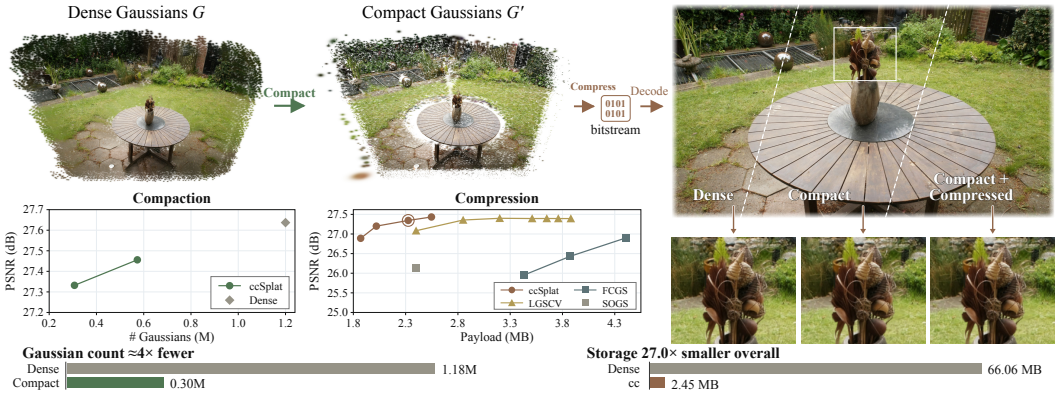

Figure 1. **ccSplat at a glance.** ccSplat compacts dense pixel-aligned Gaussian predictions and compresses the retained attributes. The visualizations and rate–distortion plots illustrate the reductions in primitive count and storage alongside rendering quality. Experiments evaluate both stages on SHARP and AnySplat.

Abstract

Feed-forward 3D Gaussian Splatting (3DGS) has emerged as a promising paradigm for efficient 3D reconstruction and rendering. However, existing methods often predict millions of Gaussians per scene, leading to substantial storage and transmission costs. We present ccSplat, a framework that leverages the pixel-aligned property of feed-forward 3DGS to jointly reduce Gaussian count and compress Gaussian attributes. For compaction, ccSplat builds on the pixel-aligned representation with lightweight pixel-wise importance and scale-refinement heads, enabling Gaussian pruning and coverage recovery on a frozen pre-trained backbone. For compression, ccSplat exploits the retained pixel grid to preserve useful image and neighborhood context, enabling effective prediction and compression of Gaussian attributes. Experiments on both the single-view backbone SHARP and the multi-view backbone AnySplat validate ccSplat across different feed-forward 3DGS settings. On nine Mip-NeRF 360 scenes with SHARP, ccSplat reduces the Gaussian count from 1.18M to 0.30M and storage from 66.06 to 2.45 MB, achieving a 27.0× reduction, while source-view PSNR changes only from 27.63 to 27.26 dB after compaction and compression.

1 Introduction

Feed-forward 3D Gaussian Splatting (3DGS) predicts a renderable scene from one or a few images in a single pass (Charatan et al., 2024; Chen et al., 2024b; Mescheder et al., 2026; Jiang et al., 2025). Many backbones assign one or more Gaussians to each input pixel. Their output size therefore grows with image resolution, layer count, or the number of views, often reaching over a million primitives. Although these models reconstruct scenes quickly, the dense outputs remain costly to render, store, and transmit.

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.

Existing efforts to reduce the storage footprint of feed-forward 3DGS have largely focused on *compaction*, i.e., reducing the number of predicted Gaussians through pruning or more compact prediction. Some methods generate Gaussians around sparse 3D anchors (Zhang et al., 2026b;a), others merge or predict them through voxel-based representations (Jiang et al., 2025; Wang et al., 2026), or place primitives at sub-pixel locations (Moreau et al., 2026). While effective at reducing Gaussian count, these approaches alter the original pixel-grid organization of feed-forward predictions. Reducing the number of Gaussians, however, addresses only part of the storage problem: practical storage and transmission further require *compression* into a compact bitstream. For per-scene optimized 3DGS, where Gaussians do not come with a natural ordering, their organization is itself an important component of compression, motivating methods that reorder attributes into image- or video-friendly layouts (Morgenstern et al., 2024; Yang et al., 2026b) or learn dependencies across Gaussians and attributes (Chen et al., 2025). Feed-forward 3DGS differs in that its predictions already inherit a structured organization from the input images. This suggests that the native pixel grid should not merely be discarded during compaction and reconstructed through a new ordering for compression, but instead preserved and directly exploited as a useful coding structure.

We introduce **cc**Splat to jointly address **c**ompaction and **c**ompression while preserving the native pixel-grid organization of feed-forward 3DGS. Rather than replacing this structure during compaction, ccSplat removes redundant Gaussians while keeping the retained ones associated with their original pixels, layers, or views. The resulting sparse pixel grid remains aligned with the input images and preserves the spatial relationships among neighboring predictions. This structure supports pixel-aligned compaction and, more importantly, provides a natural order and useful neighborhood context for compressing the retained Gaussian attributes.

For compaction, ccSplat augments a frozen pre-trained feed-forward 3DGS prediction backbone with lightweight pixel-wise importance and scale-refinement heads. The importance head is supervised by the rendering degradation caused by removing individual Gaussians, enabling pruning under a prescribed count budget, while the refinement head adjusts the scales of retained Gaussians to recover lost coverage. For compression, ccSplat preserves the retained locations on the original sparse pixel grid, which provides a shared spatial organization across attributes. The native pixel order and its associated image and neighborhood context are then exploited to predict and encode Gaussian attributes efficiently.

ccSplat supports pixel-aligned outputs from both monocular multilayer and multi-view feed-forward 3DGS backbones. We evaluate the full compaction-and-compression pipeline on two representative monocular and multi-view backbones, SHARP (Mescheder et al., 2026) and AnySplat (Jiang et al., 2025), respectively. On nine Mip-NeRF 360 scenes with SHARP, ccSplat reduces the mean Gaussian count from 1.18M to 0.30M and the storage from 66.06 to 2.45 MB, corresponding to a $27.0\times$ reduction, while source-view PSNR decreases only from 27.63 to 27.26 dB. On AnySplat, ccSplat retains at most 300K Gaussians while improving mean PSNR and SSIM over voxel fusion on held-out views. Ablation studies further validate the learned importance supervision and scale refinement, and show that preserving the native pixel order improves coding efficiency over Morton reordering. Figure 1 summarizes these gains.

Our contributions are twofold. First, we develop a compaction method that learns rendering importance and refines Gaussian scales to preserve coverage under a count budget, while retaining the pixel correspondence needed for subsequent attribute coding. Second, we design a codec that exploits this correspondence to recover shared sparse locations and predict Gaussian attributes from image and grid context, yielding a compact bitstream for the retained representation.

2 Related Work

Pixel-aligned reconstruction. Feed-forward methods predict Gaussian scenes directly from input images. Monocular predictors produce single-layer or multilayer maps (Szymanowicz et al., 2024; 2025; Mescheder et al., 2026), while multi-view predictors associate candidates with pixels across input views (Charatan et al., 2024; Chen et al., 2024b; Xu et al., 2025; Jiang et al., 2025). SHARP and AnySplat exemplify these settings. ccSplat operates on their pixel-aligned candidates, before view fusion where applicable, retaining their source-image associations.

Gaussian compaction. Compact predictors use sparse anchors with 3D-neighborhood attribute prediction (Zhang et al., 2026b), voxel or anchor supports (Wang et al., 2026; Zhang et al., 2026a), learned queries (An et al., 2026), or sub-pixel detection (Moreau et al., 2026). Adaptive allocation also uses densification scores predicted on multiscale maps (Kim et al., 2026). For reconstructed scenes, importance or sensitivity estimates guide selection (Fan et al., 2024; Hanson et al., 2025; Yang et al., 2026a). Pixel-aligned approaches can retain source associations while adjusting the selected representation. EcoSplat (Park et al., 2026) learns opacity-based selection and budget-conditioned attributes. ccSplat trains a separate importance head using the change in rendering error caused by Gaussian removal and refines retained scales conditioned on the selection mask. This design reduces the count and recovers coverage while preserving the original pixel addresses. The selected Gaussians therefore retain an organization that can also supply order and context for subsequent attribute coding.

Gaussian compression. Gaussian codecs exploit attribute redundancy through learned contexts and coding layouts. Context models capture spatial and attribute dependencies (Chen et al., 2024a; 2025), including long-range neighborhoods defined by Morton serialization (Liu et al., 2025). FCGS also supports outputs of feed-forward reconstruction networks. Other methods arrange attributes in smooth image or video grids (Morgenstern et al., 2024; Yang et al., 2026b). For pixel-aligned predictions, TinySplat (Song et al., 2026) exploits source-view geometry through camera-space transforms and compresses per-view attribute maps with a video codec. CodecSplat (Yu et al., 2026) instead codes intermediate generation features, from which the decoder predicts depth and Gaussian attributes. ccSplat codes the sparse explicit attributes retained by learned compaction. Their original pixel associations link color prediction to the decoded source image and position prediction to decoded grid neighbors.

3 METHOD

ccSplat uses the original pixel grid to connect Gaussian compaction and attribute compression (Figure 2). Compaction learns which Gaussians to retain and adjusts their scales while preserving their pixel addresses. Compression reuses these addresses to organize sparse attributes and predict them from image and grid context.

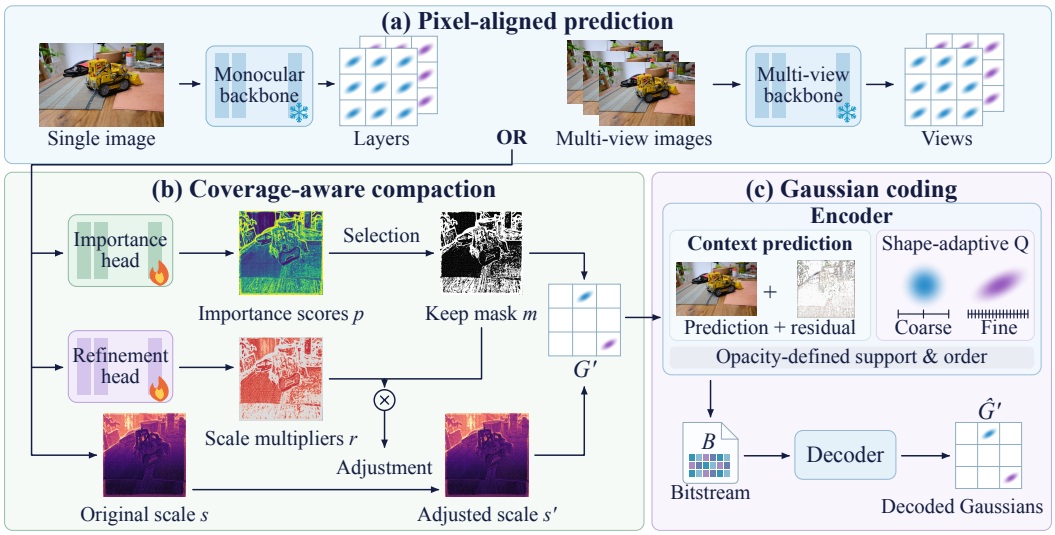

Figure 2. **Overview of ccSplat.** (a) Monocular multilayer or multi-view predictions provide pixel-aligned Gaussians. (b) Importance and coverage guide selection under a count budget. The refinement head adjusts retained scales using the mask, while pixel addresses are preserved. (c) Decoded opacity identifies the shared retained pixel layout for context-guided prediction and shape-adaptive quantization. Color prediction uses a compressed input image, and position prediction uses decoded grid neighbors.

3.1 PIXEL-ALIGNED REPRESENTATION

A monocular backbone can predict multiple Gaussian layers from one image, with each layer assigning a candidate to each pixel. A multi-view backbone instead produces candidates associated with pixels in different input views before fusion. Both outputs therefore retain a correspondence between each candidate and its generating pixel. In this work, we consider SHARP (Mescheder et al., 2026) and AnySplat (Jiang et al., 2025) as representative backbones for the monocular multilayer and multi-view settings, respectively.

We record this correspondence with an address $j = (v, \ell, r, c) \in \Omega$, where v identifies the view, ℓ the layer, and (r, c) the pixel. Unused indices take a single value. Each candidate $g_j = (\boldsymbol{\mu}_j, \mathbf{s}_j, \mathbf{q}_j, \mathbf{c}_j, \alpha_j)$ contains a 3D center, scales, a unit quaternion, color, and opacity. The address identifies its source pixel, while $\boldsymbol{\mu}_j$ is its absolute world-space center. The dense representation is $\mathcal{G} = \{g_j \mid j \in \Omega\}$.

3.2 COVERAGE-AWARE COMPACTION

Given a maximum count K, compaction produces a binary selection mask $m_j \in \{0, 1\}$ and a compact representation

$$\mathcal{A} = \{j \mid m_j = 1\}, \qquad \mathcal{G}' = \{g'_j \mid j \in \mathcal{A}\}, \qquad |\mathcal{A}| \le K. \tag{1}$$

We learn rendering importance, select a spatially distributed subset, and adjust its scales to compensate for coverage lost through pruning. Each retained Gaussian keeps its original pixel address.

Learning rendering importance. A lightweight head predicts $z_j = f_\theta(\mathbf{F}_j, \mathbf{a}_j)$ and $p_j = \sigma(z_j)$ from frozen pixel-aligned features $\mathbf{F}_j$ and cues $\mathbf{a}_j$ derived from depth, opacity, scale, and local coverage. To supervise rendering importance, we delete each primitive and re-render the scene. With renderer R and target image I, its removal score is

$$\delta_j = \|R(\mathcal{G} \setminus \{g_j\}) - I\|_2^2 - \|R(\mathcal{G}) - I\|_2^2. \tag{2}$$

A positive score indicates that removing the primitive increases the rendering error. We rank candidates by removal score within each layer and select them using the layer budgets assigned to compaction. The selected candidates form a binary teacher mask that supervises p_j. This teacher is used only during training. At inference, the importance head predicts the scores in a forward pass.

Coverage-aware selection. Global importance ranking can concentrate the budget in textured regions. We therefore distribute selections across the pixel grid. First, we increase the scores of candidates with few high-scoring neighbors and have each tile propose its highest-scoring candidate. A bounded part of the budget is also reserved for candidates scored by opacity, scale, and depth to preserve surface coverage. Global ranking of the adjusted importance scores then fills the remaining slots. The resulting mask m respects the count budget. Appendix A gives the scores and allocation rules.

Scale refinement. Selection alone can leave gaps in the dense prediction's coverage. The refinement head conditions on the retained mask and aligned features, predicting a positive scalar r_j for each retained Gaussian

$$\mathbf{s}'_j = r_j \mathbf{s}_j, \qquad j \in \mathcal{A}. \tag{3}$$

The adjustment changes footprint size while preserving scale-axis ratios and all other attributes. The output $\mathcal{G}'$ retains each Gaussian's original pixel address. This sparse grid preserves the pixel order and the image and neighborhood context used by compression.

Training. For each backbone, we train a separate pair of importance and refinement heads shared across scenes. Image reconstruction and perceptual losses supervise source-view renderings, supplemented by renderings from the frozen backbone at nearby viewpoints. A straight-through mask (Bengio et al., 2013) uses the hard selected subset in the forward pass and passes gradients to continuous scores. The backbone remains frozen throughout. Appendix A.1 gives the full objective and schedule.

3.3 COMPRESSION ON THE RETAINED PIXEL GRID

We describe the codec for the two-layer SHARP output. Each retained address in $\mathcal{A}$ includes its layer index and is shared by all attributes of that Gaussian. The opacity stream carries opacity values and

identifies the retained addresses. On the original grid, we assign zero to removed sites and positive quantized opacity codes to retained sites. Storing these integer codes losslessly lets the decoder recover $\mathcal{A}$ exactly. The encoder packs the other attributes at retained sites in a fixed pixel order. The decoder uses the recovered addresses to restore these values to the sparse grid without a separate index stream. This shared layout supports the two coding mechanisms below.

Context-guided prediction. The retained addresses provide image and neighborhood context for predicting attributes. Both ends use the same decoded context, and the encoder transmits residuals only at active sites.

For color, we store a compressed input RGB image in the bitstream. Both Gaussian layers remain aligned with this image, so its decoded pixels at the retained addresses predict color in both layers. We encode the quantized differences between Gaussian colors and these predictions.

For position, neighboring pixels often correspond to nearby points on a surface, making their Gaussian centers correlated. We first quantize each 3D center coarsely. Retained grid neighbors that have already been decoded predict these coarse coordinate codes, and we encode the prediction errors to reduce spatial redundancy. An additional coordinate residual relative to the reconstructed coarse center reduces quantization error. Prediction uses the original pixel grid even though the attribute values are stored compactly. Missing neighbors are excluded. Appendix B.1 specifies the predictors and residual coding.

Shape-adaptive quantization. An isotropic Gaussian is unchanged by rotation, whereas orientation errors can alter an anisotropic Gaussian's projected footprint. The decoded scale axes therefore guide the allocation of rotation precision. We first encode the three scale axes, then use their decoded values to assign finer rotation precision to more anisotropic Gaussians and coarser precision to the others. Both ends derive the same assignments from the decoded scales and the coded allocation settings, avoiding a separate assignment mask. Appendix B.2 gives the shape score and parameterizations.

Multi-view extension. For AnySplat, coding follows each retained view grid. Decoded source-image luminance predicts view-independent luminance, while higher-order spherical harmonic (SH) coefficients use a transmitted low-rank transform. This preserves pixel correspondence for view-dependent appearance (Appendix B.4).

4 Experiments

4.1 Setup

Training and monocular evaluation. For each backbone, we train a separate pair of heads on OpenScene, Getty Images, DL3DV, and Tanks and Temples. Mip-NeRF 360 is used only for testing, with weights fixed at the final, 3000th training step. We first evaluate SHARP (Mescheder et al., 2026) on nine Mip-NeRF 360 scenes (Barron et al., 2022), reporting PSNR, SSIM, and LPIPS (Zhang et al., 2018). Source-view metrics compare renderings with input images. Novel-view fidelity compares with dense, uncompressed SHARP renderings at 16 nearby poses per scene, measuring preservation of the original prediction. Appendix C gives the evaluation protocols. To isolate the contributions of compaction and compression, we evaluate the stages separately. Under fixed Gaussian-count budgets, we compare compaction with opacity-based selection, a LightGaussian-style importance proxy (Fan et al., 2024), and RAP (Yang et al., 2026a). We compare FCGS (Chen et al., 2025), SOGS (Morgenstern et al., 2024), and LGSCV (Yang et al., 2026b) on the same 300K-budget compact SHARP output, isolating attribute coding from Gaussian selection. The five ccSplat rate points vary coding precision while retaining this Gaussian set. Storage includes all transmitted streams and metadata in decimal MB.

Multi-view-based backbone. We further evaluate ccSplat with AnySplat (Jiang et al., 2025) to test generalization to multi-view pixel-aligned predictions. AnySplat's voxel fusion merges Gaussians across views and discards their original pixel-grid alignment. To exploit this alignment, ccSplat instead compacts the per-view Gaussian grids directly. We compare compaction with AnySplat's original voxel fusion, then compare compression methods on the same compact representation, as in the SHARP setting.

4.2 RESULTS ON SHARP

Compaction under a common budget. Table 1 compares compaction strategies on the same dense SHARP predictions under a 300K-Gaussian budget, about one quarter of the original count. We also train a scale-refinement head on RAP's selection masks, providing coverage recovery for that selector. ccSplat reaches 27.33 dB, exceeding RAP with refinement by 1.19 dB with higher SSIM and lower LPIPS. This gain supports our selection mechanism even when the baseline also adjusts retained scales. Table 4a separates the contributions of importance supervision and coverage-aware selection.

Figure 3 further shows that our method better preserves continuous surface coverage and fine structures after compaction.

Table 1. **Compaction under a 300K budget on SHARP.** Direct selection and RAP with refinement retain 300K Gaussians. Full ccSplat realizes 299,334 on average. All metrics use the same source views and reference images across nine scenes.

Method	PSNR↑	SSIM↑	LPIPS↓
Opacity selection	21.47	.635	.275
LightGaussian-style score proxy	21.42	.582	.424
RAP selector	24.38	.736	.188
RAP + selector-specific scale refinement	26.14	.775	.199
ccSplat (compaction only)	**27.33**	**.836**	**.113**

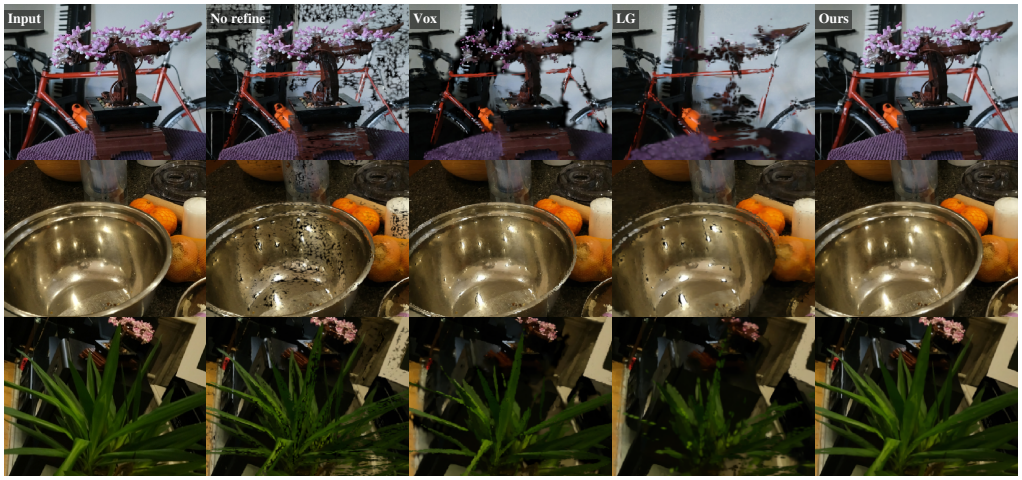

Figure 3. **Qualitative comparison after compaction** across the input, pruning without scale refinement (No refine), voxelization (Vox), LightGaussian-style pruning (LG), and ccSplat (Ours). The bowl surface and plant leaves illustrate coverage gaps and loss of fine structures of the baselines.

Compression on the same compact representation. On the same 300K-budget compact input, our codec reduces storage to 2.45 MB with a further 0.07 dB decrease in source-view PSNR (Table 2), yielding 27.0× compression relative to the original 66.06 MB dense output. Under the same compact input, ccSplat achieves higher source-view reconstruction quality while using less storage than SOGS, FCGS, and LGSCV at the representative operating points. Figure 4 confirms this advantage over a broader range of rate–distortion tradeoffs: ccSplat retains better source-view quality at lower rates, while LGSCV reaches higher fidelity only at its largest rates.

4.3 RESULTS ON ANYSPLAT

Compaction comparison. The upper rows of Table 3 compare compaction of dense AnySplat predictions. Compared with AnySplat's voxel-fusion output of 0.94–1.40M Gaussians, our compact

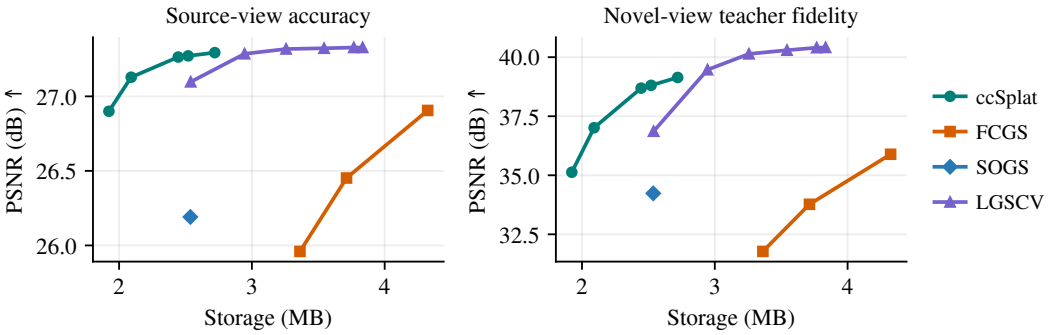

Figure 4. **Rate–distortion comparison on SHARP.** Each point is an equal-scene mean over nine SHARP scenes on the same compact input. The left panel compares source views with input images. The right panel compares novel views with dense SHARP. Storage includes all transmitted streams and metadata. The full PSNR, SSIM, and LPIPS curves are in Figure 9.

Table 2. **Compaction and compression on SHARP.** Equal-scene means on nine Mip-NeRF 360 scenes. All image metrics compare source views with input images. Upper rows precede coding, and lower rows use the same compact input. The representative rate points are SOGS default, FCGS 4×10^{-4}, LGSCV rate point 6, and ccSplat M. Counts are realized means. "–" denotes storage not reported here.

Representation / codec	# Gaussians	MB↓	PSNR↑	SSIM↑	LPIPS↓
Dense SHARP	1,179,648	66.06	27.63	.853	.084
ccSplat (compaction only)	299,334	–	27.33	.836	.113
ccSplat (compaction only) + SOGS	299,334	2.54	26.19	.786	.140
ccSplat (compaction only) + FCGS	299,334	3.36	25.96	.770	.164
ccSplat (compaction only) + LGSCV	299,334	2.54	27.10	.818	.129
ccSplat (compaction + compression)	299,334	**2.45**	**27.26**	**.832**	**.115**

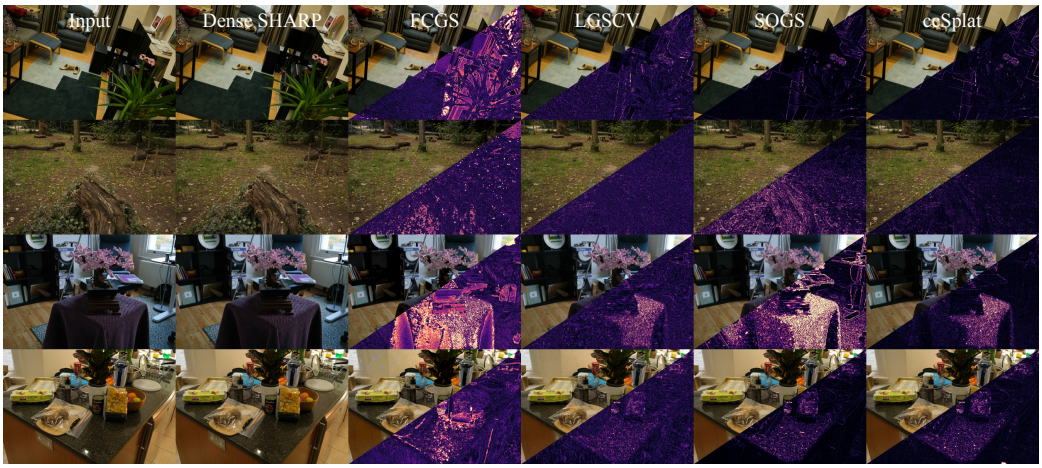

Figure 5. **Qualitative comparison on SHARP.** Rows show room, stump, bonsai, and counter. Rate points match Table 2, including our shared-RGB M rate point. Upper-left triangles show renderings, and lower-right triangles show mean absolute error from Dense SHARP in $[0, 1]$ sRGB. Errors include compaction and coding losses, with a common range $[0, 0.0575]$ and larger values clipped. Darker colors mean smaller errors.

representation uses at most 300K, while increasing target-view PSNR from 20.79 to 22.94 dB and SSIM from .608 to .681. Figure 6 shows that the reduced representation preserves scene appearance.

Table 3. **Compaction and compression on AnySplat.** Equal-scene means over 60 target views from five scenes. Upper rows evaluate compaction. Lower rows encode the same compact SH3 input.

Representation / codec	# Gaussians	SH degree	MB↓	PSNR↑	SSIM↑	LPIPS↓
AnySplat (voxel fusion)	0.94–1.40M	4	–	20.79	.608	.242
ccSplat (compaction only)	300,000	4	–	22.94	.681	.242
ccSplat (compaction only)	300,000	3	74.40	22.75	.674	.253
ccSplat (compaction only) + SOGS	299,209	3	5.87	22.72	.670	.263
ccSplat (compaction only) + FCGS	300,000	3	5.22	22.67	.666	.265
ccSplat (compaction only) + LGSCV	295,936	3	5.16	22.64	.670	.264
ccSplat (compaction + compression)	300,000	3	5.10	22.72	.672	.258

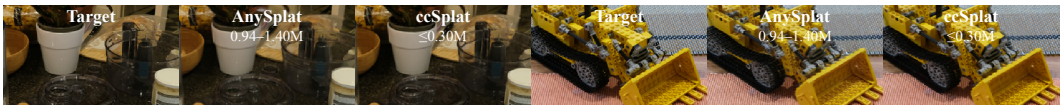

Figure 6. **Appearance after multi-view compaction.** The first and last three patches show counter and kitchen, respectively. Each group shows the target, AnySplat with voxel fusion, and ccSplat. Counts denote the evaluation-wide AnySplat range and our maximum budget.

Compression on the same compact representation. The lower rows of Table 3 compare codecs on the same compact representation. AnySplat outputs SH degree 4, whereas most compared codecs support degree 3. For the codec comparison, we use a compact representation with SH degree 3. Our codec compresses this representation to 5.10 MB with only a 0.03 dB decrease from its uncompressed PSNR of 22.75 dB. Compared with the baselines, ccSplat achieves higher or comparable quality at the smallest size, demonstrating generalization to multi-view backbones.

4.4 Ablation Studies

Table 4. **Compaction and compression ablations.** Nine-scene source-view PSNR in dB. (a) A 300K maximum budget, with ΔPSNR relative to full compaction. (b) M-preset controls with layer-wise HEVC color coding instead of the main shared-RGB carrier.

(a) Compaction

Variant	PSNR↑	ΔPSNR
Full ccSplat	27.33	—
No refinement	25.17	−2.16
No removal teacher	26.46	−0.87
No coverage-aware selection	26.78	−0.55

(b) Compression

Variant	MB↓	PSNR↑
Reference	2.461	27.260
No spatial prediction	2.605	27.260
Direct log scales	2.525	27.263
Uniform rotation	2.441	27.245
No position residual	2.114	25.854

Compaction components. Table 4a tests the roles of importance supervision, coverage-aware selection, and scale refinement. Removing the removal-based teacher lowers PSNR by 0.87 dB, supporting supervision from rendering changes beyond the reconstruction losses. Removing coverage-aware selection costs 0.55 dB, showing the benefit of distributing the budget across the grid. Refinement has the largest effect, with a 2.16 dB drop when removed, consistent with the gaps in Figure 3. The mask-conditioned head adjusts footprints to the selected subset while keeping centers and pixel addresses fixed. These results explain how learned importance and coverage recovery produce a compact representation that retains its structure for coding.

Compression components. Table 4b uses a fixed layer-wise HEVC carrier in place of the main shared-RGB carrier to isolate non-color components. Spatial prediction saves 0.144 MB at unchanged quality, showing that retained grid neighborhoods also provide useful context for position coding. Position residuals add 0.347 MB and improve PSNR by 1.406 dB, preserving accuracy beyond the coarse position base. Appendix D.4 details the variants and the scale–rotation tradeoffs.

Preserving Pixel Correspondence. Compaction preserves a shared pixel layout across all five attributes (Figure 7). The following controls test whether retaining this correspondence helps the codec exploit image context, linking the two stages.

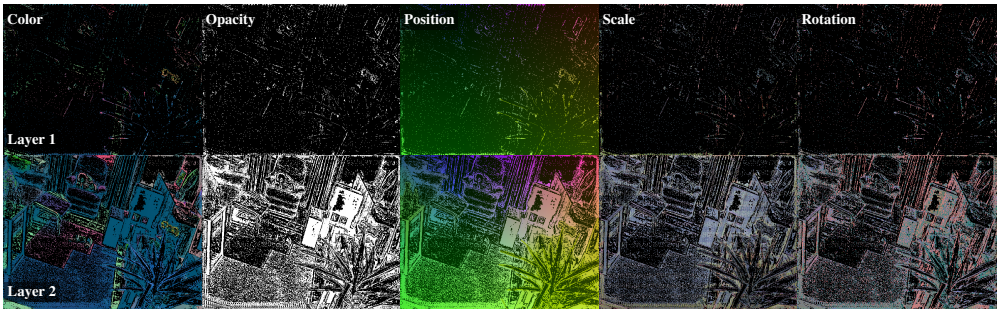

Figure 7. **Shared pixel layout across attributes.** Rows show two Gaussian layers and columns show color, opacity, position, scale, and rotation. Attributes share retained pixel addresses within each layer, while the two layers may retain different sites.

Figure 8 compares four layouts with identical reconstructed attributes and the same pixel-conditioned predictor. Total sizes include the support and permutation streams needed to recover pixel addresses. Preserving the native layout reduces storage from 2.762 MB for Morton to 2.441 MB, an 11.6% reduction.

Without pixel-conditioned color prediction, native and Morton have nearly equal sizes (2.841 and 2.848 MB). Together, these controls show the value of retaining pixel addresses through compaction so that compression can use image context efficiently (Appendix D.3).

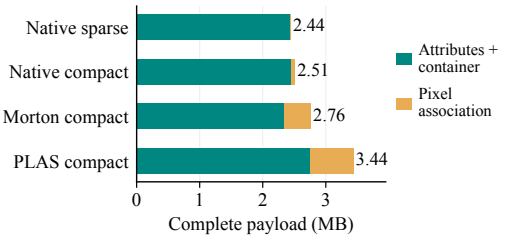

Figure 8. **Coding efficiency of pixel layouts.** Nine-scene M-point means at identical reconstruction. Orange counts support and permutation streams. Implementation details are in Appendix D.3.

5 Conclusion

We presented ccSplat, a framework that connects compaction and compression through the native pixel structure of feed-forward 3DGS. Learned importance and scale refinement reduce Gaussian count and recover coverage while preserving the correspondence between Gaussians and input pixels. The codec then predicts and encodes attributes using the preserved pixel order together with image and grid context. On SHARP, ccSplat achieves approximately $4\times$ fewer Gaussians and $27\times$ storage compression, with a 0.37 dB decrease in source-view PSNR across nine Mip-NeRF 360 scenes. Experiments on AnySplat further demonstrate compaction and compression in the multi-view setting. These results highlight the value of preserving the structure of feed-forward predictions throughout compaction and compression.

Limitations. The compaction heads use aligned backbone features, and the codec relies on retained pixel correspondence. Adapting this design to generators without an explicit pixel structure remains an open direction.

AI USE STATEMENT

We used generative AI tools to assist with literature discovery, language editing, and feedback on methodological and experimental descriptions. The authors reviewed the revisions and are responsible for the final text, figures, references, and scientific claims, including material prepared with the assistance of generative AI.

REPRODUCIBILITY STATEMENT

Section 3 describes the compaction and coding procedures. Appendices A and B provide the training objectives, selection and refinement rules, coding parameters, and decoding procedure. Section 4.1 and Appendix C specify the datasets, evaluation settings, and comparison protocols. Appendix D provides full rate–distortion results, pixel-correspondence controls, component metrics, and additional visual comparisons.

REFERENCES

Honggyu An, Jaewoo Jung, Mungyeom Kim, Chaehyun Kim, Minkyeong Jeon, Jisang Han, Kazumi Fukuda, Takuya Narihira, Hyunah Ko, Junsu Kim, Sunghwan Hong, Yuki Mitsufuji, and Seungryong Kim. Learning compact 3D representations from feed-forward novel view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 363–373, 2026.

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

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. doi: 10.48550/arXiv.1308.3432.

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelSplat: 3D Gaussian splats from image pairs for scalable generalizable 3D reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19457–19467, 2024.

Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. HAC: Hash-grid assisted context for 3D Gaussian splatting compression. In *European Conference on Computer Vision*, volume 15065 of *Lecture Notes in Computer Science*, pp. 422–438. Springer, 2024a. doi: 10.1007/978-3-031-72667-5_24.

Yihang Chen, Qianyi Wu, Mengyao Li, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. Fast feedforward 3D Gaussian splatting compression. In *International Conference on Learning Representations*, 2025.

Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. MVSplat: Efficient 3D Gaussian splatting from sparse multi-view images. In *European Conference on Computer Vision*, volume 15079 of *Lecture Notes in Computer Science*, pp. 370–386. Springer, 2024b. doi: 10.1007/978-3-031-72664-4_21.

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. LightGaussian: Unbounded 3D Gaussian compression with 15x reduction and 200+ FPS. In *Advances in Neural Information Processing Systems*, volume 37, pp. 140138–140158. Curran Associates, Inc., 2024. doi: 10.52202/079017-4447.

Alex Hanson, Allen Tu, Vasu Singla, Mayuka Jayawardhana, Matthias Zwicker, and Tom Goldstein. PUP 3D-GS: Principled uncertainty pruning for 3D Gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5949–5958. IEEE, 2025. doi: 10.1109/CVPR52734.2025.00558.

Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, Dahua Lin, and Bo Dai. AnySplat: Feed-forward 3D Gaussian splatting from unconstrained views. *ACM Transactions on Graphics*, 44(6):1–16, 2025. doi: 10.1145/3763326.

Injae Kim, Chaehyeon Kim, Minseong Bae, Minseok Joo, and Hyunwoo J. Kim. F4Splat: Feed-forward predictive densification for feed-forward 3D Gaussian splatting. In *European Conference on Computer Vision*, volume 17027 of *Lecture Notes in Computer Science*, pp. 1–19. Springer, 2026. doi: 10.1007/978-3-032-37602-2_1.

Zhening Liu, Rui Song, Yushi Huang, Yingdong Hu, Xinjie Zhang, Jiawei Shao, Zehong Lin, and Jun Zhang. Feed-forward 3D Gaussian splatting compression with long-context modeling. *arXiv preprint arXiv:2512.00877*, 2025. doi: 10.48550/arXiv.2512.00877.

Lars Mescheder, Wei Dong, Shiwei Li, Xuyang Bai, Marcel Santos, Peiyun Hu, Bruno Lecouat, Mingmin Zhen, Amaël Delaunoy, Tian Fang, Yanghai Tsin, Stephan R. Richter, and Vladlen Koltun. Sharp monocular view synthesis in less than a second. In *International Conference on Learning Representations*, 2026.

Arthur Moreau, Richard Shaw, Michal Nazarczuk, Jisu Shin, Thomas Tanay, Zhensong Zhang, Songcen Xu, and Eduardo Pérez-Pellitero. Off The Grid: Detection of primitives for feed-forward 3D Gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11756–11766, 2026.

Wieland Morgenstern, Florian Barthel, Anna Hilsmann, and Peter Eisert. Compact 3D scene representation via self-organizing Gaussian grids. In *European Conference on Computer Vision*, volume 15143 of *Lecture Notes in Computer Science*, pp. 18–34. Springer, 2024. doi: 10.1007/978-3-031-73013-9_2.

Jongmin Park, Minh-Quan Viet Bui, Juan Luis Gonzalez Bello, Jaeho Moon, Jihyong Oh, and Munchurl Kim. EcoSplat: Efficiency-controllable feed-forward 3D Gaussian splatting from multi-view images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26010–26020, 2026.

Zetian Song, Jiaye Fu, Jiaqi Zhang, Xiaohan Lu, Chuanmin Jia, Siwei Ma, and Wen Gao. TinySplat: Feedforward approach for generating compact 3D scene representation. *IEEE Transactions on Circuits and Systems for Video Technology*, 36(4):5567–5580, 2026. doi: 10.1109/TCSVT.2026.3664794.

Stanislaw Szymanowicz, Christian Rupprecht, and Andrea Vedaldi. Splatter Image: Ultra-fast single-view 3D reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10208–10217, 2024.

Stanislaw Szymanowicz, Eldar Insafutdinov, Chuanxia Zheng, Dylan Campbell, João F. Henriques, Christian Rupprecht, and Andrea Vedaldi. Flash3D: Feed-forward generalisable 3D scene reconstruction from a single image. In *2025 International Conference on 3D Vision (3DV)*, pp. 670–681. IEEE, 2025.

Weijie Wang, Yeqing Chen, Zeyu Zhang, Hengyu Liu, Haoxiao Wang, Zhiyuan Feng, Wenkang Qin, Feng Chen, Jia-Wang Bian, Zheng Zhu, Donny Y. Chen, and Bohan Zhuang. VolSplat: Rethinking feed-forward 3D Gaussian splatting with voxel-aligned prediction. In *European Conference on Computer Vision*, volume 17014 of *Lecture Notes in Computer Science*, pp. 1–19. Springer, 2026. doi: 10.1007/978-3-032-37232-1_1.

Haofei Xu, Songyou Peng, Fangjinhua Wang, Hermann Blum, Daniel Barath, Andreas Geiger, and Marc Pollefeys. DepthSplat: Connecting Gaussian splatting and depth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16453–16463, 2025.

Kaifa Yang, Qi Yang, Yiling Xu, and Zhu Li. RAP: Fast feedforward rendering-free attribute-guided primitive importance score prediction for efficient 3D Gaussian splatting processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 33323–33332, 2026a.

Qi Yang, Geert Van Der Auwera, and Zhu Li. Lightweight 3D Gaussian splatting compression via video codec. In *2026 Data Compression Conference (DCC)*, pp. 362–371, 2026b. doi: 10.1109/DCC66757.2026.00044.

Pengpeng Yu, Runqing Jiang, Qi Zhang, Dingquan Li, Jing Wang, and Yulan Guo. CodecSplat: Ultra-compact latent coding for feed-forward 3D Gaussian splatting. *arXiv preprint arXiv:2605.25563*, 2026. doi: 10.48550/arXiv.2605.25563.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 586–595, 2018.

Xiaoxue Zhang, Xiaoxu Zheng, Yixuan Yin, Tiao Zhao, Kaihua Tang, Michael Bi Mi, Zhan Xu, and Dave Zhenyu Chen. AnchorSplat: Feed-forward 3D Gaussian splatting with 3D geometric priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18924–18933, 2026a.

Zicheng Zhang, Xiangting Meng, Ke Wu, and Wenchao Ding. SparseSplat: Towards applicable feed-forward 3D Gaussian splatting with pixel-unaligned prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5049–5058, 2026b.

A Compaction Implementation

The following settings describe the two-layer SHARP implementation. For AnySplat, selection operates on each source-view grid before fusion, with retained counts summed across views.

Selection and refinement. For a local mean importance $\bar{p}_j$, the selection score is $p'_j = p_j[1 + \lambda_{\text{div}}(1 - \bar{p}_j)]$. The support score favors larger opaque Gaussians and accounts for opacity across both layers. In the second layer, it also favors regions with smooth depth. With mean scale $\bar{s}$ and depth d, the score is

$$h_{\ell,r,c} = \alpha_{\ell,r,c}\bar{s}_{\ell,r,c} + \lambda_\alpha \alpha_{\ell,r,c} + \lambda_{\text{sh}}(\alpha_{1,r,c} + \alpha_{2,r,c}) + \mathbb{I}_{\ell=2}\frac{\lambda_d}{1 + \|\nabla \log d_{r,c}\|}, \tag{4}$$

Each layer has a budget K_ℓ for its current target ratio ρ_ℓ, with $\sum_\ell K_\ell = K$. Within each layer, each tile proposes its highest-p' candidate, and a bounded fraction of the second-layer budget is reserved for tile-wise maxima of h. Remaining slots are filled by global p' ranking. Excess proposals are truncated by their corresponding scores, keeping all operations within K_ℓ. The final SHARP allocation is approximately 30K and 270K Gaussians in the first and second layers. Invalid export sites can reduce the realized count slightly below 300K. The refinement head maps its scalar output a_j to $r_j = \exp(\text{clip}(a_j, -1, 2))$ and applies the masked scale update $\mathbf{s}'_j = \mathbf{s}_j[1 + (r_j - 1)m_j]$. A mean $|\log r_j|$ penalty over retained sites regularizes this correction.

A.1 Training Objective and Schedule

The importance head uses shallow 1×1 and 3×3 convolutions on aligned features and depth, opacity, scale, and coverage cues. The removal teacher in Equation 2 deletes each Gaussian and re-renders against the same target. Applying the student's layer budgets to its scores gives a teacher mask m^{teach}, supervised by

$$\mathcal{L}_{\text{teach}} = \lambda_{\text{teach}} \|p - \text{sg}(m^{\text{teach}})\|_2^2. \tag{5}$$

Here sg stops gradients. For soft scores $m_s = \sigma(z)$, hard budget mask m_h, and support-reservation mask m_r, we use

$$\widetilde{m}_s = \max(m_s, m_r), \qquad m = m_h - \text{sg}(\widetilde{m}_s) + \widetilde{m}_s. \tag{6}$$

The forward mask follows the inference rule, with gradients through the continuous scores (Bengio et al., 2013). Additional penalties are $\lambda_{\text{budget}} \sum_\ell (\overline{p}_\ell - \rho_\ell)^2$ for the target layer ratios, $\lambda_{\text{smooth}}(\|\nabla_r z\|_2^2 + \|\nabla_c z\|_2^2)$ for logit smoothness, and $\lambda_{\text{cons}}\|p - \text{sg}(m_h)\|_2^2$ for mask consistency.

The objective sums source-view loss, teacher supervision, and the listed regularizers, adding novel-view loss during novel-view cycles. Source-view supervision combines ℓ_1, LPIPS (Zhang et al., 2018), and Gram-matrix matching. Alternating novel-view cycles use dense-backbone renderings as targets with ℓ_1, MSE, and perceptual losses on valid pixels. For each frozen backbone, we train one pair of heads on OpenScene, Getty Images, DL3DV, and Tanks and Temples, sharing the weights across scenes. Mip-NeRF 360 is reserved for testing. Training uses 3000 steps, batch size 1, and base learning rate 5×10^{-5}. All reported tests use the final weights after step 3000. The budget decreases from dense to the final layer ratios over the first 2100 steps and stays at the final budget thereafter. Source and novel-view supervision alternate every 100 steps. Training poses use rotations within $\pm 10^\circ$ and translations within ± 0.01.

B Compression Implementation

Shared support and order. The implementation below describes the compact two-layer SHARP output. The AnySplat extension is described in Appendix B.4. Layer and image dimensions are stored in metadata. For the exported 8-bit opacity divided by 255, α_j^{map}, and precision b_α, it encodes

$$q_j^\alpha = \begin{cases} 0, & j \notin \mathcal{A}, \\ \max(1, \text{round}[\alpha_j^{\text{map}}(2^{b_\alpha} - 1)]), & j \in \mathcal{A}. \end{cases} \tag{7}$$

The integer codes are stored losslessly, so positive codes recover $\mathcal{A}$ exactly. Rendering opacity is $\text{round}[255 q_j^\alpha / (2^{b_\alpha} - 1)]/255$. All attributes gather active sites in layer–row–column order and scatter decoded values back to these addresses. Opacity supplies the shared support and order without a separate index list.

B.1 Context-Guided Prediction

Color. Both Gaussian layers use the same compressed input RGB image for color prediction. The encoder and decoder both use its decoded pixels. SHARP sampling applies EXIF orientation, bilinear resizing to 1536×1536 with aligned corners, and 2×2 average pooling to 768×768. WebP quality is 30. The decoded image's RGB values are divided by 255 and transformed to YCoCg without prior linearization using $T(R, G, B) = (R/4 + G/2 + B/4,\ (R - B)/2,\ -R/4 + G/2 - B/4)$. Gaussian colors are likewise mapped to YCoCg from sRGB. Metadata stores their bounds $c^{\min}_{\ell,k}, c^{\max}_{\ell,k}$ for each layer ℓ and transformed channel k. The normalized 8-bit Gaussian color code is $C_{j,k}$. For prediction, the channel range is bounded below by 10^{-12}, giving $R_{\ell,k} = \max(c^{\max}_{\ell,k} - c^{\min}_{\ell,k}, 10^{-12})$. With decoded RGB image $\widehat{I}$ and residual step Δ_k, we use

$$P_{j,k} = \left\lfloor 255 \ \mathrm{clip}\left(\frac{T(\widehat{I})_{j,k} - c^{\min}_{\ell(j),k}}{R_{\ell(j),k}}, 0, 1\right)\right\rfloor, \tag{8}$$

$$e_{j,k} = \mathrm{round}\left(\frac{C_{j,k} - P_{j,k}}{\Delta_k}\right), \qquad \widehat{C}_{j,k} = \mathrm{clip}(P_{j,k} + \Delta_k e_{j,k}, 0, 255). \tag{9}$$

The six residual streams contain only active sites. Reconstruction follows the attribute-map normalization convention, replacing channel ranges below 10^{-6} by 1. It scales $(\widehat{C}_{j,k} + 0.5)/255$ by this range and adds the channel minimum. Applying T^{-1}, clipping to $[0, 1]$, and converting sRGB to linear RGB then recovers the rendering colors.

Position. Absolute world coordinates are uniformly quantized to a 12-bit base per layer and axis using transmitted bounds. For range $R_\mu = \mu^{\max} - \mu^{\min}$, the base code is $q^0 = \mathrm{round}[4095(\mu - \mu^{\min})/\max(R_\mu, 10^{-12})]$. Reconstruction uses $\widehat{\mu}^b = \mu^{\min} + R_\mu q^0/4095$, recovering constant coordinates when $R_\mu = 0$. Base symbols are predicted on the original sparse lattice from decoded immediate left (L), upper (U), and upper-left (UL) neighbors using

$$\widehat{q} = \begin{cases} \min(L, U), & UL \geq \max(L, U), \\ \max(L, U), & UL \leq \min(L, U), \\ L + U - UL, & \text{otherwise.} \end{cases} \tag{10}$$

With L, U present but UL absent, prediction is $\lfloor (L + U)/2 \rfloor$. Otherwise it copies the first available neighbor in the order L, U, UL, or uses the 12-bit midpoint if none is present. Base residuals $q^0 - \widehat{q}$ are coded separately for the eight neighbor-validity contexts. An additional residual restores precision relative to the reconstructed base $\widehat{\boldsymbol{\mu}}^b$

$$e^{\mu}_{j,a} = \mathrm{round}\left(\frac{\mu_{j,a} - \widehat{\mu}^b_{j,a}}{\Delta_x}\right), \qquad \widehat{\mu}_{j,a} = \widehat{\mu}^b_{j,a} + e^{\mu}_{j,a}\Delta_x. \tag{11}$$

B.2 Shape-Adaptive Quantization

For $l_a = \log s'_a$, the scale transform is

$$t_0 = (l_x + l_y + l_z)/3, \qquad t_1 = l_x - l_y, \qquad t_2 = (l_x + l_y)/2 - l_z. \tag{12}$$

Each layer and component uses a median anchor $b_{\ell,k}$ and codes $q^s_{j,k} = \mathrm{round}[(t_{j,k} - b_{\ell(j),k})/\Delta_s]$. The anchors and step are stored as float32 and read back before quantization. The inverse linear transform and exponentiation recover the three scale axes.

Normalized quaternions are sign-flipped when $q_w < 0$, retaining their sign at $q_w = 0$. Rotation uses three coordinates and reconstructs them by

$$(u, v, \phi) = (q_x, q_y, \mathrm{atan2}(q_z, q_w)), \qquad \widehat{\mathbf{q}} = \mathrm{normalize}[\rho\cos\phi, u, v, \rho\sin\phi], \tag{13}$$

where quantized (u, v) on or outside the unit disk are first projected to radius $\sqrt{1 - 10^{-12}}$, and $\rho = \sqrt{\max(1 - u^2 - v^2, 0)}$. Precision follows reconstructed-scale anisotropy $\eta_j = \max_a \log \widehat{s}_{j,a} - \min_a \log \widehat{s}_{j,a}$. Encoder and decoder rank all retained sites across both layers by decreasing η_j, breaking ties by pixel address. The first $N_f = \mathrm{round}(fN)$ use fine steps, and the rest use coarse steps. Metadata stores the integer N_f and step settings. Each layer and tier retains pixel order and transmits all three coordinates, so no per-site assignment mask is needed.

B.3 Coding Presets and Bitstream

Table 5 gives the five SHARP presets, all using the same compact support. Position steps use input coordinate units, log-scale steps are dimensionless, and phase steps are in radians.

Table 5. **SHARP codec presets.** All use a 12-bit position base and WebP quality 30. Color steps use normalized 8-bit code units. f is the fine rotation fraction, and κ multiplies all three fine rotation steps to give coarse steps.

Preset	$(\Delta_Y, \Delta_{Co}, \Delta_{Cg})$	b_α	Δ_x	Δ_s	$(\Delta_{u,v}, \Delta_\phi)$	f	κ
XH	(1,2,2)	5	.0005	.032	(.012,.024)	.25	4
H	(2,4,4)	4	.0005	.064	(.009,.018)	.25	4
M	(3,5,5)	4	.0005	.064	(.012,.024)	.25	4
L	(4,7,7)	3	.0005	.128	(.024,.048)	.25	8
VL	(4,7,7)	2	.0010	.128	(.024,.048)	.10	8

Signed symbols use ZigZag varints followed by XZ. Position-base streams store lengths for their eight support contexts. Opacity uses uint8 NPY planes compressed with XZ, so b_α describes quantization rather than tightly packed storage. All streams, the WebP carrier, and metadata for dimensions, ranges, anchors, and steps count toward physical payload bytes.

Decoding first recovers opacity and support, then color from the transmitted RGB carrier, position from causal neighbors, and scales before scale-conditioned rotation. Scattering restores the sparse attribute maps. Only the portable payload is required, without the original image, backbone weights, or an external association map.

B.4 AnySplat Extension

Each input view contributes its own pixel grid and compressed luminance image. Quantized logit opacities identify retained sites using a reserved inactive symbol, and the decoded image predicts DC luminance at these sites. Chroma is coded separately. We fit a shared low-rank transform to the 45 non-DC SH coefficients and transmit its mean and basis together with uniformly quantized transform coefficients. For each view, the encoder chooses the smaller of direct coding and causal prediction from retained grid neighbors selected using decoded luminance, then applies byte-plane XZ coding. Position coding uses the retained grid, and rotation precision follows decoded scales as in the SHARP implementation. All image streams, transform parameters, and metadata are included in the reported storage.

C Evaluation Protocols

Monocular evaluation. SHARP uses bicycle, bonsai, counter, flowers, garden, kitchen, room, stump, and treehill from Mip-NeRF 360. Source-view metrics compare with input images. Novel-view fidelity compares with dense SHARP renderings at 16 nearby poses per scene with angular deviations no greater than 10°, measuring preservation of the dense prediction under local viewpoint changes. Metrics are averaged within each scene and then equally across scenes, using PSNR, SSIM, and AlexNet LPIPS. All main codec comparisons use the same 300K-budget compact input. Storage includes all transmitted files in decimal MB (10^6 bytes).

Compaction baselines. Table 1 uses the same dense SHARP predictions. The LightGaussian-style baseline uses an opacity–scale score proxy. RAP uses released weights with a DC-only attribute adapter, and its refined variant uses a scale-refinement head trained for that selector.

Multi-view evaluation. The AnySplat benchmark uses bonsai, counter, garden, kitchen, and room. Each scene has three subsets of eight context and four held-out target views, giving 60 targets. Cameras are jointly estimated from context and target images and aligned with an all-context Sim(3) transform. Targets therefore participate in camera estimation as well as evaluation. Both compaction and compression are evaluated on these target views. All codecs encode the same compact input.

Compressed renderings are compared with target images and with renderings of that input before coding. Metrics are averaged within each scene and then equally across scenes. Figure 10 uses ten coding configurations shared across all subsets, distinct from the five SHARP presets in Table 5.

D Additional Experimental Results

D.1 Full Rate–Distortion Results

Table 6 reports the five fixed-support presets and novel-view fidelity before coding. Figure 9 provides the full source- and novel-view metric curves for all codecs.

Table 6. **Rate–distortion and stage fidelity.** Equal-scene means over nine scenes. Source metrics use input images and novel-view metrics use dense SHARP renderings. The five presets code the same 300K-budget compact input. Storage is complete payload in decimal MB. Dashes mark quantities not reported here.

		Source view			Novel view		
Representation	MB↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Compact ≤600K	–	–	–	–	45.173	.9863	.0197
Compact ≤300K	–	–	–	–	40.539	.9734	.0333
XH	2.720	27.293	0.8340	0.1138	39.143	0.9694	0.0357
H	2.520	27.272	0.8326	0.1144	38.808	0.9676	0.0371
M	2.445	27.264	0.8319	0.1148	38.690	0.9669	0.0377
L	2.091	27.128	0.8260	0.1185	37.013	0.9586	0.0446
VL	1.923	26.900	0.8188	0.1229	35.128	0.9480	0.0546

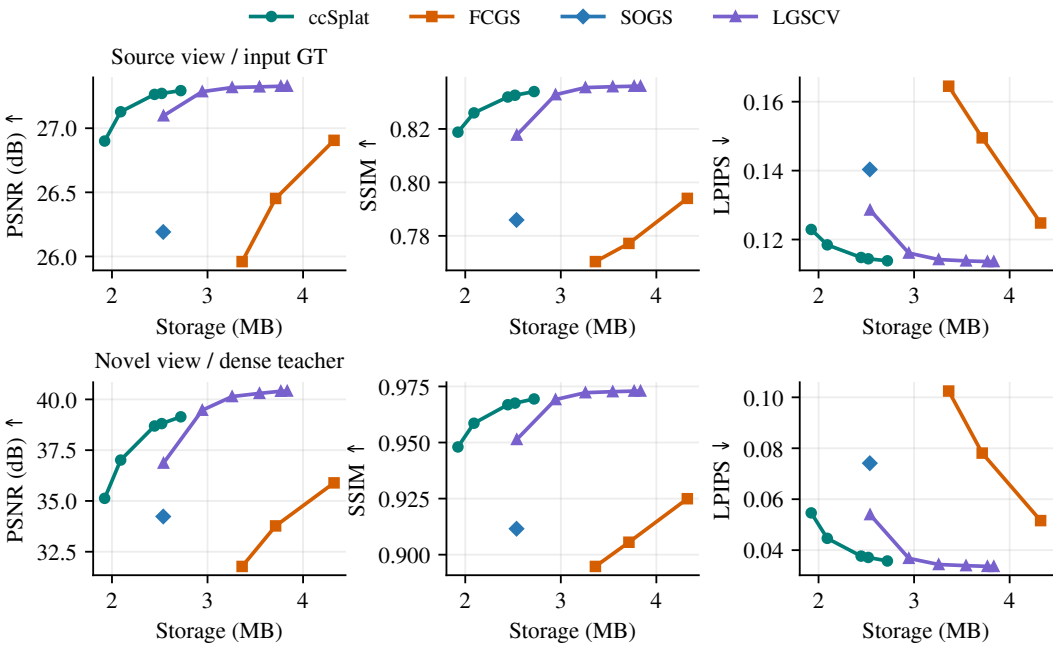

Figure 9. **Full rate–distortion curves.** Equal-scene means over nine scenes. Source metrics use input images and novel-view metrics use dense SHARP renderings. Storage includes all transmitted streams and metadata. ccSplat uses five presets on fixed compact support.

D.2 Additional AnySplat Results

Rate–distortion comparison. Figure 10 compares codecs across rates on the same compact input. At 22.67 dB target-view PSNR, ccSplat uses 3.55 MB compared with 5.22 MB for FCGS, a 32.0% reduction. Relative to LGSCV, ccSplat reduces BD-rate by 22.45% for target-view PSNR and 17.97% for fidelity to renderings of the uncompressed input over the respective overlapping quality ranges. These results extend the coding gains to multi-view predictions with view-dependent appearance.

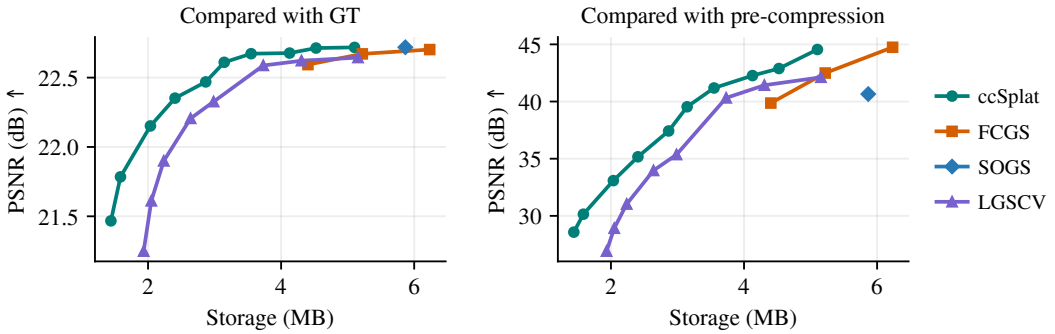

Figure 10. **Compression rate–distortion on AnySplat.** Equal-scene means on the same compact input. Compressed renderings are compared with GT images (left) and pre-compression renderings of the same compact input (right). Storage includes all transmitted streams and metadata. All ten ccSplat presets are shown.

Appearance after compaction. Figure 6 compares matched crops from AnySplat's voxel-fusion output and our compact representation with the target images. The reduced representation preserves scene appearance while using at most 300K Gaussians.

D.3 Pixel Correspondence Controls

For Figure 8, all carriers encode the same quantized Gaussian tuples with the same pixel-conditioned color predictor. Payload-only decoded attribute hashes agree at all nine scenes and five precisions. Association streams comprise support and, for Morton and PLAS, a permutation back to native pixel ranks for the shared image predictor.

Native compact packs raster-ordered active tuples into a square carrier. Morton interleaves bits of normalized 3D coordinates. PLAS uses normalized XYZ and SH-DC RGB, random initialization, minimum block size 16, and improvement threshold 10^{-4}. Zero opacity marks padding. Support is bit-packed and XZ-compressed. Each permutation uses the smallest XZ-compressed representation among forward or inverse mappings in uint32, unsigned varints, or signed-delta varints. The controlled native payload of 2.441 MB and the main payload of 2.445 MB reconstruct identical attributes but use different container metadata.

Table 7. **Pixel correspondence controls.** Nine-scene M-point sizes in MB. Each group reconstructs identical attributes across carriers. Association counts additional support and permutation streams. The RGB-disabled group uses a separate absolute-color codec. The dash denotes an unreported setting.

	RGB predictor enabled			Disabled
Carrier	Other payload	Association	Total	Total
Native sparse	2.441	0.000	2.441	2.841
Native compact	2.449	0.059	2.509	–
Morton compact	2.341	0.420	2.762	2.848
PLAS compact	2.751	0.687	3.438	3.041

The final column of Table 7 disables RGB prediction and codes absolute YCoCg symbols with a common causal syntax. No carrier then transmits support or an inverse permutation solely to restore pixel identity. Native and Morton rates are close in this control, which separates the neighborhood contribution from the combined benefit of pixel-conditioned prediction and implicit association.

D.4 Compression Components

Table 8 expands the main compression ablations. The reference keeps the non-color settings of preset M and uses the same color reconstruction for all variants. Within each layer, active Y and CoCg planes use HEVC with $\mathrm{QP}_Y = 16$ and $\mathrm{QP}_C = 22$.

Table 8. **Compression component metrics.** Nine-scene means with the non-color settings of preset M, fixed compact support, and common HEVC color coding. Source metrics compare with input images. Novel metrics compare with dense SHARP. Storage includes the complete payload in decimal MB.

		Source view			Novel view		
Variant	MB↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Reference	2.461	27.260	0.8317	0.1148	38.640	0.9666	0.0380
No spatial prediction	2.605	27.260	0.8317	0.1148	38.640	0.9666	0.0380
Direct log scales	2.525	27.263	0.8318	0.1146	38.677	0.9667	0.0378
Uniform rotation	2.441	27.245	0.8311	0.1152	38.425	0.9658	0.0385
No position residual	2.114	25.854	0.7642	0.1545	32.316	0.8793	0.0962

No spatial prediction replaces sparse MED with a fixed midpoint, preserving the 12-bit position symbols and decoded attributes. Direct log scales codes three independent log-scale axes, with rotation tiers recomputed from the decoded scales. Uniform rotation uses one precision for all retained sites. No position residual keeps only the 12-bit base, reducing coordinate precision. Thus spatial prediction isolates rate at identical reconstruction, whereas scale and rotation replacements change both rate and distortion at the tested settings.

D.5 Additional Visual Comparisons

Figure 11 complements the four scenes in Figure 5 with the remaining five scenes. All codecs use the common compact input and the operating points in Table 2.

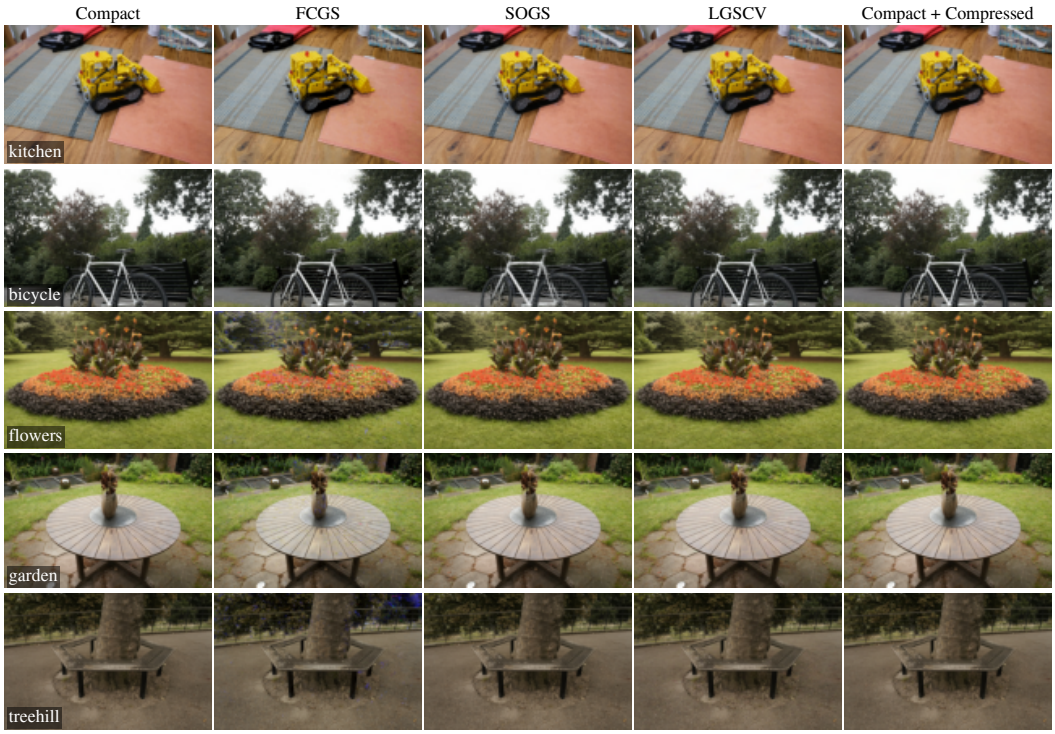

Figure 11. **Additional source-view comparisons.** Rows show kitchen, bicycle, flowers, garden, and treehill. Columns show the compact input, FCGS, SOGS, LGSCV, and compact + compressed ccSplat.