AnyPcc: Compressing Any Point Cloud with a Single Universal Model

Supplementary Material

6. Theoretical Analysis

6.1. Proof of Equivalence between Channel-wise and Spatial Auto-regressive Models

Theorem 1. *Auto-regressive prediction on the 8 channels (bits) of an occupancy code is information-theoretically equivalent to an auto-regressive prediction on the corresponding 8 sub-voxels in space.*

Proof. Let l denote a coarse scale and $l+1$ a finer scale in the voxel hierarchy. A single parent voxel at scale l, located at coordinate $\mathbf{v}^{(l)} \in \mathbb{Z}^3$, corresponds to a $2 \times 2 \times 2$ block of 8 child voxels at scale $l+1$. The coordinate of a child voxel is given by $\mathbf{v}_{\delta}^{(l+1)} = 2\mathbf{v}^{(l)} + \delta$, where the offset $\delta \in \{0,1\}^3$.

Let $O(\mathbf{v}^{(l+1)}) \in \{0,1\}$ be the binary occupancy state of a voxel at scale $l+1$. The 8-bit occupancy code for the parent voxel $\mathbf{v}^{(l)}$, denoted as $o(\mathbf{v}^{(l)})$, encapsulates the occupancy states of its 8 children. We can define a fixed mapping (e.g., lexicographical or Morton order) $\pi : \{0,1,\dots,7\} \to \{0,1\}^3$ that links a bit index k to a spatial offset δ. The k-th bit of the occupancy code is thus defined as:

$$o_k(\mathbf{v}^{(l)}) = O(\mathbf{v}_{\pi(k)}^{(l+1)}). \tag{1}$$

A **spatial auto-regressive model at fine scale** $l+1$ factorizes the joint probability of the 8 child voxels' occupancy as a product of conditional probabilities, following the order defined by π:

$$P(\{O_k^{(l+1)}\}_{k=0}^{7}|\mathcal{C}) = \prod_{k=0}^{7} P(O_k^{(l+1)}|\{O_j^{(l+1)}\}_{j<k}, \mathcal{C}), \tag{2}$$

where $\mathcal{C}$ represents the external context derived from scale l and neighboring voxels, and we use the shorthand $O_k^{(l+1)} \equiv O(\mathbf{v}_{\pi(k)}^{(l+1)})$.

A **channel-wise auto-regressive model at coarse scale** l similarly factorizes the probability of the 8-bit occupancy code:

$$P(o(\mathbf{v}^{(l)})|\mathcal{C}) = \prod_{k=0}^{7} P(o_k(\mathbf{v}^{(l)})|\{o_j(\mathbf{v}^{(l)})\}_{j<k}, \mathcal{C}). \tag{3}$$

By substituting Equation (1) into Equation (3), we can directly observe that the channel-wise model becomes identical to the spatial model in Equation (2). The set of previously decoded bits $\{o_j(\mathbf{v}^{(l)})\}_{j<k}$ provides the exact same information as the set of previously decoded voxels $\{O(\mathbf{v}_{\pi(j)}^{(l+1)})\}_{j<k}$. Therefore, the two modeling approaches are mathematically equivalent. All predictive operations for the local $2 \times 2 \times 2$ block can be performed in the channel dimension of the occupancy code. □

6.2. Proof of Effective Receptive Field Expansion under Sparse Convolutions

Theorem 2. *A sparse convolution with a kernel of size k over the occupancy code space (scale l) achieves an effective receptive field equivalent to that of a sparse convolution with a kernel of size $2k$ applied directly on the finer voxel space (scale $l+1$).*

Proof. This analysis is particularly critical in the context of sparse point clouds, where submanifold sparse convolutions are employed. Unlike dense convolutions where the receptive field expands predictably (e.g., two stacked 3×3 kernels have a 5×5 field), submanifold convolutions only propagate information between active sites (occupied voxels). In highly sparse scenarios, this can severely restrict information flow, making the effective receptive field much smaller than the theoretical kernel size would suggest. Our proof demonstrates how operating on the occupancy code space mitigates this issue.

For clarity, we analyze a 1D convolution, which generalizes to 3D. Consider a sparse convolution with a kernel of size k applied to features associated with occupancy codes at scale l. An output feature at an active site is computed from a neighborhood of up to k active input sites. Let the spatial coordinates of k such active parent voxels at scale l along an axis be $x, x+1, \dots, x+k-1$.

The distance between the centers of the two extreme voxels in this neighborhood is $(x+k-1)-x = k-1$. Each of these parent voxels at scale l maps to a pair of child voxels at scale $l+1$. For a parent at coordinate x, its children are at coordinates $2x$ and $2x+1$.

The receptive field in the finer voxel space (scale $l+1$) is the set of all child voxels corresponding to the parent voxels in the kernel's neighborhood. The first parent voxel at x corresponds to child voxels at $\{2x, 2x+1\}$. The last parent voxel at $x+k-1$ corresponds to child voxels at $\{2(x+k-1), 2(x+k-1)+1\}$.

The total span of the receptive field is the maximum possible distance between any two child voxels within this span. This is realized between a child from the first parent voxel and a child from the last. Let their coordinates be s_{first} and s_{last}.

$$s_{\text{first}} = 2x + \delta_{\text{first}}, \tag{4}$$

$$s_{\text{last}} = 2(x+k-1) + \delta_{\text{last}}, \tag{5}$$

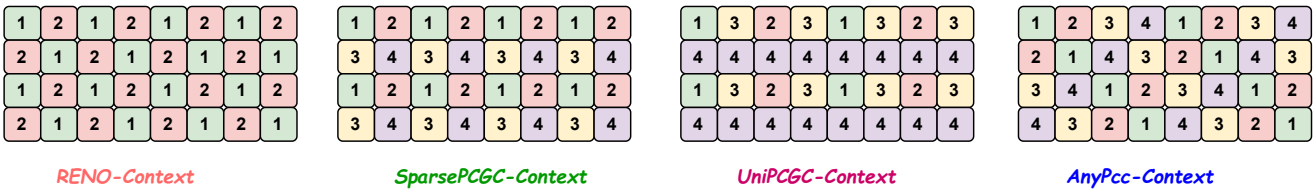

Figure 1. Illustration of our context model in 2D, highlighting its distinction from prior works. The principles shown are directly generalizable to 3D.

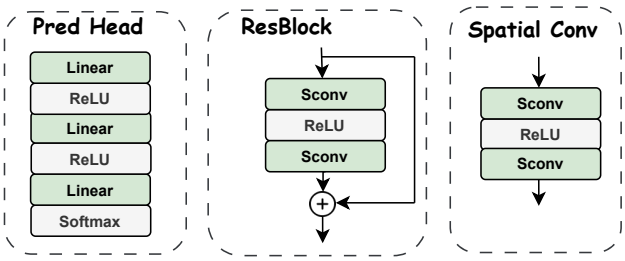

Figure 2. Illustration of the detailed architecture of UCM.

where $\delta \in \{0, 1\}$. The distance is $|s_{\text{last}} - s_{\text{first}}| = |2(x + k - 1) + \delta_{\text{last}} - (2x + \delta_{\text{first}})| = |2k - 2 + \delta_{\text{last}} - \delta_{\text{first}}|$.

To maximize this distance, we set $\delta_{\text{last}} = 1$ and $\delta_{\text{first}} = 0$. The maximum distance is:

$$D_{\max} = 2k - 2 + 1 - 0 = 2k - 1. \tag{6}$$

Thus, a sparse convolution with kernel size k at the coarse scale l can connect features that are up to $2k - 1$ voxels apart at the fine scale $l + 1$. A direct convolution at the fine scale would require a kernel of size $2k$ to achieve the same span $(2k - 1)$, which confirms the theorem. This demonstrates a crucial advantage: by operating on the relatively denser occupancy code representation, our model can establish long-range dependencies that would be broken by extreme sparsity in a direct voxel-level sparse convolution, thereby ensuring a more robust contextual range. □

7. Implementation Details

7.1. Lossless Compression

Building upon the introduction of our Universal Context Model (UCM) and Instance-Adaptive Fine-Tuning (IAFT) strategy in the main paper, this section expands on our method. We provide the example of context model in Figure 1 and the detailed component architectures of our UCM in Figure 2 for reference. The main purpose of this section is to offer a detailed comparison of the UCM's context model against prior works. To facilitate this analysis, we present all comparisons in a consistent 2D setting at the same spatial scale, where a 2×2 block represents the sub-voxels of an occupancy code. The fundamental advantage of our approach is its operation at occupancy code scale, which grants both greater efficiency and a larger receptive field, as illustrated in Figure 2 of the main text. The following analysis will demonstrate this key distinction.

- **RENO** [18] partitions the channels of each occupancy code into two groups for autoregressive processing. Consequently, it overlooks the coarse-grained spatial priors at the occupancy code scale while also lacking any interaction of neighboring information between the two groups. These limitations severely restrict its compression efficiency.
- **SparsePCGC** [12] implements spatial grouping at the voxel level. It shares a key limitation with RENO's channel grouping: both methods exclusively capture fine-grained priors and fail to incorporate coarse-grained spatial information. A further drawback is that its operation at the voxel scale results in a smaller receptive field compared to RENO.
- **UniPCGC** [15] extends SparsePCGC by introducing a non-uniform grouping strategy. It reorganizes and further subdivides SparsePCGC's first group (in the figure, splitting the first group into two new, smaller groups). This design introduces a strong inductive bias: it assumes that the first group provides sufficient contextual priors, and that a more fine-grained modeling of it will yield performance gains. This strategy is effective for uniformly distributed point clouds but fails to generalize to the majority of real-world point clouds, which have highly irregular distributions.
- **AnyPcc** operates at the occupancy code scale to leverage a more comprehensive context. It is uniquely designed to integrate both fine-grained channel priors (*intra-occupancy-code*) and coarse-grained spatial priors (*inter-occupancy-code*), as illustrated in the Figure 1. The model first encodes initial groups $\{1, 2\}$ which build a coarse-grained representation of the local geometry. This representation then serves as a rich spatial prior for encoding the subsequent groups $\{3, 4\}$. Concurrently, dependencies within the groups (e.g., group 2 referencing 1, and 4 referencing 3) capture the fine-grained channel priors. By holistically integrating both types of dependencies, AnyPcc achieves consistently robust performance across point clouds of all types and densities.

Table 1. IImplementation details for our model configuration, training, and compression strategies.

Model Configuration (Lossless Path)	
Feature Channels C	64
Convolution Kernel Size	5
Core Blocks	`Target-Net/Prior-Net` blocks for prior and target feature processing
Prediction Head MLP Arch.	$C \to 32 \to 32 \to 16$
Grouping Strategy	2-Stage Spatial Grouping & 2-Stage Channel Grouping
Model Configuration (Lossy Path)	
Feature Channels C	64
Convolution Kernel Size	5
Core Blocks	`Target-Net/Prior-Net` blocks for prior and target feature processing
Prediction Head MLP Arch.	$C \to 32 \to 32 \to 256$
Grouping Strategy	Single-Grouping
Strategy	Top-k Occupancy Prediction
Configuration Parameter	`--lossy_top_layers` $N = (1, 2, 3)$
Mechanism	Switches to the lossy model for the N finest-resolution layers.
Ours Model Pre-training Hyperparameters	
Optimizer	Adam
Initial Learning Rate	5e-4
Learning Rate Schedule	Decays by a factor of 0.1 at steps 100k and 150k
Total Training Steps	170,000
Batch Size	1
Dataset Sampling	Weighted Random Sampling
Sampling Weights	`KITTI`: 2.5, `8iVFB`: 2.5, `GS`: 2.0, `ScanNet`: 1.0, `Ford`: 1.0, `Thuman`: 1.0
Instance-Adaptive Fine-Tuning (IAFT) Hyperparameters	
Tunable Modules	All `nn.Linear` layers within the four prediction heads
Optimizer	Adam
Learning Rate	4e-3
LR Scheduler	None
Fine-tuning Iterations	200
Loss Function	BPP (Cross-Entropy) + L1 Regularization
L1 Regularization λ	4e-4

7.2. Lossy Compression

In this section, we provide further details on our lossy compression strategy. Our multi-scale representation facilitates a straightforward approach to lossy compression. In a lossless layer, the bitrate $\mathcal{R}_{\text{lossless}}$ is determined by the entropy of the ground-truth occupancy codes $\mathbf{o}_{\text{gt}}$ given the predicted probabilities:

$$\mathcal{R}_{\text{lossless}} = \sum_i -\log_2 P(o_i = o_{i,\text{gt}} \mid \text{Context}_i). \quad (7)$$

A naive lossy approach could simply omit this entropy coding for higher-resolution layers. This approach is well-suited for sparse point clouds. However, this approach leads to a significant loss of points from the point cloud, resulting in a substantial drop in PSNR.

To address this, we adopt a more sophisticated strategy. Instead of transmitting entropy-coded symbols for the designated lossy layers, our encoder transmits only the ground-truth point count [12], $k = |\mathcal{P}_{\text{gt}}|$. The decoder's task is then to predict the k most probable occupied locations from the set of all candidate locations $\mathcal{C}_{\text{cand}}$. This process achieves effective lossy compression while preserving the correct point count. A critical design choice for this is the context model. Our full, lossless models employs a multi-stage prediction scheme, which can lead to error accumulation in a lossy decoding scenario. Therefore, we deliberately switch to the more robust, single-stage model for the lossy layers. The lossy decoding process proceeds as follows. For each candidate location $c_i \in \mathcal{C}_{\text{cand}}$, the baseline model predicts a full probability distribution P_i over the 256 possible occupancy

Table 2. Detailed description of the train dataset.

Dataset	Bit-Depth	Frames / Seq	Total Frames	Cond.	OOD	Avg. Points	Seq*
8iVFB	10-bit	300	600	Easy	✗	954807	Longdress Soldier
MVUB	10-bit	/	941	Easy	✗	1339529	Andrew David Sarah
Thuman2.0	10-bit	/	500	Easy	✗	497907	0000-0499
ScanNet	2cm	/	1513	Easy	✗	147977	Scan
KITTI	1mm	/	23201	Easy	✗	122780	00-10
Ford	1mm	1500	1500	Easy	✗	82627	01
GausPcc-1K	1mm	/	1000	Medium	✗	431165	-

* Seq refers to the specific sequence used in the train, and - means that all sequences of the data set are used.

codes.

1. **Occupancy Probability Estimation:** First, we compute the probability that each location is occupied by summing the probabilities of all non-zero outcomes:

$$p_i^{\text{occ}} = \sum_{v=1}^{255} P_i(o = v) = 1 - P_i(o = 0). \tag{8}$$

2. **Location Selection:** The decoder identifies the set of indices $\mathcal{I}_{\text{dec}}$ corresponding to the k locations with the highest occupancy probabilities:

$$\mathcal{I}_{\text{dec}} = \text{Top-K-Indices}(\{p_i^{\text{occ}}\}_{i \in \mathcal{C}_{\text{cand}}}, k). \tag{9}$$

3. **Value Prediction:** For each selected location c_j where $j \in \mathcal{I}_{\text{dec}}$, the final occupancy code $\hat{o}_j$ is predicted by selecting the most likely non-zero value:

$$\hat{o}_j = \underset{v \in \{1,\dots,255\}}{\arg\max} \; P_j(o = v). \tag{10}$$

The final decoded point cloud for the layer is the set of predicted pairs $\{(c_j, \hat{o}_j)\}_{j \in \mathcal{I}_{\text{dec}}}$. Since this prediction is based on model probabilities rather than ground-truth data, it naturally introduces loss while maintaining geometric structure. This hybrid approach effectively mitigates catastrophic point loss in dense regions and enables a controllable, high-performance lossy compression mode.

8. AnyPcc Benchmark Dataset

8.1. Trainset

Training Datasets. Our training data is comprehensive, incorporating a variety of sources to ensure broad coverage. Specifically, we include standard benchmarks widely used in the Point Cloud Compression (PCC) community, such as `8iVFB`, `MVUB`, `ScanNet`, `KITTI`, and `Ford`. To further enrich our training diversity, we supplement these with the `Thuman` and `GausPcc-1K` [16] datasets. Additionally, we incorporate the `Frog`, `Head`, `Egyptian`, and `ULB` datasets, which are recommended by the MPEG AI-PCC group for training purposes. All training sets are first quantized according to the precisions specified in Table 3. Additionally, the `KITTI` and `Ford` datasets undergo a second quantization step where the resulting integer coordinates are further scaled by a factor of Q=4.

Training Strategy. The prevailing paradigm in Point Cloud Compression (PCC) involves training models that are specialized for distinct data categories. For a fair comparison with existing methods, we first follow this standard practice by training a series of **specialized models**, one for each data category. However, a primary objective of AnyPcc is to develop a single, universal compressor that performs robustly across arbitrary point clouds. To this end, we also pursue a second strategy: training a single **unified model** on a large-scale, composite dataset. This dataset is constructed by sampling from multiple sources with the following rates: `8iVFB` and `MVUB` (2.5, collectively referred to as `8iVFB` for brevity), `Thuman` (1.0), `ScanNet` (1.0), `Ford` (1.0), `KITTI` (2.5), and `GS` (2.0). This unified approach facilitates a rigorous evaluation of our model's generalization capabilities. To ensure a fair assessment, all training sequences are strictly disjoint from our test set, as detailed in Table 3.

8.2. Testset

As detailed in Table 3, we provide a comprehensive analysis of the datasets used in our evaluation. We distinguish these datasets as either Out-of-Distribution (OOD) or In-Distribution (ID). A dataset is considered **OOD** if our model was not exposed to semantically similar data during training. In our benchmark, this includes `GS`, `VGGT`, `S3DIS`, `RS`, and `Random`. Conversely, a dataset is considered **ID** when the test data is of the same type as the training data. For instance, for `KITTI`, we train on sequences 00-10 and test on the first 100 frames of sequences 11-21. Based on characteristics such as point distribution, bit-depth or quantization step, and scale, we categorize all datasets into three levels of compression difficulty. For the `KITTI` and `Ford` sequences, we employ a two-step quantization process. First, the original coordinates are quantized to a

Table 3. Detailed description of the test dataset.

Dataset	Bit-Depth	Frames / Seq	Total Frames	Cond.	OOD	Avg. Points	Seq*
8iVFB	10-bit	300	600	Easy	✗	760375	Loot Red
MVUB	10-bit	/	461	Easy	✗	1259792	Phil Ricardo
Owlii	11-bit	600	2400	Easy	✗	2635971	-
Thuman	10-bit	/	20	Easy	✗	441319	0500-0519
ScanNet	2cm	100	100	Easy	✗	126760	scan_test
KITTI	1mm	100	1100	Easy	✗	119826	11-21
Ford	1mm	100	200	Easy	✗	83061	02-03
Dense_vox12	12-bit	/	2	Medium	✗	3222415	-
Sparse_vox12	12-bit	/	3	Medium	✗	996846	-
GS	1mm	/	13	Medium	✗	288413	-
VGGT	1mm	/	50	Medium	✓	257467	-
S3DIS	16-bit	/	1	Medium	✓	47062002	Stanford_Area_2
RS	10-12bit	/	16	Hard	✓	1058013	-
NS	10-12bit	/	15	Hard	✓	1670503	-
CS	10-12bit	/	14	Hard	✓	1531537	-

* Seq refers to the specific sequence used in the test, and - means that all sequences of the data set are used.

1mm precision by dividing them by 0.001. Second, these integer coordinates are further scaled by a quantization parameter Q=16, a setting aligned with previous work [18], before lossless compression. For all other datasets, this second scaling step is omitted (i.e., Q=1).

Easy Difficulty Datasets. This category includes several benchmarks widely adopted in the PCC community: `8iVFB`, `MVUB`, `Owlii`, `ScanNet`, `KITTI`, and `Ford`. These datasets predominantly feature either dense or sparse distributions. This allows learning-based methods to achieve strong performance by specializing in a specific data distribution. We also include `Thuman`, a mesh dataset that we converted to 10-bit point clouds through sampling and quantization. Given its distribution similarity to other human-centric datasets, we classify its compression difficulty as easy.

Medium Difficulty Datasets. This group comprises datasets with greater scale and complexity. `Dense-vox12` and `Sparse-vox12` are standard test sets defined by MPEG AI-PCC group. Additionally, `S3DIS` is a large-scale indoor scene dataset with point counts reaching tens of millions. The higher point counts and bit-depth of these datasets justify their classification as medium difficulty. Furthermore, we introduce two specialized datasets into this category. The first is our custom `VGGT` dataset, generated by reconstructing point clouds from 50 multi-view images (resized to 518x518) using the VGGT [14] algorithm. The second is the Gaussian Splatting (`GS`) dataset, which is created by extracting geometric coordinates from a pre-trained 3DGS model, following the methodology in [7] and [16]. Both `VGGT` and `GS` point clouds exhibit a unique characteristic of being locally dense yet globally sparse, distinguishing them from conventional datasets and thus warranting a medium difficulty rating.

Hard Difficulty Datasets. To evaluate the robustness of Point Cloud Compression (PCC) algorithms under challenging conditions, we systematically construct three distinct datasets by applying controlled perturbations to source point clouds from the `8iVFB`, `Owlii`, and `Dense-vox12` collections. These datasets, designated as Random Sampling (`RS`), Noise-added (`NS`), and Complex Sinusoidal (`CS`), are designed to disrupt the intrinsic structural regularities of point clouds. They serve as out-of-distribution (OOD) benchmarks to probe the generalization capabilities of PCC models and simulate real-world corruptions like sensor noise, packet loss, or non-rigid deformations.

- **Random Sampling (RS).** The `RS` dataset is generated via a spatially-dependent probabilistic downsampling. For each point $\mathbf{p}_i = (x_i, y_i, z_i)$ in the original cloud $\mathcal{P}_{\text{ori}}$, we first normalize its coordinates to a unit cube $[0, 1]^3$ based on the cloud's bounding box, yielding $\hat{\mathbf{p}}_i$. A retention probability $P(\hat{\mathbf{p}}_i)$ is then computed using a composite trigonometric function:

$$P(\hat{\mathbf{p}}_i) = \frac{1}{6}\Big(\sin(2\pi f_x \hat{x}_i + \phi_x) + \cos(2\pi f_y \hat{y}_i + \phi_y) + \sin(2\pi f_z \hat{z}_i + \phi_z) + 3\Big), \tag{11}$$

where $\{f_x, f_y, f_z\}$ are frequency parameters and

Table 4. Performance comparison with SparsePCGC-U and RENO-U. The table presents the compression performance of AnyPcc against 2 methods across 7 datasets, with the best and second-best results highlighted in red and yellow cells.

Dataset	RENO	RENO-U	SparsePCGC	SparsePCGC-U	Ours	Ours-U	GPCC
8iVFB	0.70	0.76	0.57	0.58	0.54	0.57	0.76
MVUB	1.00	1.07	0.69	0.73	0.67	0.75	0.94
Owlii	0.59	0.61	0.48	0.49	0.47	0.47	0.59
Thuman	1.64	1.72	1.70	1.63	1.58	1.64	2.00
ScanNet	2.15	2.26	1.86	1.90	1.83	1.88	2.03
KITTI	7.06	7.36	6.80	7.43	6.18	6.45	8.17
Ford	9.38	9.60	9.77	10.20	8.40	8.57	10.32
Gain over GPCC	-5.17%	-0.10%	-16.48%	-13.96%	-21.76%	-18.45%	0.00%
Enc/Dec Times	0.19/0.20		2.2/2.1		0.39/0.40		3.3/2.4

$\{\phi_x, \phi_y, \phi_z\}$ are phase shifts. A point is retained if a random variable $r_i \sim \mathcal{U}(0,1)$ satisfies $r_i \leq P(\hat{\mathbf{p}}_i)$. This process introduces non-uniform density variations, creating a sparsely sampled point cloud $\mathcal{P}_{\text{RS}}$.

- **Noise-added (NS).** The `NS` dataset is formed by corrupting point coordinates with Additive White Gaussian Noise (AWGN). The noise variance is adaptively scaled with the point cloud's spatial extent. The transformation for each point $\mathbf{p}_i \in \mathcal{P}_{\text{ori}}$ is:

$$\mathbf{p}'_i = \mathbf{p}_i + \boldsymbol{\epsilon}_i, \quad \text{where} \quad \boldsymbol{\epsilon}_i \sim \mathcal{N}\left(\mathbf{0}, \sigma^2 \mathbf{I}_3\right). \tag{12}$$

The standard deviation σ is proportional to the diagonal length of the original cloud's axis-aligned bounding box (AABB), controlled by a ratio parameter γ:

$$\sigma = \gamma \cdot \|\mathbf{p}_{\max} - \mathbf{p}_{\min}\|_2. \tag{13}$$

Crucially, following the noise injection, the coordinates of the resulting points are rounded to the nearest integers, and any duplicate points are removed. This simulates sensor quantization effects. The final point cloud is thus $\mathcal{P}_{\text{NS}} = \text{Unique}(\text{Round}(\{\mathbf{p}'_i\}))$.

- **Complex Sinusoidal (CS).** The `CS` dataset is created by applying a non-rigid deformation. This transformation, T_{deform}, first normalizes the input cloud $\mathcal{P}_{\text{ori}}$ to a unit cube, applies a sinusoidal deformation, and then rescales the result to the original bounding box dimensions. The core deformation is a composition of radial and per-axis sinusoidal scaling functions:

$$\mathcal{P}_{\text{CS}} = T_{\text{deform}}(\mathcal{P}_{\text{ori}}, \alpha), \tag{14}$$

$$\text{where } \mathbf{p}' = \mathbf{S}_{\text{coord}} \circ \mathbf{S}_{\text{radial}}(\hat{\mathbf{p}}), \tag{15}$$

$$\text{and } \mathbf{S}_{\text{radial}}(\hat{\mathbf{p}}) = \hat{\mathbf{p}} \cdot (1 + \alpha \sin(5\pi \|\hat{\mathbf{p}}\|_2)). \tag{16}$$

Here, $\hat{\mathbf{p}}$ is a normalized point, α controls the deformation degree, and $\mathbf{S}_{\text{coord}}$ represents a per-axis sinusoidal scaling. The resulting datasets, with their deliberately degraded and less predictable structures, present a significant challenge to existing compression algorithms, thus serving as robust benchmarks for evaluating model generalization under adverse conditions.

Summary. To our knowledge, we establish the most comprehensive benchmark for point cloud compression to date, designed to rigorously evaluate algorithms on two critical axes: generalization and stability. To assess generalization, our benchmark incorporates a wide data spectrum, ranging from standard objects and scenes to LiDAR scans and challenging point clouds from modern pipelines such as feed-forward reconstruction (`VGGT`) and neural representations (`GS`). For stability, we introduce demanding datasets with procedurally generated corruptions that stress-test algorithmic robustness against real-world imperfections. By systematically evaluating performance in these extensive scenarios, our benchmark aims to guide future research toward developing more versatile and resilient compression solutions.

9. More Experimental Results

9.1. Lossless Compression

Comparison with SparsePCGC-U and RENO-U. To facilitate a comparison with Unicorn-U [13], we conduct experiments using our unified training dataset. Since Unicorn is not yet open-source, we use its predecessor SparsePCGC [12] as our primary baseline. We retrain both RENO [18] and SparsePCGC [12] on our unified dataset, denoting these versions as RENO-U and SparsePCGC-U, respectively. The results are presented in Table 4, under strictly aligned training and testing conditions for all methods. The comparison is conducted on seven datasets where AnyPcc is evaluated without its Instance-Adaptive Fine-Tuning (IAFT) module to ensure a fair and direct comparison of the base models.

Table 5. Performance comparison on the dense point cloud Benchmark. The table presents the compression performance of AnyPcc against eleven methods across 2 datasets, with the best and second-best results highlighted in red and yellow cells.

Method	MPEG 8i				MVUB		Metrics
	Loot ×300	Red&black ×300	Boxer ×1	Thaidancer ×1	Phil ×245	Ricardo ×216	Ours-CR Gain
Non-Learning							
GPCC v23 [1]	0.82	0.69	0.65	0.70	0.96	0.89	-28.42%
Octree-Based							
OctAttention [3]	0.62	0.73	0.59	0.65	0.79	0.72	-18.54%
ECM-OPCC [6]	0.55	0.66	0.51	0.58	0.76	0.69	-10.71%
EHEM [11]	0.58	0.69	0.55	0.62	/	/	-17.37%
TopNet [17]	0.53	0.65	0.51	0.65	0.73	0.64	-9.67%
INR-Based							
LINR-PCGC [5]	0.57	0.69	/	/	0.84	0.80	-16.74%
LINR-PCGC-2 [5]	0.52	0.63	/	/	0.77	0.71	-8.25%
Voxel-Based							
VoxelDNN [10]	0.58	0.67	0.55	0.58	0.76	0.69	-12.73%
MsVoxelDNN [9]	0.73	0.87	0.70	0.85	1.02	0.95	-34.60%
UniPCGC [15]	0.49	0.59	0.45	0.51	0.80	0.73	-4.85%
SparsePCGC [12]	0.53	0.64	0.49	0.56	0.71	0.65	-6.55%
Unicorn [13]	0.53	0.64	0.49	0.56	0.71	0.65	-6.55%
RENO [18]	0.65	0.77	0.61	0.69	1.03	0.97	-28.26%
AnyPcc (Ours)	0.48	0.59	0.44	0.51	0.70	0.64	0.00%
AnyPcc-U (Ours-U)	0.52	0.62	0.48	0.54	0.78	0.72	-7.96%

Our results indicate that SparsePCGC-U exhibits more stable performance on object point clouds. We attribute this to SparsePCGC's use of a spatial auto-regressive eight-group context, which introduces a strong inductive bias well-suited for dense object data. In contrast, AnyPcc-U employs a four-group context that combines spatial and channel information, resulting in a weaker inductive bias and consequently superior performance in sparser scenarios. As discussed in Sec. 9.3, AnyPcc can be configured with additional spatial and channel groups to introduce a stronger inductive bias if needed. Furthermore, AnyPcc features a lightweight architecture, achieving over a 5x speedup in both encoding and decoding compared to SparsePCGC, while also delivering better performance.

In summary, training on a unified dataset leads to a slight degradation in performance on specific datasets, an observation consistent with the findings for Unicorn-U. However, this approach significantly enhances the model's generalization capabilities, which is a crucial step towards universal point cloud compression. Nevertheless, pre-training on large-scale data alone is insufficient for robust generalization to out-of-distribution (OOD) data, underscoring the necessity of the IAFT strategy proposed in this work.

Comparison in Dense Point Cloud Benchmark. To ensure a fair comparison with leading methods such as OctAttention [3], VoxelDNN [10], UniPCGC [15], and SparsePCGC [12] on human point clouds, we evaluate our method under their specific training and testing protocols, with results presented in Table 5. This protocol involves training on 10-bit versions of `Soldier` and `Longdress` from the `8iVFB` [2] dataset, and `Andrew`, `David`, and `Sarah` from the `MVUB` [8] dataset. The evaluation is performed on the `Loot`, `Redandblack`, `Phil`, `Ricardo`, `Boxer`, and `Thaidancer` sequences. Our comprehensive benchmark results are detailed in Table 5. We compare against a wide range of baselines: the traditional method GPCC v23 [1]; voxel-based deep learning methods including UniPCGC [15], SparsePCGC [12], RENO [18], VoxelDNN [10], and MsVoxelDNN [9]; octree-based methods like OctAttention [3], ECM-OPCC [6], EHEM [11], and TopNet [17]; and implicit compression methods such as LINR-PCGC [5] and LINR-PCGC-2 [5]. For human point cloud compression, Unicorn [13] employs the same methodology as SparsePCGC. Since Unicorn is not open-source, we report the performance of this method using the official SparsePCGC implementation. Note that LINR-

Table 6. Lossy Compression performance comparison on the AnyPcc Benchmark. The table presents the compression performance of AnyPcc against six methods across 8 † diverse datasets using BD-Rate. The best and second-best results highlighted in red and yellow cells.

Dataset	OOD	RENO	SparsePCGC	Unicorn*	OctAttention	TopNet	GPCC	Ours	Ours-U
8iVFB	✗	/	-20.29%	-20.29%	/	/	-94.39%	0.00%	-10.54%
MVUB	✗	/	-16.00%	-16.00%	/	/	-93.61%	0.00%	-12.13%
Owlii	✗	/	-23.70%	-23.70%	/	/	-93.76%	0.00%	-11.75%
Thuman	✗	/	-54.89%	-54.89%	/	/	-91.41%	0.00%	-16.54%
ScanNet	✗	/	-82.15%	-37.34%	/	/	-81.57%	0.00%	-2.96%
Ours BD-Rate Gains		/	-39.41%	-30.44%	/	/	-90.95%	0.00%	-10.78%
KITTI	✗	-12.87%	-4.49%	-3.62%	-10.17%	-4.48%	-22.45%	0.00%	-3.85%
Ford	✗	-11.08%	-6.79%	1.18%	-5.96%	-1.18%	-18.44%	0.00%	-0.99%
VGGT	✓	-8.13%	-2.33%	/	-8.33%	-3.65%	0.94%	0.00%	5.42%
Ours BD-Rate Gains		-10.69%	-4.54%	/	-8.15%	-3.10%	-13.31%	-0.00%	0.19%
Enc/Dec Time (s)		0.17/0.17	1.28/1.20	/	1.20/562	1.50/758	1.83/0.97	0.20/0.18	
Total Params (M)		9.03	27.03	/	29.61	23.59	/	104.44	14.92

† For lossy compression, our evaluation is focused on 8 datasets. The other 7 were excluded because competing methods produced unstable results, preventing a fair alignment of rate points. Since the lossy strategy for these datasets mirrors that of KITTI, their lossless performance trends offer a reliable indication of their likely lossy performance.

* The results for Unicorn are cited directly from the original publication as its implementation is not open-source.

PCGC-2 is a variant of LINR-PCGC with a longer fine-tuning duration, resulting in a higher encoding time.

- **Comparison to SparsePCGC.** Compared to SparsePCGC / Unicorn, AnyPcc is more lightweight as it introduces only four predictive groups, yet it delivers superior performance. Furthermore, AnyPcc can be extended with additional channel and spatial groups to potentially achieve even higher compression efficiency. The impact of our context modeling choices is further analyzed in the Sec. 9.3.
- **Comparison to UniPCGC.** It is noteworthy that UniPCGC builds upon SparsePCGC by introducing a non-uniform eight-group strategy. This approach is particularly effective for point clouds with a uniform spatial distribution. In the `8iVFB` dataset, UniPCGC achieves performance second only to AnyPcc. However, this strategy introduces a strong inductive bias by assuming that the initial groups can always provide sufficient prior information. This assumption does not hold for many real-world point clouds with non-uniform distributions, which explains its suboptimal performance on the `MVUB` dataset.
- **Comparison to LINR-PCGC.** Furthermore, the practical application of implicit methods like LINR-PCGC and LINR-PCGC-2 is limited by their substantial encoding times (ranging from 2 to 18 seconds) and their restriction to multi-frame point cloud sequences. Moreover, their compression efficiency is not state-of-the-art.
- **Summary.** We have aligned our evaluation with the established community standards for human point cloud compression and conducted a comprehensive benchmark. The results confirm that AnyPcc consistently achieves state-of-the-art performance across all tested conditions.

9.2. Lossy Compression

We present further experimental results for lossy compression in Table 6. Our method demonstrates superior performance, outperforming various learning-based approaches and the conventional G-PCC v23 codec. The table quantifies this advantage by reporting the Bjontegaard Delta Rate (BD-Rate) gains of our model ("Ours") over competitors. A lower BD-Rate value signifies a greater improvement. For instance, -90% indicates that our method achieves a 90% bitrate saving for the same PSNR quality on average. Furthermore, our encoding and decoding times are comparable to those of RENO, one of the fastest methods, underscoring AnyPcc's role as a lightweight yet unified solution for point cloud compression. Notably, our "Ours-U" variant utilizes a single model to perform both lossy and lossless compression across diverse point cloud types, which substantially enhances the practicality of deep learning-based PCC. Moreover, We provide point-to-point error heatmaps for the dense 8iVFB, Owlii and Thuman dataset in Figure 3. The error distribution visually confirms that AnyPcc preserves fine geometric details effectively.

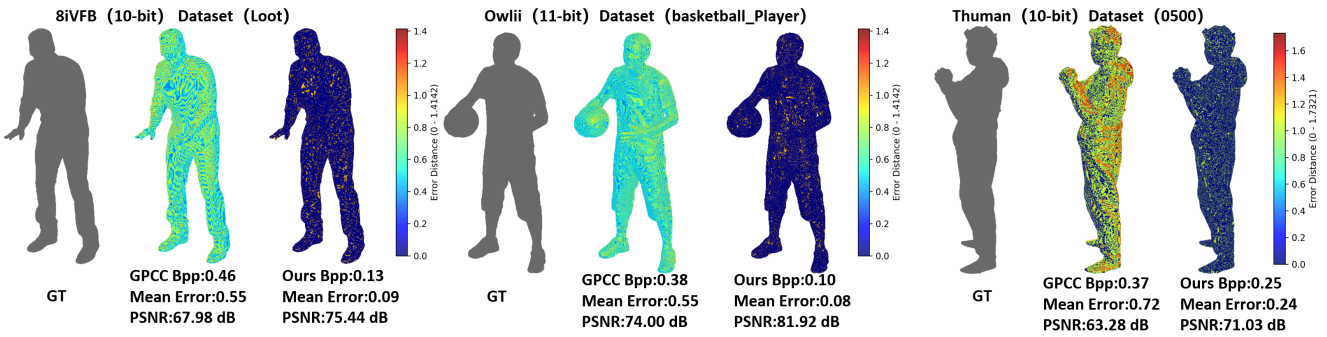

Figure 3. Point-to-point error heatmaps for the dense 8iVFB, Owlii and Thuman datasets.

9.3. Ablation Studies

Context Model. We provide a comprehensive ablation study of our context model in Table 7, investigating the impact of increasing the number of channel and spatial groups. Within our framework, we reimplement the core ideas of SparsePCGC [12] (uniform spatial grouping) and UniPCGC [15] (non-uniform spatial grouping). These correspond to our 8-group model in the Spatial1 experiment and our ue-4-group model in the Spatial2 experiment, respectively. It is important to note that these are conceptual replications and may differ from the original implementations. Our findings show that the non-uniform grouping of ue-4-group yields worse overall performance than uniform grouping, suggesting its applicability is limited to specific datasets. Furthermore, the uniform eight-group strategy of SparsePCGC is outperformed by our proposed hybrid approach (e.g., 2 spatial groups + multiple channel groups), demonstrating the superior efficiency of our context design.

Generally, when the number of groups in one dimension (either spatial or channel) is fixed, increasing the number of groups in the other dimension tends to improve performance. This is because a greater number of groups introduces a stronger inductive bias, which can simplify the learning process for the model. An interesting exception occurs when the number of channel groups is fixed at 2: increasing the spatial groups from 2 to 4 leads to a performance degradation on in-distribution datasets. We hypothesize that at the occupancy code scale, an excessive number of spatial groups creates too large a gap between contexts, diminishing the relevance of the reference information and thus hindering performance. Conversely, when starting from a single group in either dimension, increasing the group count consistently improves performance.

It is also crucial to consider the trade-off between performance and complexity. For spatial grouping, a higher number of groups necessitates more sequential auto-regressive decoding steps, which increases both encoding and decoding time. A similar trade-off exists for channel grouping. However, a special case arises with a single channel group: this requires the arithmetic coder to handle a 256-way categorical distribution at once, which is computationally expensive and slows down the coding process. This observation is consistent with findings in RENO. As the number of channel groups increases from 2 to 8, performance generally improves at the cost of increased computational complexity. Considering these trade-offs, a configuration with 2 spatial groups and multiple channel groups emerges as a favorable choice. For our final model, we select 2 spatial and 2 channel groups. We do not opt for 4 or 8 channel groups because they demand significantly more GPU memory for training and increase decoding complexity. In contrast, the 2-group configuration provides a more lightweight and efficient solution.

The analysis above considers performance averaged across all datasets. However, the detailed breakdown in Table 7 offers a novel insight: different context designs excel on different types of point clouds. This comprehensive ablation provides a valuable reference for future research, highlighting that the optimal context modeling strategy is likely data-dependent.

Fine-tuning Iteration and Module Sensitivity. In this subsection, we demonstrate the effectiveness of our Instance-Adaptive Fine-Tuning (IAFT) strategy, particularly for out-of-distribution (OOD) data. As illustrated in Figure 4, the points along the red curve represent the model's performance at various fine-tuning iterations, ranging from 10 to 1200. The results clearly show that the model converges most rapidly during the initial stages, yielding the most significant bitrate reduction. After approximately 200 iterations, the performance gains begin to saturate, with further improvements becoming marginal. We therefore select this as our default fine-tuning stopping point. This highlights a key advantage of our method: it offers a dynamic trade-off between encoding time and compression performance. In resource-constrained scenarios, one can opt for fewer iterations to achieve faster encoding, sacrificing a small amount of compression efficiency. Conversely, when computational resources are readily avail-

Table 7. Ablation study of the context model. The table is split into two main experiments: varying the number of groups for Spatial2 / Channel2 and Spatial1 / Channel1 models respectively.

Dataset	OOD	Spatial2 (Channel: 1 to 4)				Channel2 (Spatial: 1 to 4)		
		1-group	2-group	4-group	ue-4-group	1-group	2-group	4-group
8iVFB	✗	0.604	0.570	**0.563**	**0.563**	0.635	**0.570**	0.579
MVUB	✗	0.832	0.752	**0.744**	0.752	0.854	**0.752**	0.772
Owlii	✗	0.498	0.465	**0.456**	**0.456**	0.512	**0.465**	0.469
Thuman	✗	1.667	1.636	**1.633**	1.634	1.677	**1.636**	1.649
ScanNet	✗	1.922	**1.826**	1.857	1.849	2.062	**1.826**	1.877
KITTI	✗	6.511	6.453	6.371	**6.353**	6.877	**6.453**	6.489
Ford	✗	8.632	8.565	8.459	**8.452**	9.106	**8.565**	8.599
Dense	✗	6.201	**6.179**	6.194	6.195	6.275	**6.179**	6.217
Sparse	✗	9.710	**9.659**	9.690	9.731	9.728	**9.659**	9.680
GS	✗	13.393	**13.304**	13.333	13.330	13.647	13.304	**13.248**
VGGT	✓	7.310	7.222	**7.218**	7.250	7.320	**7.222**	7.230
S3DIS	✓	11.454	11.317	11.166	**11.163**	11.772	11.317	**11.296**
RS	✓	3.674	3.643	**3.637**	3.642	3.692	3.643	**3.625**
NS	✓	4.887	4.863	**4.840**	4.865	4.883	4.863	**4.850**
CS	✓	3.405	3.347	**3.308**	3.335	3.548	3.347	**3.322**
Avg bpp	–	5.380	5.320	**5.298**	5.305	5.506	**5.320**	5.327
Enc Time	–	0.492	**0.436**	0.571	0.568	**0.382**	0.436	0.539
Dec Time	–	0.507	**0.445**	0.584	0.581	**0.391**	0.445	0.551

Dataset	OOD	Spatial1 (Channel: 1 to 8)				Channel1 (Spatial: 1 to 4)		
		1-group	2-group	4-group	8-group	1-group	2-group	4-group
8iVFB	✗	0.731	0.635	0.595	**0.565**	0.731	0.604	**0.581**
MVUB	✗	1.045	0.854	0.796	**0.733**	1.045	0.832	**0.782**
Owlii	✗	0.593	0.512	0.482	**0.460**	0.593	0.498	**0.473**
Thuman	✗	1.767	1.677	1.652	**1.625**	1.767	1.667	**1.646**
ScanNet	✗	2.196	2.062	1.986	**1.909**	2.196	1.922	**1.878**
KITTI	✗	7.048	6.877	6.571	**6.390**	7.048	6.511	**6.484**
Ford	✗	9.268	9.106	8.704	**8.483**	9.268	8.632	**8.599**
Dense	✗	6.436	6.275	6.314	**6.217**	6.436	6.201	**6.197**
Sparse	✗	9.810	9.728	9.804	**9.794**	9.810	9.710	**9.675**
GS	✗	13.839	13.647	13.650	**13.570**	13.839	13.393	**13.297**
VGGT	✓	7.529	7.320	7.320	**7.265**	7.529	7.310	**7.241**
S3DIS	✓	12.094	11.772	11.441	**11.220**	12.094	11.454	**11.312**
RS	✓	3.799	3.692	3.681	**3.659**	3.799	3.674	**3.629**
NS	✓	4.952	**4.883**	4.892	4.932	4.952	4.887	**4.835**
CS	✓	3.740	3.548	3.481	**3.411**	3.740	3.405	**3.326**
Avg bpp	–	5.656	5.506	5.425	**5.349**	5.656	5.380	**5.330**
Enc Time	–	0.436	**0.382**	0.496	0.724	**0.436**	0.492	0.554
Dec Time	–	0.451	**0.391**	0.507	0.742	**0.451**	0.507	0.568

able, a longer fine-tuning duration can be used to maximize compression performance. Our framework thus enables a controllable complexity-performance trade-off. Importantly, even with zero fine-tuning, our base model already outperforms several state-of-the-art methods on many datasets, underscoring the strong foundational performance of our approach.

Moreover, We performed ablation studies on IAFT, as shown in Figure 5. Just **10 steps** of IAFT yield significant gains on OOD data, confirming robustness under low latency constraints. Extending fine-tuning to Spatial Convolutions incurs a **15× time increase** and higher weight over-

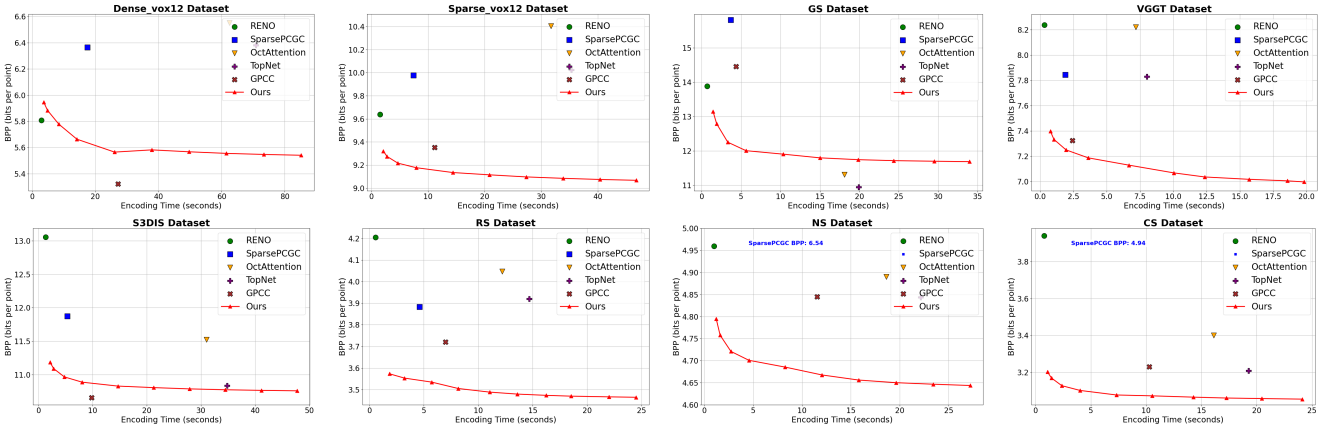

Figure 4. Trade-off between encoding complexity and performance. Our method allows for a flexible balance between these two factors. On the NS and CS datasets, the BPP value for SparsePCGC is annotated directly due to its significantly poor performance, ensuring visual clarity.

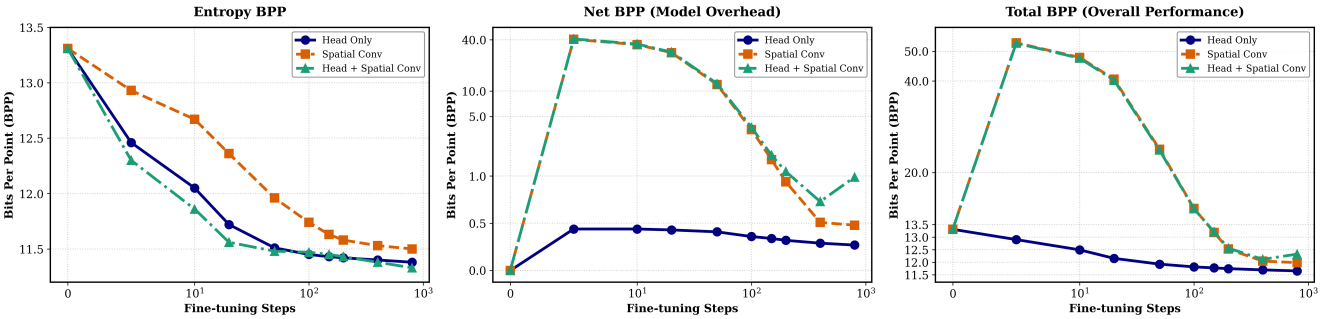

Figure 5. Impact of finetune steps and module sensitivity in GS dataset.

head without proportional coding gains. Thus, fine-tuning only the lightweight head offers the optimal trade-off.

Quantization & Coder Selection. To efficiently compress the network weights transmitted to the decoder, we employ uniform scalar quantization with a step size $\Delta = 2^{-0.5 \times q}$. As illustrated in Figure 6, varying the quantization parameter q reveals a clear trade-off: larger q values preserve finer weight details, which effectively lowers the point cloud's entropy coding bitrate but simultaneously increases the weight payload. Consequently, the Total BPP exhibits a stable, gentle U-shaped trend, reaching an optimal balance at $q = 16$ across all tested coders. Crucially, all quantization settings consistently outperform the pre-trained model baseline. Furthermore, we observe that our method is highly robust to the choice of weight coders. At the optimal $q16$ setting, a standard `GZIP` coder achieves a Total BPP of 11.93, which closely matches the 11.74 BPP achieved by the computationally heavier `DeepCABAC`. This demonstrates that the significant coding gains primarily derive from our proposed strategy itself, rather than relying on complex parameter entropy coders.

Training Data Ratio. We train the Ours-U model on a mixed dataset comprising `8iVFB`, `Thuman`, `ScanNet`, `KITTI`, `Ford`, and a general `GS` dataset. In this section, we investigate the impact of the training data mixing ratio on the final model performance. We acknowledge that determining an analytically optimal mixing ratio is intractable due to the infinite solution space. Therefore, our objective is to identify an empirically effective and well-balanced configuration. To this end, we systematically adjust the sampling weight of key subsets within the total training data, generating multiple training configurations. The performance of each configuration is then quantitatively analyzed by evaluating the resulting model's bitrate on both the specific target dataset and our comprehensive test suite (denoted "All Datasets").

We begin with a baseline weight configuration, denoted as w_1 : {`8iVFB`: 2.5, `Thuman`: 1.0, `ScanNet`: 1.0, `KITTI`: 2.5, `Ford`: 1.0, `GS`: 2.0}. We note that during training, these weights are normalized to form a probability distribution from which datasets are randomly sampled for each training batch. As Figure 7 depicts, w_1 represents this initial configuration. The subsequent configurations, w_2 to

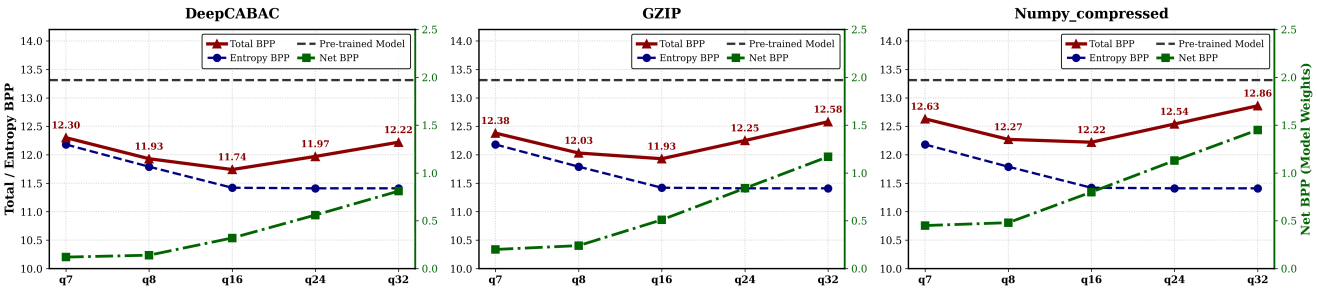

Figure 6. Impact of weight quantization and coder selection in GS dataset.

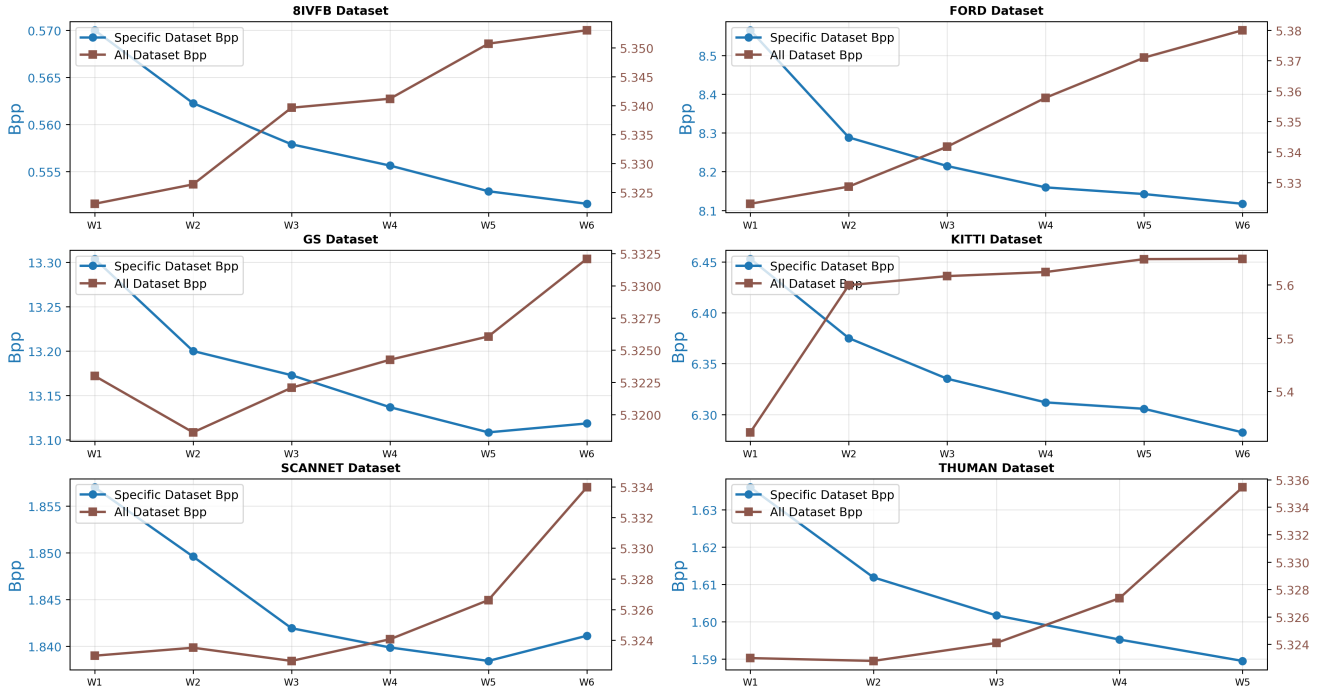

Figure 7. A comparative analysis of compression performance on the Specific Dataset and the All Dataset as a function of training set composition. The w_1 model is trained on a balanced dataset. For models w_2 and beyond, the proportion of the specific subset in the training data is systematically increased. The blue and orange-brown lines illustrate the model's performance on the Specific Dataset and the All Dataset, respectively.

w_6, are generated by incrementally increasing the weight of the target dataset by 2.0 for each step. The results illustrate a clear trend: as the sampling weight of a specific dataset increases, the model's performance on that particular dataset improves. However, this is often accompanied by a decrease in performance on the comprehensive benchmark. This outcome validates our initial choice of w_1 as a reasonable and well-balanced configuration for general-purpose compression.

9.4. Practical Deployment and Latency Analysis.

It is important to note that the encoding latency of 2.84s reported in Table 1 of main paper represents the extreme upper bound of our framework, corresponding to the maximum fine-tuning setting (IAFT-200) designed strictly for peak rate-distortion performance. In real-world deployment, our approach offers high flexibility to balance efficiency and coding gain: (1) *Standard Benchmarks:* For standard autonomous driving scenarios like the KITTI dataset, our pre-trained model exhibits strong generalization capabilities. It achieves a superior 6.18 BPP compared to RENO's 7.06 BPP without any instance-specific fine-tuning. Under this configuration, the encoding latency is merely 0.28s, which fully satisfies the stringent low-latency requirements of practical applications. (2) *Favorable Efficiency Trade-off:* The proposed IAFT module converges

rapidly. By limiting the fine-tuning process to just 10 iterations, the latency on the dense GS dataset drastically drops to 1.35s, effectively lowering the overall average encoding time across all datasets to a highly competitive **0.56s**. Even with this accelerated configuration, our method consistently maintains a clear performance margin over existing baselines. (3) *Standardization Compatibility:* From a system-level perspective, the updated IAFT weights are highly compact and seamlessly transmitted as standard side information. This architectural design aligns perfectly with existing point cloud compression standards (e.g., MPEG and AVS pipelines), ensuring straightforward integration into established industrial systems.

10. Point Cloud Visualization

In this section, we visualize the 15 datasets used in our experiments, as depicted in Figure 8. Each point cloud is color-coded by its local point density, transitioning from blue (sparse) to red (dense). The visualization clearly shows the diverse geometric structures in the benchmark: each category has a unique density profile, and challenging datasets show highly non-uniform distributions. Our unified model (Ours-U) successfully compresses all these structurally heterogeneous point clouds, which is a strong testament to its stability and generalization.

11. Future Work

Hybrid Explicit-Implicit Compression. AnyPcc demonstrates a powerful principle: fine-tuning a small subset of network parameters for each instance can yield significant compression gains. This approach becomes advantageous whenever the bitrate cost of transmitting the updated weights is less than the resulting bitrate savings in the geometry. This hybrid paradigm, which synergizes the strengths of explicit (entropy coding) and implicit (network-based representation) compression, can be readily extended to other domains such as point cloud attribute compression, as well as image and video compression. It offers the unique benefit of adjustable encoding complexity through control of the fine-tuning process. Furthermore, by integrating recent advances in implicit neural representations and leveraging parameter-efficient fine-tuning techniques from the large model domain, such as LoRA [4], we can develop even more efficient and versatile compression schemes.

Scaling Laws of Compression Models. Our ablation study, which explores channel counts from 32 to 128, reveals a clear trend of improved compression performance with increased model size, suggesting the potential for a scaling law. Drawing inspiration from the success of Large Language Models (LLMs), a promising future direction is to systematically scale up the AnyPcc architecture (e.g., to one billion parameters or beyond) to investigate the ultimate limits of neural point cloud compression.

Self-Supervised Learning and Data Curation. At its core, point cloud compression is a self-supervised learning task where the model learns to predict the data from itself. As we scale these models, the availability of vast and diverse datasets becomes paramount. Future work should therefore focus on two key areas: curating larger and more comprehensive collections of real-world point cloud data, and leveraging powerful generative models to synthesize high-quality training data. These efforts will be critical for successfully scaling self-supervised compression models and pushing the boundaries of generalization.

References

[1] Chao Cao, Marius Preda, Vladyslav Zakharchenko, Euee S Jang, and Titus Zaharia. Compression of sparse and dense dynamic point clouds—methods and standards. *Proceedings of the IEEE*, 109(9):1537–1558, 2021. 7

[2] Eugene d'Eon, Harrison Bob, Taos Myers, and Philip A. Chou. 8i voxelized full bodies - a voxelized point cloud dataset. In *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006*, 2017. 7

[3] Chunyang Fu, Ge Li, Rui Song, Wei Gao, and Shan Liu. Octattention: Octree-based large-scale contexts model for point cloud compression. In *Proceedings of the AAAI conference on artificial intelligence*, pages 625–633, 2022. 7

[4] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022. 13

[5] Wenjie Huang, Qi Yang, Shuting Xia, He Huang, Zhu Li, and Yiling Xu. Linr-pcgc: Lossless implicit neural representations for point cloud geometry compression. *arXiv preprint arXiv:2507.15686*, 2025. 7

[6] Yiqi Jin, Ziyu Zhu, Tongda Xu, Yuhuan Lin, and Yan Wang. Ecm-opcc: Efficient context model for octree-based point cloud compression. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7985–7989. IEEE, 2024. 7

[7] 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. 5

[8] Charles Loop, Qin Cai, Sergio Orts Escolano, and Philip A Chou. Jpeg pleno database: Microsoft voxelized upper bodies-a voxelized point cloud dataset. *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document m38673 M*, 72012, 2021. 7

[9] Dat Thanh Nguyen, Maurice Quach, Giuseppe Valenzise, and Pierre Duhamel. Multiscale deep context modeling for lossless point cloud geometry compression. *arXiv preprint arXiv:2104.09859*, 2021. 7

[10] Dat Thanh Nguyen, Maurice Quach, Giuseppe Valenzise, and Pierre Duhamel. Learning-based lossless compression

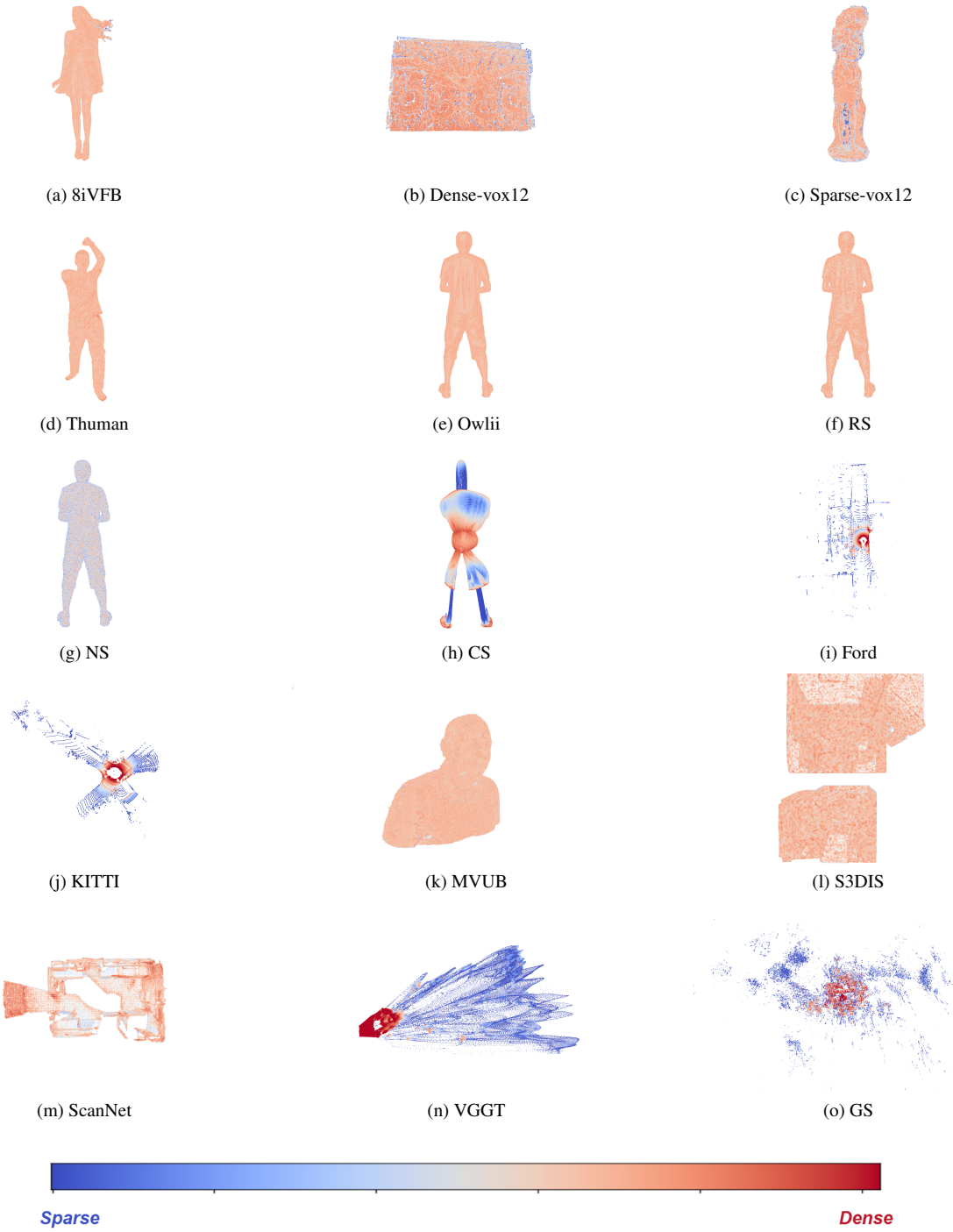

Figure 8. Visualization of 15 different point cloud datasets used in our experiments.

of 3d point cloud geometry. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4220–4224. IEEE, 2021. 7

[11] Rui Song, Chunyang Fu, Shan Liu, and Ge Li. Efficient hierarchical entropy model for learned point cloud compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14368–14377, 2023. 7

[12] Jianqiang Wang, Dandan Ding, Zhu Li, Xiaoxing Feng, Chuntong Cao, and Zhan Ma. Sparse tensor-based multiscale representation for point cloud geometry compression. *IEEE Transactions on Pattern Analysis and Machine Intelli-*

gence, 2022. 2, 3, 6, 7, 9

[13] Jianqiang Wang, Ruixiang Xue, Jiaxin Li, Dandan Ding, Yi Lin, and Zhan Ma. A versatile point cloud compressor using universal multiscale conditional coding–part i: Geometry. *IEEE transactions on pattern analysis and machine intelligence*, 2024. 6, 7

[14] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025. 5

[15] Kangli Wang and Wei Gao. Unipcgc: Towards practical point cloud geometry compression via an efficient unified approach. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(12):12721–12729, 2025. 2, 7, 9

[16] Kangli Wang, Shihao Li, Qianxi Yi, and Wei Gao. A novel benchmark and dataset for efficient 3d gaussian splatting with gaussian point cloud compression. *arXiv preprint arXiv:2505.18197*, 2025. 4, 5

[17] Xinjie Wang, Yifan Zhang, Ting Liu, Xinpu Liu, Ke Xu, Jianwei Wan, Yulan Guo, and Hanyun Wang. Topnet: Transformer-efficient occupancy prediction network for octree-structured point cloud geometry compression. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pages 27305–27314, 2025. 7

[18] Kang You, Tong Chen, Dandan Ding, M Salman Asif, and Zhan Ma. Reno: Real-time neural compression for 3d lidar point clouds. *arXiv preprint arXiv:2503.12382*, 2025. 2, 5, 6, 7