DeepPCC: Learned Lossy Point Cloud Compression

Junzhe Zhang, Gexin Liu, Junteng Zhang, Dandan Ding, and Zhan Ma

***Abstract*—We propose the DeepPCC, an end-to-end learning-based approach for the lossy compression of large-scale object point clouds. For both geometry and attribute components, we introduce the Multiscale Neighborhood Information Aggregation (NIA) mechanism, which applies resolution downscaling progressively (*i.e.*, dyadic downsampling of geometry and average pooling of attribute) and combines sparse convolution and local self-attention at each resolution scale for effective feature representation. Under a simple autoencoder structure, scale-wise NIA blocks are stacked as the analysis and synthesis transform in the encoder-decoder pair to best characterize spatial neighbors for accurate probability approximation of geometry occupancy and attribute intensity for point cloud compression (PCC). Experiments demonstrate that the DeepPCC remarkably outperforms state-of-the-art rules-based MPEG G-PCC and learning-based JPEG VM both quantitatively and qualitatively, evidencing that DeepPCC is a promising solution for emerging AI-based PCC. The source code will be made publicly accessible soon.**

***Index Terms*—Point cloud compression, geometry, attribute, sparse convolution, local self-attention**

I. Introduction

Large-scale point clouds have emerged as a prevalent format to realistically represent 3D objects in various applications, including AR/VR (Augmented Reality/Virtual Reality), Digital Twin, *etc.*, for an immersive experience. Typically, a large-scale point cloud sample comprises millions of unordered and sparsely distributed 3D points, with each point usually characterized by its geometry coordinate, *e.g.*, (x, y, z) in a Cartesian coordinate system, and attribute components such as RGB color, normal, and reflectance. For a color point cloud frame that has 1 million points and uses 8-bit precision to represent both geometry and RGB attributes of each point, it requires about 6 MB data size per frame, which apparently cannot be sustained reliably for a variety of access networks used in practices [2]. This thus urges the development of high-efficiency Point Cloud Compression (PCC) solution for uncompromised service provisioning [3].

A. Background and Motivation

The lossy compression of point clouds is a classical rate-distortion (R-D) optimization problem in information theory [4], where the goal is to obtain a compact representation of the input sample with the least bit rate consumption under a given distortion metric, *e.g.*, MSE (Mean Square Error) or MS-SSIM (Multiscale Structural Similarity [5]). In principle, the derivation of this compact representation of an input point cloud is mainly determined by how to properly organize its data to sufficiently exploit spatial (and/or temporal) correlations for redundancy reduction.

Given the unordered and sparse nature of point clouds, an enormous amount of effort has been paid over the past two decades to effectively organize spatial neighborhoods for correlation exploration. Such neighborhoods could be formed through the use of 1D transversal, 2D projection, or 3D space indexing. A number of pioneering explorations have been developed, as reviewed in [2], leading to the establishment of two international standards for PCC under the MPEG committee in 2017: Video-based PCC (V-PCC) and Geometry-based PCC (G-PCC) [3], [6], [7].

V-PCC first projects a 3D point cloud frame onto a set of perpendicular 2D planes [8], and then employs conventional video coding standards [9], [10] to process the resulting sequence of projected 2D images. In contrast, G-PCC separates the compression of geometry and attribute information, using a 3D octree model or a trisoup model to encode the geometry information and employing Region-Adaptive Hierarchical Transform (RAHT) [11] or hierarchical nearest neighbor prediction-based Lifting Transform (Predlift) to represent the attributes (such as RGB colors) on top of the restored geometry.

Owing to the advances of deep learning technologies, recent years have witnessed the exponential growth of learning-based PCC solutions [12]. More importantly, both MPEG and JPEG committees have launched explorations of AI (Artificial Intelligence) based PCC, with the aim of establishing it as the next-generation standard [13], [14]. It is worth pointing out that the JPEG committee has released the very first learning-based Verification Model (VM), leading the standardization development of AI-based PCC for the future[1].

This work proposes DeepPCC, a new learning-based lossy PCC approach for the compression of large-scale object point clouds.

B. Our Approach and Contribution

Unlike existing works that focus on either geometry or attribute information compression [12], the proposed DeepPCC can process both geometry and attribute components of an input point cloud. Following the conventions used in MPEG G-PCC standard [15], we first compress the geometry, and then the attribute components such as colors, assuming full knowledge of the corresponding geometry. Voxelized point clouds are utilized in DeepPCC, which is consistent with the processing of MPEG G-PCC and JPEG VM. However, the point-based compression method without voxelizing raw points is beyond the scope of this work; interested readers can refer to [16]–[21] for more details.

J. Zhang, G. Liu, J. Zhang, and D. Ding are with the School of Information Science and Technology, Hangzhou Normal University, Hangzhou, Zhejiang 311121, China. Email: DandanDing@hznu.edu.cn

Z. Ma is with the School of Electronic Science and Engineering, Nanjing University, Nanjing, Jiangsu, 210093 China. Email: mazhan@nju.edu.cn

[1] https://jpeg.org/jpegpleno/pointcloud.html

Fig. 1: **Visual Reconstructions** of point clouds compressed using DeepPCC, MPEG G-PCC, and JPEG VM. Both geometry and color attributes are lossy compressed. Two G-PCC models, *e.g.*, TMC13v6 and TMC13v19, and the latest JPEG VM are used. Bitrates are made as close as possible to have a fair comparison. However, as these methods do not have rate control yet, bitrate mismatch happens inevitably. Yet, the proposed DeepPCC shows clear gains for both quantitative and qualitative measures. bpp - bits per point; PSNR: Peak-to-signal-noise ratio; D1: point-to-point PSNR; Y: Luma PSNR; PCQM: a full-reference quality metric for colored point cloud [1]. *Please zoom in for more details.*

In theory, the efficiency of PCC highly depends on the effectiveness of exploiting inter-point or inter-voxel correlations[2], which is challenging due to the unordered and sparse nature of 3D points. To address this challenge, we first assume the use of a uniform voxel representation model for 3D position indexing [22]. For a given point, we can easily access its neighbors in a $n \times n \times n$ cube centered at it. Straightforwardly, dense 3D convolutions can be applied on each point to aggregate and embed spatial neighbors for compact representation, as studied in [23], [24], which however introduces unbearable space and time complexity due to the uniform computations on every voxel regardless of its occupancy state. Although block partition or patch slicing methods can be utilized to reduce the complexity to some extent, their performance is often constrained to local optima [25].

Instead, we propose to apply sparse convolution [26] and local self-attention [27] to strictly limit the computations only to positively occupied voxels (POV) in close proximity, which reduces complexity significantly and enables us to process the entire large-scale point cloud frame directly. The proposed local self-attention performs computations only among K nearest POV neighbors (KNN) of a specific POV, which thus is referred to as the KNN self-attention for convenience.

In DeepPCC, we take advantage of sparse convolution and KNN self-attention by either directly stacking them or allocating them in parallel branches to achieve efficient Neighborhood Information Aggregation (NIA). Since the POVs in a point cloud frame are sparsely distributed, it is highly likely that only a limited number of POVs will fall within the fixed-size receptive field of sparse convolution, which may not provide sufficient information to accurately characterize neighborhood correlations for compression. In such cases, KNN self-attention can gather K neighboring POVs without being limited by the fixed-size receptive field, providing the necessary information for NIA. Yet, using a smaller K may not capture sufficient POVs nearby, while using a larger K could lead to a sharp increase in computational complexity. Therefore, to balance the complexity and efficiency, we often set a reasonable and fixed value of K. To further enhance information embedding capacity, we propose to progressively downscale the point cloud (*e.g.*, 3 times in this work), and at each resolution scale, the NIA block is incorporated.

As illustrated in Fig. 2, the DeepPCC adopts a simple autoencoder structure, in which it stacks scale-wise NIA blocks to form respective analysis and synthesis transform in paired encoder and decoder. To help the audience understand the DeepPCC better, we draw two parallel branches: one branch for the progressive downscaling[3] to generate multi-

[2]After the voxelization, raw points are mapped to positively-occupied voxels (POV). For simplicity, we may interchangeably use the term "point" and "POV".

[3]As will be detailed in subsequent sections, dyadic downsampling at each axis in Cartesian coordinate system is applied for the geometry component, while average pooling is used for corresponding attributes.

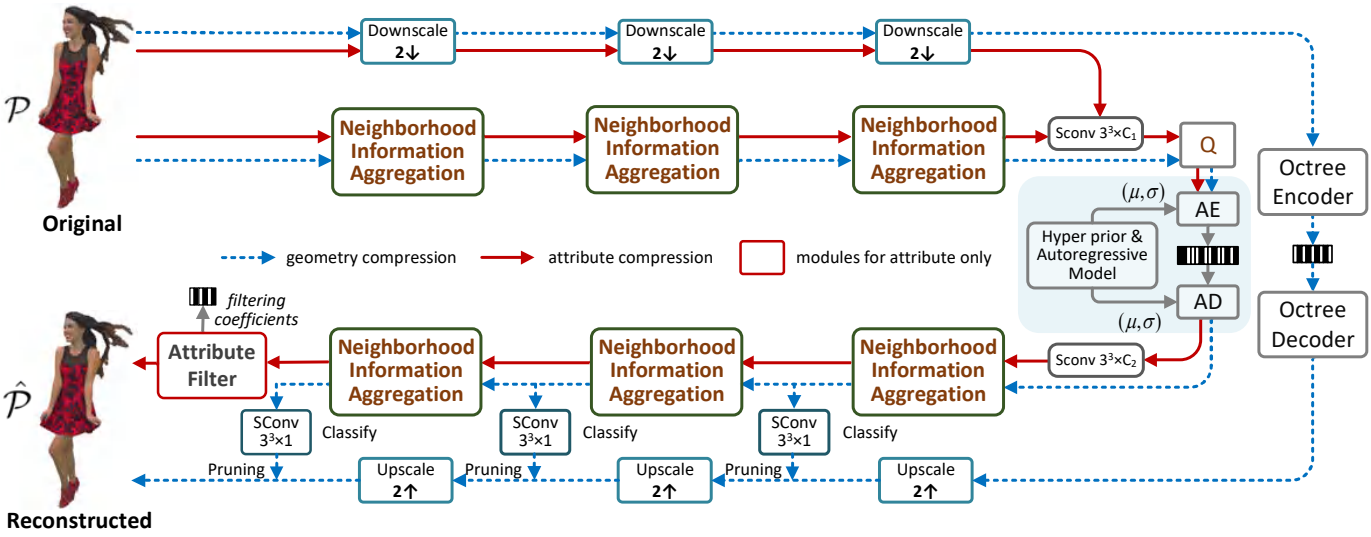


Fig. 2: **DeepPCC** used for lossy point cloud compression. For both geometry compression (indicated by the blue dotted lines) and attribute compression (indicated by the red solid lines), multiscale NIA blocks are devised to best characterize neighborhood correlation for geometry- and attribute-related latent features in encoding. In decoding, geometry occupancy is scale-wisely decoded and refined using lower-scale information, while the attribute features are decoded to the final resolution in one shot. An Attribute Filter is additionally introduced to enhance the quality of attribute reconstruction. SConv $k^3 \times c$: Sparse convolution with $k \times k \times k$ kernel and c channels output; $s\downarrow/s\uparrow$: downscaling/upscaling with a ratio of s at $x-$, $y-$, and $z-$axis; (μ, σ): learned entropy parameters.

resolution point clouds; and the other branch for applying the corresponding NIA in feature domain at each resolution scale.

In lossy geometry encoding, the geometry coordinates of POVs at the lowest scale (after downsampling) are losslessly coded using the G-PCC coder for simplicity, while the corresponding latent features at the bottleneck are quantized and encoded using adaptive contexts. In the lossy attribute encoding, the latent features are also encoded using adaptive contexts, given losslessly or lossy compressed geometry priors. During the decoding process, the geometry occupancy is gradually refined from the lower scale to the higher scale. Such a scale-wise geometry reconstruction and refinement not only provides better quality but also significantly reduces the space and time complexity [25]. On the other hand, the attribute features are decoded to the final resolution in one shot. To alleviate the compression noise, an Attribute Filter is applied at the end to enhance the reconstruction quality.

Currently, the proposed DeepPCC supports three modes: DeepPCGC for lossy geometry compression only, DeepPCAC for lossy attribute compression with losslessly compressed geometry, and DeepPCC for lossy compression of both geometry and attribute under the same framework shown in Fig. 2.

The main contributions of this paper are summarized below:

- The proposed DeepPCC is a simple yet efficient learning-based lossy PCC approach that can process both geometry and attribute components under a unified framework, offering a competitive alternative solution to existing rules-based MPEG G-PCC and learning-based JPEG VM solutions;
- DeepPCC demonstrates significant performance gains over both MPEG G-PCC and JPEG VM across a variety of testing conditions. For the lossy compression of both geometry and attribute information, DeepPCC offers more than 60% BD-BR gains in comparison to the G-PCC (TMC13v19) when measuring the overall distortion using PCQM (see Table VI); Similar gains are reported when using the JPEG VM as the anchor. Qualitatively, DeepPCC produces more visually-pleasing reconstructions (see Fig. 1).
- The superior performance of DeepPCC comes from the utilization of multiscale neighborhood information aggregation, which enables the efficient characterization of spatial correlations for compact representation. Moreover, using sparse convolution and KNN self-attention limits the computation of DeepPCC to positively-occupied voxels only, making it attractive to practitioners due to its affordable complexity.

TABLE I: Notations

Abbreviation	Description
PCC	Point Cloud Compression
PCG	Point Cloud Geometry
PCA	Point Cloud Attribute
POV	Positively-Occupied Voxel
NIA	Neighborhood Information Aggregation
KNN	K Nearest Neighbors
RAHT	Region-Adaptive Hierarchical Transform
PSNR	Peak-to-signal-noise ratio
BD-BR	Bjøntegaard Delta Rate
BD-PSNR	Bjøntegaard Delta PSNR

Table I gives the notations and abbreviations frequently used in this work.

II. Related Work

This section briefly reviews related publications that focus on the use of learning tools to improve the efficiency of PCC. In the meantime, a recent survey paper [12] on learning-based algorithms proposed in the past years serves as a good reference. Additionally, an article by Cao et al. in [2] can help audiences understand classical rules-based algorithms in PCC.

A. Point Cloud Geometry Compression (PCGC)

Many works were devoted to studying point cloud geometry compression (PCGC). To exploit the correlation across points, an appropriate presentation model is first required to construct the local neighborhood for information characterization and embedding. There are three popular representation models: point-based model [16], [18]–[20], [28]–[30], uniform voxel representation [23], [24], [31], octree scheme [32]–[36], and multiscale sparse representation [22], [25].

These approaches utilized the deep neural networks (DNNs) involving various computational layers, such as 3D dense convolution, sparse convolution, self-attention, or MLP (MultiLayer Perceptron), to predict the geometry occupancy probability of each voxel or point through the aggregation of neighborhood priors. As seen, their effective spatial neighborhood is defined by the representation models aforementioned. Among them, the work proposed by Guarda *et al.* [24] was now part of the JPEG Pleno PCC software model (*i.e.*, JPEG VM) under development; the SparsePCGC developed by Wang *et al.* [22] is under investigation in MPEG 3D Graphics group as a potential solution for next-generation AI PCC. Both approaches have shown significant performance improvement over the G-PCC in terms of point cloud geometry compression.

Instead of developing end-to-end solutions, another avenue is to restore the reconstruction quality by compensating for G-PCC-induced compression impairments. For example, GRASP-Net [37], DGPP [38], and GRNet [39] were proposed to remove artifacts induced by the underlying G-PCC coder.

B. Point Cloud Attribute Compression (PCAC)

Learning-based point cloud attribute compression (PCAC) just attracted attention recently. Compared with the geometry occupancy that presents binary status (occupied or not), attribute components (*e.g.*, RGB colors) exhibit larger variations across neighboring points, which, to some extent, makes them more difficult to characterize and embed in the raw data format. To this end, existing works often applied transforms (*e.g.*, Graph Fourier Transform [40], [41] and RAHT [11]) upon attributes to better represent signals in the transform domain. Note that the RAHT which is basically an adaptive Haar wavelet transform is adopted in G-PCC.

Some methods directly use stacked DNNs as the transform in an end-to-end framework. Two typical examples are the Deep-PCAC [42] and SparsePCAC [43]. In Deep-PCAC, MLPs were used to directly process unorganized, unordered points, with each input point cloud divided into 2048-point patches for processing. As seen, Deep-PCAC is constrained to exploiting point correlations within local patches and is prone to falling into local optima. Consequently, their results suffered a significant loss in BD-PSNR (up to 5.67 dB) compared to an earlier G-PCC version (TMC13v10). SparsePCAC used a simple autoencoder structure with sparse convolutions, offering better performance and requiring much less complexity than Deep-PCAC. Although SparsePCAC still presented inferior performance to the most recent G-PCC version (TMC13v19), it shows a promising direction for attribute compression using learning-based methods.

Instead of solely relying on the DNNs to characterize neighborhood correlations, both Nguyen *et al.* [35] and Wang *et al.* [44] proposed to use local neighbors explicitly structured by underlying representation models, such as octree or multiscale sparse tensor, to predict attribute probability. Their methods demonstrated superior performance compared to G-PCC in lossless mode. However, the decoding runtime of [35] is unbearable due to the use of autoregressive contexts in the entropy coder, while the decoding time of [44] is comparable to that of G-PCC because of its multistage parallelism strategy.

Similar to explorations in PCGC, some works employed DNNs to enhance the G-PCC attribute compression. For example, 3DAC [45] used an MLP-based entropy model to better encode the RAHT coefficients of G-PCC, which resulted in some gains on samples from ScanNet and Semantic KITTI datasets. Another example is CARNet [46] that developed a neural filter on top of the G-PCC for artifact removal and reported more than 20% compression gains.

Remark. As seen, most works focused on compressing either geometry or attribute information individually. Although Nguyen *et al.* [35] provided a solution to encode geometry and attribute under the same framework, it was limited to the lossless mode and therefore unsuitable for applications that desire lossy compression. Moreover, its extremely-long decoding complexity makes it impractical for real use.

III. DeepPCC

The journey of DeepPCC started from PCGCv2 [25], PCGFormer [27], and recent SparsePCAC [43]. The former two only discussed the lossy geometry compression while the latter focused on the attribute coding. Although PCGCv2/PCGFormer showed impressive performance improvements to the octree-based geometry compression in G-PCC, SparsePCAC was still largely inferior to the RAHT-based attribute coding of G-PCC. This work proposes a unified framework for lossy geometry and attribute compression, providing a comprehensive and promising solution for next-generation AI-based PCC. Fig 2 sketches the general structure of DeepPCC. As DeepPCC separates the processing of geometry and attribute coding, we next present more details on each.

A. Lossy DeepPCGC

The lossy geometry compression of DeepPCC, *a.k.a.*, DeepPCGC, is illustrated using the blue dotted lines in Fig. 2. Network implementation with detailed layers of DeepPCGC is given in the supplementary material.

Given an input point cloud frame $\mathcal{P}$, we use $\{C_X,\ F_X\}$ to represent its geometry part PCG, where C_X collects the

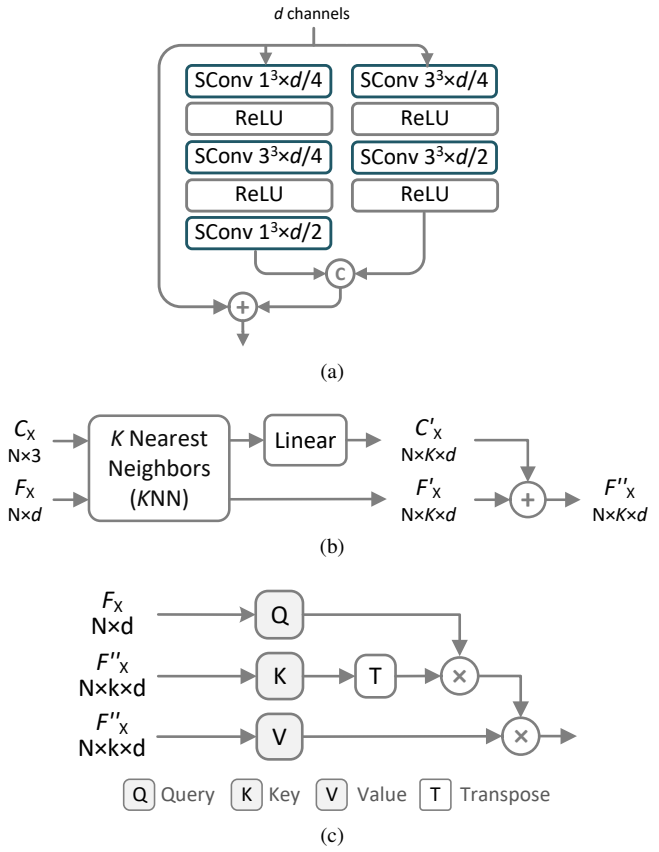


Fig. 3: **NIA Modules.** (a) **IRN** stacks sparse convolution (SConv) layers for information embedding. (b) **KNN** constructs adaptive local neighborhood by searching K nearest neighbors of each POV. (c) **KNN Self-Attention** used for local information aggregation where an attention map is generated for every group of K POVs in close proximity. Attentive features are then produced by multiplying the input value with the corresponding attention map.

coordinates of POVs and F_X is the feature vector associated with POVs. Specifically, for the input PCG, all elements in F_X are set to "1" (or TRUE) for the indication of positive occupation. F_X will be adapted by adjusting the channels from one scale to another to embed information aggregated from NIA blocks.

Encoder Transform. For the encoding, input geometry coordinates are dyadically downsampled to generate multi-resolution PCGs. At each scale, the NIA block $G_{\text{NIA}}(\cdot)$ is comprised by stacking an Inception-Residual Network (IRN) layer (see Fig. 3a), a KNN Self-Attention layer (see Fig. 3b and 3c), multiple sparse convolutional (SConv) layers, and ReLU layers for neighborhood correlation characterization and feature embedding. In this work, spatial resolution is downsampled three times to produce the thumbnail PCG C_Y, which is implemented using Sconv with a stride of 2. Correspondingly, three NIA blocks are devised to finally produce the latent feature F_Y for entropy coding, *i.e.*,

$$\{C_Y, F_Y\} = G_{\text{NIA}}\left(G_{\text{NIA}}\left(G_{\text{NIA}}\left(\{C_X, F_X\}\right)\right)\right). \quad (1)$$

Note that when processing features in NIA blocks, corresponding geometry coordinates are implicitly inferred.

Entropy Coding. At the bottleneck, the thumbnail PCG C_Y is losslessly encoded using the G-PCC (octree) codec; while the corresponding F_Y are quantized and entropy-coded using adaptive contexts learned from the data. Currently, we use joint hyperpriors and autoregressive neighbors in context modeling for optimal performance at the cost of the increased complexity [47]. We can simplify the context model by just using the hyperpriors or even fully factorized density model [48] for much faster processing speed. Following the Laplacian distribution used in [47], [49], entropy probability approximation can be translated as the derivation of entropy parameters (μ, σ).

Decoder Transform. For the decoding, it basically mirrors the encoder processing to progressively upscale, classify, and reconstruct PCGs. Specifically, at each scale, the NIA is reused to generate the occupancy probability for each voxel at the current scale. A classifying and pruning operation follows to sort the top M points as restored coordinates. Note that M is a preset point number counted for each scale in the encoder as ground truth and encapsulated in the bitstream for decoding. As seen, lower-scale geometry reconstruction serves as the conditional prior for such scale-wise refinement, which not only improves the reconstruction quality with less distortion but also significantly reduces the space and time complexity [25].

Loss Function. Recalling that geometry occupancy is a binary state, a hierarchical binary cross-entropy (BCE) loss is then applied to train the DeepPCGC model, *i.e.*,

$$\mathcal{L} = R + \lambda_g \cdot D = R_{\hat{y}} + R_{\hat{z}} + \lambda_g \cdot \mathcal{L}_{\text{BCE}}, \quad (2)$$

where λ_g controls the R-D trade-off. $R_{\hat{y}}$ and $R_{\hat{z}}$ are the bitrate of latent features $\hat{y}$ and hyperprior encoder $\hat{z}$ in DeepPCGC, respectively. The loss $\mathcal{L}_{\text{BCE}}$ is defined as

$$\mathcal{L}_{\text{BCE}} = \sum_i -(x_i \log_2 p(x_i) + (1 - x_i) log_2(1 - p(x_i))), \quad (3)$$

where x_i denotes the true occupancy state of a voxel and $p(x_i)$ is the estimated occupancy probability of this voxel. On the other hand, $1 - x_i$ represents the non-occupied voxel with the corresponding probability $1 - p(x_i)$. Such a BCE loss is also widely used in other learning-based point cloud geometry compression works [12].

B. Lossy DeepPCAC

The lossy attribute compression of DeepPCC, *a.k.a.*, DeepPCAC, is illustrated using red lines in Fig. 2, which shares a similar encoder-decoder structure (*e.g.*, transform and entropy coder) with DeepPCGC. Its Detailed network structure is also provided in the supplementary material. As aforementioned, geometry is losslessly compressed in this mode.

Encoder Transform. At the encoder, **NIA** blocks are also applied scale-wisely to effectively characterize and embed attribute information (*e.g.*, RGB colors in this study) of neighboring spatial regions for the derivation of attribute-related latent features. The same equation in Eq. (1) can be used to formulate the attribute information aggregation, where F_X

Fig. 4: **Attribute Filter** that uses filtering coefficients to guide the neural network for adaptive filtering. As such, the appearance of the reconstructed point cloud will be adjusted according to the input, uncompressed point cloud. The filtering coefficients will be coded and transmitted in the bitstream. 16 coefficients are employed in this work.

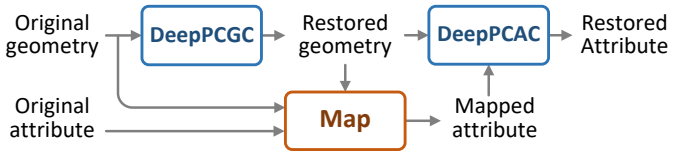


Fig. 5: **DeepPCC for Unified Lossy Coding** where DeepPCGC and DeepPCAC are connected consecutively. we first use DeepPCGC to encode and restore geometry coordinates and then map the original attributes to these restored coordinates for attribute coding using DeepPCAC.

and F_Y represent the original attribute tensor and its relevant latent features at the bottleneck, respectively. The geometry coordinates are implicitly inferred during the processing steps.

Different from deepPCGC which sequentially stacks of IRN and KNN Self-Attention layers for geometry compression, DeepPCAC allocates them in parallel branches to separately characterize attribute correlations in the local spatial neighborhood. The latent features obtained from each processing branch are then concatenated with the thumbnail point cloud downscaled three times at the bottleneck for entropy coding. Note that the attribute downscaling is achieved through average pooling, *i.e.*, eight voxels are squeezed to one voxel, with its intensity assigned using the average attribute value from the corresponding eight voxels.

Entropy Coding The same entropy engine is shared with DeepPCGC and DeepPCAC models. However, their context probabilities generated are different and substantially rely on their respective data.

Decoder Transform. At the decoder, symmetric NIA blocks are applied to process attribute-related features scale-wisely for the restoration of attribute intensities. Similarly, two parallel branches are devised to respectively apply the IRN and KNN Self-Attention layers. These two branches remain separate until they reach the last scale, where they are merged to produce the attribute reconstruction. Compression artifacts are inevitably introduced, and thus an additional Attribute Filtering is augmented next to further improve the reconstruction quality

Attribute Filtering. We devise a network using stacked sparse convolutions (Fig. 4) to generate 16 reconstruction candidates $\mathbf{R} = \{\mathbf{r}_0, \mathbf{r}_1, \cdots, \mathbf{r}_{15}\}$, which are then linearly combined by a set of filtering coefficients $A = \{a_0, a_1, \cdots, a_{15}\}$ for the final reconstruction $\hat{P} = \mathbf{R}A^T = a_0\mathbf{r}_0 + a_1\mathbf{r}_1 + \cdots + a_{15}\mathbf{r}_{15}$. These filtering coefficients are scaled to integers, compressed using the fixed-length codes, and explicitly signaled in the bitstream. The derivation of A can refer to our previous work [50].

Loss Functions. DeepPCAC is trained using a two-stage training approach. In the first stage, we use the Lagrangian loss for R-D optimization to train the multiscale network (without Attribute Filter) only:

$$\mathcal{L}_{s_1} = R + \lambda_a \cdot D = R_{\hat{y}} + R_{\hat{z}} + \lambda_a \cdot \left\| P - \widetilde{P} \right\|_2^2, \quad (4)$$

where λ_a is the parameter that controls the R-D trade-off. $R_{\hat{y}}$ and $R_{\hat{z}}$ are the bitrate of latent features $\hat{y}$ and hyperprior encoder $\hat{z}$ in DeepPCAC, respectively. The MSE distortion is derived between ground truth $P \in \mathcal{P}$ and reconstructed output $\widetilde{P} \in \widetilde{\mathcal{P}}$ in the YUV color space (without filtering).

Secondly, we fix the weighting parameters of the multiscale model and train the Attribute Filter only:

$$\mathcal{L}_{s_2} = \left\| P - \hat{P} \right\|_2^2 = \| P - (a_0\mathbf{r}_0 + \cdots + a_{15}\mathbf{r}_{15}) \|_2^2, \quad (5)$$

with $A = \{a_0, \cdots, a_{15}\}$ obtained by the least square optimization and $\{\mathbf{r}_0, \cdots, \mathbf{r}_{15}\}$ generated by the network of Attribute Filter (see Fig. 4).

C. Unified Lossy DeepPCC

As for the lossy compression of both geometry and attribute components of a given point cloud, DeepPCC connects pretrained DeepPCGC and DeepPCAC models, as illustrated in Fig. 5. More specifically, restored geometry coordinates from DeepPCGC are passed to DeepPCAC as implicit geometry prior, with which original attributes are mapped onto these lossy-compressed coordinates for subsequent processing.

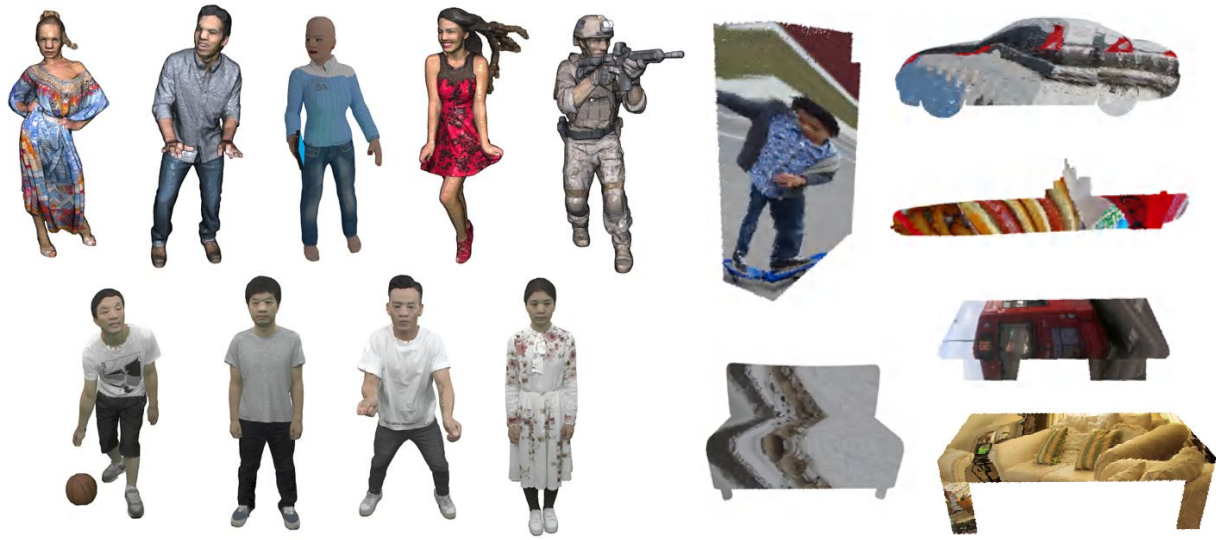
To perform the mapping, DeepPCC utilizes a KNN-based operation. For each reconstructed coordinate, DeepPCC identifies its K_M nearest points in the original, uncompressed point cloud and computes the average attribute value of these K_M points as its attribute. In our implementation, we set $K_M = 3$. Using a larger K_M may lead to oversmoothed attribute intensity, which is not recommended.

IV. Experiments

A. Testing and Training Conditions

Datasets. We randomly sampled 10,000 models from ShapeNet dataset [51] to first build our training dataset for point cloud geometry compression. This dataset is referred to as the ShapeNet PCG training dataset, in which we uniformly set 8-bit geometry precision. Such a collection of ShapeNet PCG samples is used to train the geometry compression model of DeepPCC, *i.e.*, DeepPCGC.

To establish the training dataset for the attribute compression of DeepPCC (*i.e.*, DeepPCAC), we followed the methodology discussed in SparsePCAC [43], in which color images randomly selected from COCO dataset [52] were projected onto the ShapeNet PCG samples to synthesize the color attributes.



(a) (b)

Fig. 6: **Testing Point Clouds.** (a) MPEG test sequences. (b) ShapeNet sequences with synthesized color.

TABLE II: BD-BR of DeepPCGC Compared with State-of-the-art Methods

Class	Point Cloud	GPCC (octree)		GPCC (trisoup)		ADL-PCC [24]		PCGCv2 [25]		PCGFormer [27]	
		D1 (%)	D2 (%)	D1 (%)	D2 (%)	D1 (%)	D2 (%)	D1 (%)	D2 (%)	D1 (%)	D2 (%)
MPEG 8iVFB	longdress	-90.10	-84.34	-51.61	-56.97	-65.98	-58.54	-19.48	-13.12	-8.14	-8.93
	loot	-91.60	-90.90	-64.15	-70.67	-70.30	-69.55	-41.67	-48.14	-6.88	-5.84
	queen	-92.21	-87.84	-77.22	-65.36	-69.00	-61.58	-24.71	-21.51	-14.90	-14.86
	redandblack	-87.47	-83.10	-41.17	-53.54	-57.46	-49.75	-25.26	-11.87	-7.05	-7.02
	solider	-88.63	-83.28	-44.11	-46.94	-64.54	-57.81	-19.47	-14.55	-5.05	-4.86
MPEG Owlii	basketball	-94.32	-93.00	-81.24	-75.37	-73.74	-70.14	-33.23	-32.53	-12.31	-11.78
	dancer	-92.96	-89.97	-76.46	-68.66	-72.53	-68.25	-27.85	-30.57	-11.80	-11.30
	exercise	-93.08	-89.99	-75.17	-67.88	-71.62	-68.54	-31.03	-34.10	-12.78	-12.58
	model	-91.44	-87.24	-65.56	-61.50	-68.30	-65.89	-19.45	-26.64	-13.66	-13.42
	Average	**-91.31**	**-87.74**	**-64.08**	**-62.99**	**-68.16**	**-63.34**	**-26.91**	**-25.89**	**-10.29**	**-10.07**

As for the datasets used for testing (see Fig 6), we used the testing samples selected by the PCC standardization in MPEG and JPEG committees [53], [54]. These samples are very different from the training data generated by the ShapeNet, thereby showcasing the robustness and generalization of DeepPCC. Figure 6a visualizes the testing samples, including:

- Five 8i Voxelized Full Bodies (8iVFB) [55]: *longdress*, *loot*, *queen*, *redandblack*, and *soldier*.
- Four sequences in the Owlii dataset [56][4]: *basketball_player*, *dancer*, *exercise*, and *model*.

For comparative studies, we also randomly selected 1000 new models covering 55 classes from the synthesized ShapeNet (visualized in Fig. 6b) for testing. Notice that these test samples are excluded from the training dataset.

Settings. DeepPCC is implemented using PyTorch and MinkowskiEninge [57], and runs on a computer with an Intel® i9-12900K CPU, 32 GB memory, and Nvidia 3090Ti RTX GPU. The network model is optimized by Adam with parameters β_1 and β_2 set to 0.9 and 0.999, respectively. The learning rate decays from 1e-4 to 1e-5 every two epochs. We randomly initialize the model, and each model is trained on the dataset for up to 50 epochs. It takes around 50 to 60 hours to converge on our platform.

To train DeepPCGC for geometry compression only, the λ_g in the loss function (Eq. (2)) is set as 0.1, 1, 5, and 10 to obtain four different lossy coding models; while to train DeepPCAC for lossy attribute coding, we adjust the R-D weight λ_a in the loss function (Eq. (4)) as 100, 600, 1000, and 2000 to obtain four different models.

The test settings closely follow the common test condition (CTC) suggested by the standardization community [53], [54]. We use bpp (*i.e.*, bits per point) to measure the compressed bitrate. For geometry distortion, we use point-to-point distance (D1) and point-to-plane distance (D2) to derive corresponding D1 PSNR and D2 PSNR; while for attribute distortion, we use Y PSNR and YUV PSNR to respectively evaluate luma component individually and superimposed luma and chroma components jointly. Moreover, a full-reference quality metric for colored 3D point clouds (PCQM) [1] which is reportedly more consistent with the human vision system is used to quantify the overall distortion of superimposed geometry and attribute information.

Benchmark. Next, we conduct experiments to demonstrate the performance from the following three aspects:

- **DeepPCGC** for lossy geometry compression only;

[4]We quantized sequences in Owlii to 10 bits for PCAC compression.

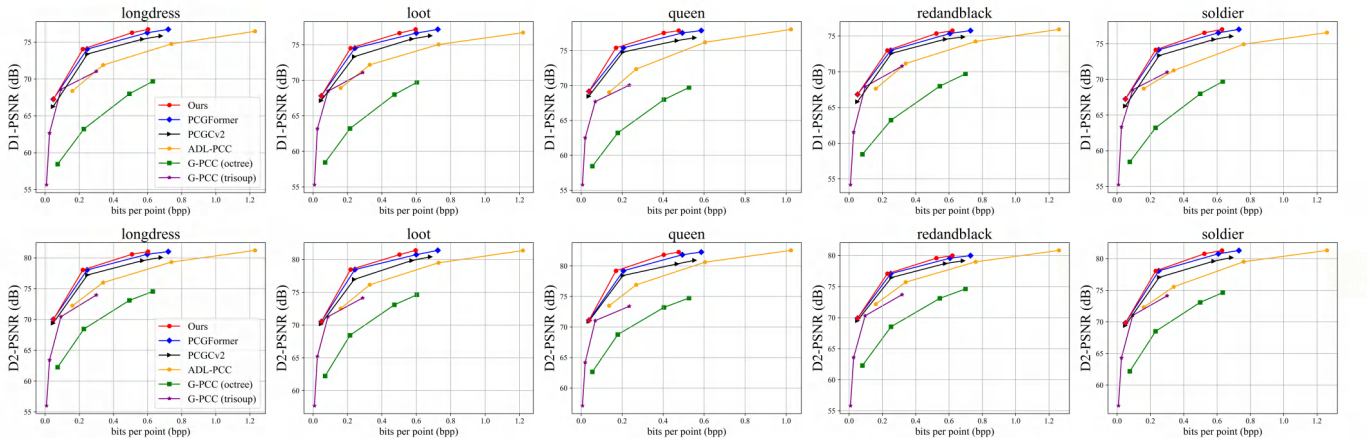


Fig. 7: **R-D Comparison across Different Geometry Compression Methods.** All neural models are trained under the same conditions. R-D curves measured in D1 and D2 PSNR are both provided.

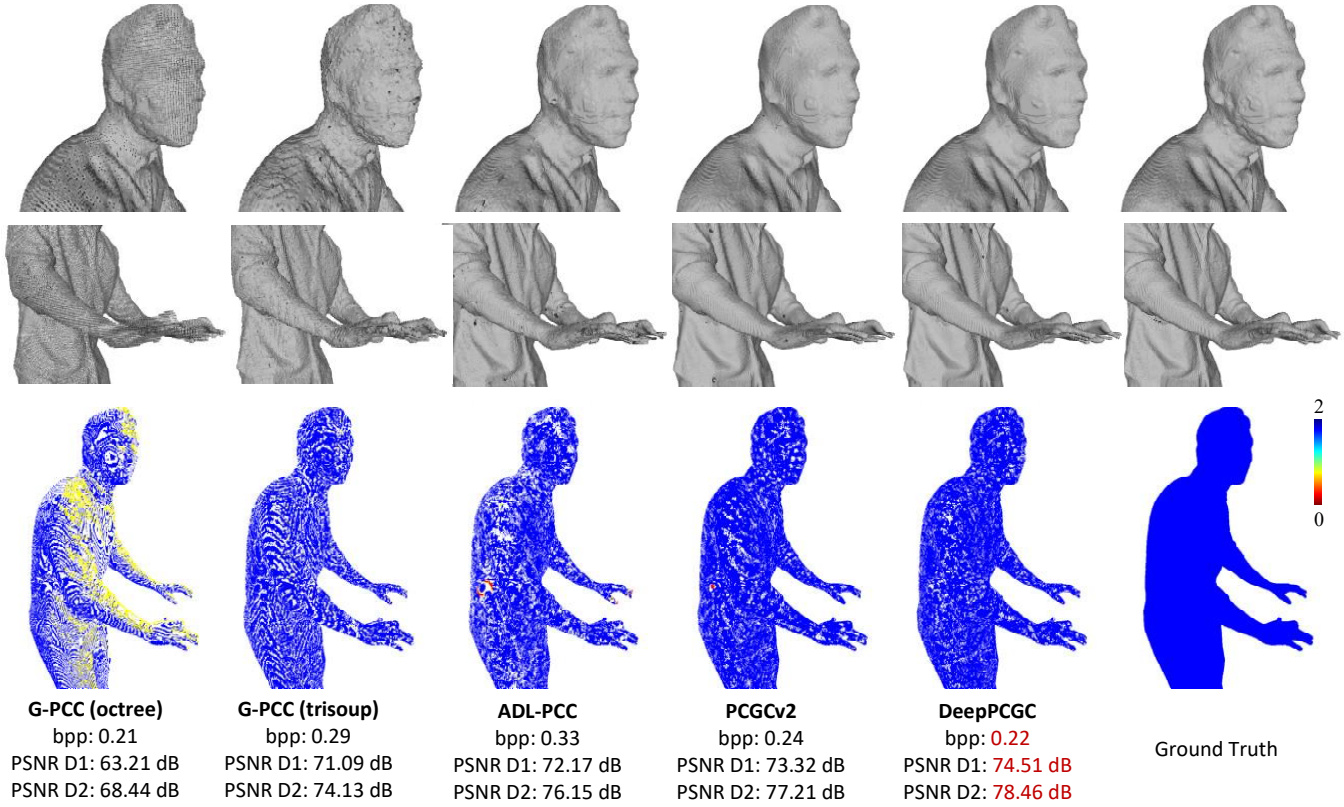


Fig. 8: **Visualized Reconstructions** of samples compressed using DeepPCGC, G-PCC (octree), G-PCC (trisoup), ADL-PCC [24], and PCGCv2 [25]. *Please zoom in for details.*

- **DeepPCAC** for lossy attribute coding but assuming the lossless compression of the corresponding geometry part;
- **DeepPCC** for lossy compression of both geometry and attribute components.

B. DeepPCGC Performance

BD-BR Comparison. Table II presents the BD-BR gains of DeepPCGC against state-of-the-art methods, including G-PCC (octree), G-PCC (trisoup), ADL-PCC [24], PCGCv2 [25], and PCGFormer [27]. The latest G-PCC, *i.e.*, TMC13v19, is employed. As seen, compared with the anchor G-PCC (octree), DeepPCC achieves 91.31% (87.74%) BD-BR reduction in D1 (D2) measurement on average. Moreover, DeepPCC significantly outperforms other learning-based methods, *e.g.*, it surpasses the ADL-PCC by 68.16% (63.34%) BD-BR gains. Note that ADL-PCC was adopted in JPEG VM for geometry coding. R-D curves plotted in Fig. 7 consistently show evidence the leading performance of DeepPCGC across a variety of bitrates.

The DeepPCGC achieves a BD-BR reduction of 26.91%

TABLE III: BD-BR and BD-PSNR of DeepPCAC to State-of-the-art Methods

Class	Point Cloud	TMC13v6				SparsePCAC [43]				TMC13v19			
		BD-BR (%)		BD-PSNR (dB)		BD-BR (%)		BD-PSNR (dB)		BD-BR (%)		BD-PSNR (dB)	
		Y	YUV	Y	YUV	Y	YUV	Y	YUV	Y	YUV	Y	YUV
MPEG 8iVFB	longdress	-54.91	-52.54	2.19	2.05	-32.07	-26.36	1.00	0.79	-13.48	-8.46	0.36	0.23
	loot	-35.61	-13.48	1.25	0.36	-35.02	-17.26	1.30	0.93	34.69	40.19	-0.79	-1.81
	queen	-45.76	-38.03	1.16	0.93	-46.17	-40.92	1.14	1.02	-1.72	9.11	-0.07	-0.34
	redandblack	-60.72	-51.46	2.55	1.89	-35.61	-29.73	1.11	0.87	-16.18	1.76	0.40	-0.19
	soldier	-58.76	-51.23	2.59	1.98	-41.65	-47.05	1.38	1.71	-13.66	-1.74	0.43	0.01
MPEG Owlii	basketball	-59.96	-55.64	3.08	2.15	-44.43	-48.86	1.80	1.81	-14.96	-1.69	0.48	-0.05
	dancer	-65.81	-61.88	3.53	2.68	-47.24	-49.63	1.88	1.91	-26.84	-21.74	0.98	0.48
	exercise	-54.43	-61.59	2.10	1.61	-47.93	-63.68	1.70	1.92	-1.93	9.25	0.05	-0.27
	model	-60.17	-60.64	3.03	2.68	-38.47	-42.32	1.43	1.57	-16.64	-20.23	0.55	0.61
	Average	**-55.13**	**-49.61**	**2.39**	**1.81**	**-40.95**	**-40.64**	**1.42**	**1.39**	**-7.86**	**0.72**	**0.27**	**-0.15**
ShapeNet 1000 sequences	**Average**	**-54.98**	**-57.62**	**2.46**	**2.35**	**-19.05**	**-21.52**	**0.58**	**0.61**	**-32.14**	**-34.99**	**1.23**	**1.14**

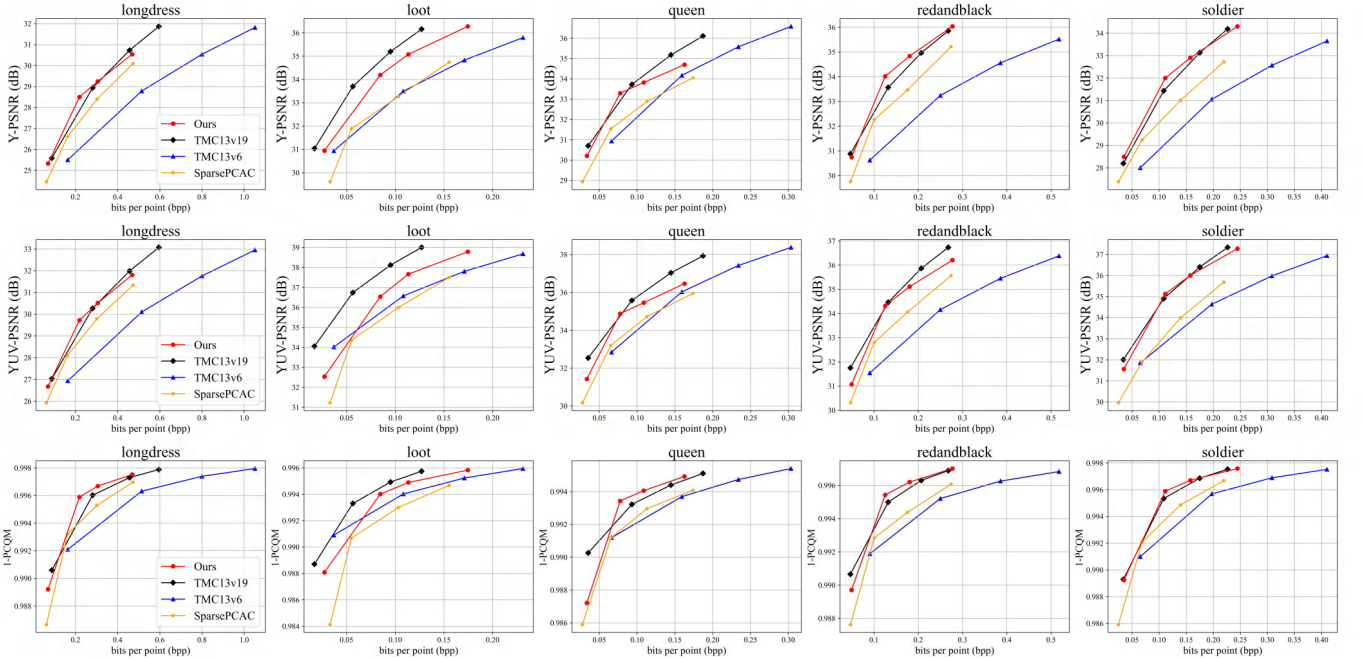


Fig. 9: **R-D Comparison across Different Attribute Compression Methods.** The colorized ShapeNet samples are used to train the SparsePCAC and DeepPCAC models. R-D curves measured using Y PSNR, YUV PSNR, and PCQM metrics are provided.

(25.89%) over the PCGCv2 through the introduction of self-attention and context-adaptive entropy coding using joint autoregressive neighbors and hyperpriors. Moreover, it outperforms the PCGFormer by 10.29% (10.07%) BD-BR by using context-adaptive entropy coding.

Qualitative Comparison. We visualize the reconstructed point cloud geometry of various methods in Fig. 8. Clearly, the reconstructed geometry of G-PCC (octree) suffers from severe compression noise and exhibits noticeable errors compared to the ground truth. This occurs because G-PCC (octree) significantly reduces the number of points after compression, leading to insufficient points for accurate surface reconstruction. Although G-PCC (trisoup) has less compression noise, it still presents holes in their reconstructions, such as the face of *loot*. ADL-PCC and PCGCv2 yield smoother surfaces; however, certain regions still present large errors, as indicated by the red color in the error maps. In contrast, the proposed DeepPCGC approach produces visually pleasing reconstructions that are very close to the ground truth.

C. DeepPCAC Performance

BD-BR Comparison. In this section, assuming that the geometry is losslessly compressed, we compare the DeepPCAC with the G-PCC TMC13v6, the latest G-PCC TMC13v19, and SparsePCAC [43]. The SparsePCAC is trained and tested using the same conditions as ours.

In Table III, for MPEG test samples, DeepPCAC presents noticeable gains over existing methods, *e.g.*, it exceeds the SparsePCAC by 40.95% and 40.64% BD-BR gains. When compared with the TMC13v19 anchor, DeepPCAC achieves an average 7.86% BD-BR reduction on the Y component. R-

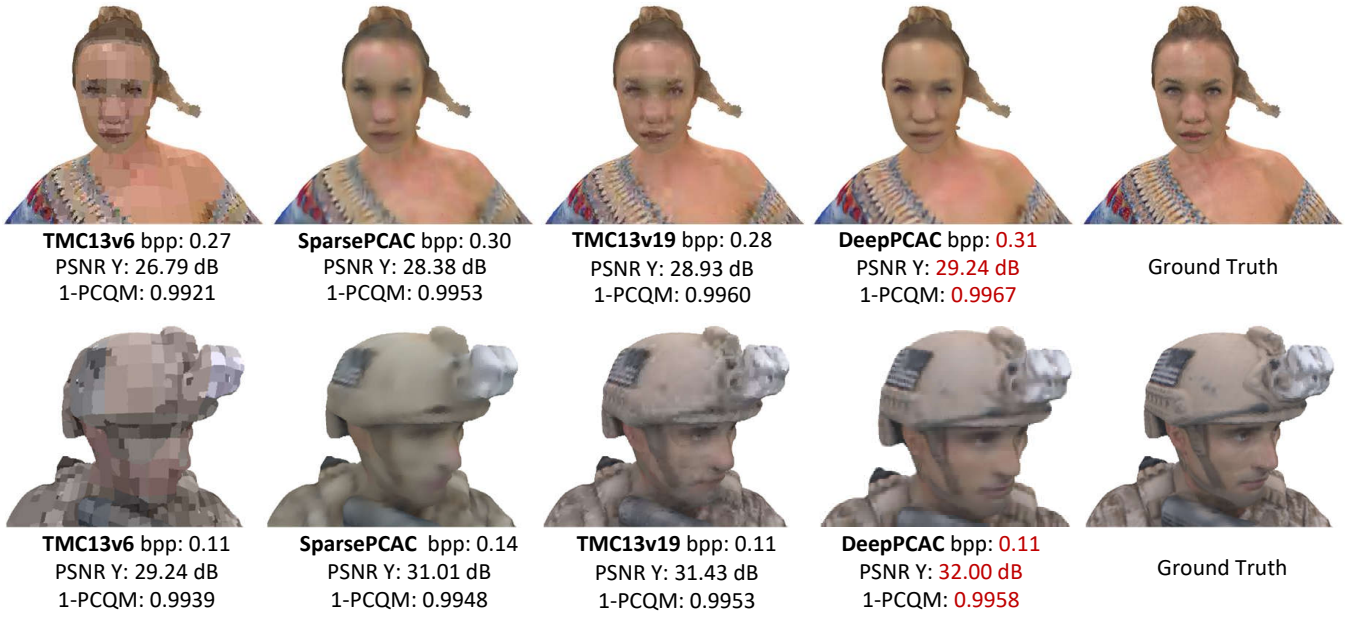


Fig. 10: **Visualized Reconstructions** of samples compressed using DeepPCAC, TMC13v6, SparsePCAC, and the latest TMC13v19. *Please zoom in for details.*

TABLE IV: PCQM Performance of DeepPCAC against State-of-the-art Methods

Class	Point Cloud	TMC13v6	SparsePCAC	TMC13v19
		BD-BR (%)	BD-BR (%)	BD-BR (%)
MPEG 8iVFB	longdress	-49.68	-27.27	-19.20
	loot	-10.95	-23.30	34.15
	queen	-51.03	-55.29	-34.22
	redandblack	-54.56	-38.01	-22.38
	soldier	-47.86	-35.69	-15.12
MPEG Owlii	basketball	-48.33	-36.41	-9.87
	dancer	-56.26	-45.50	-24.55
	exercise	-33.23	-27.51	-24.53
	model	-50.37	-32.48	-14.42
	Average	**-44.70**	**-35.72**	**-9.01**

D curves are exemplified in Fig. 9 to supplement the average BD-BR/BD-PSNR measures.

For the YUV space, 0.72% BD-BR loss is observed due to the substantial loss in *loot* sequence. This is potentially caused by the irregular texture of the shirt, which can not be well characterized by the current DeepPCAC model trained from a vastly different synthesized ShapeNet dataset. An interesting problem is to generate training samples that are more realistic and closer to the point clouds used in practice. To validate our hypothesis, Table III also reports the performance using the synthesized ShapeNet test dataset. As seen, the DeepPCC uniformly outperforms state-of-the-art methods by a large margin, *e.g.*, it gains as large as 32.14% and 34.99% BD-BR over the TMC13v19 and 19.05% and 21.52% over the SparsePCAC in Y and YUV spaces, respectively.

Furthermore, DeepPCAC also exhibits superiority when evaluated by the PCQM metric [1]. The PCQM is a linear combination of geometry and color features, which are computed between a distorted point cloud and its corresponding reference. As presented in Fig. 9 and Table IV, on average, DeepPCAC provides significant performance gains.

TABLE V: Experimental Settings in G-PCC and DeepPCC

GPCC	Octree (PQS*)	0.25	0.5	0.75	0.875
	RAHT (QP**)	35	35	36	36
DeepPCC	DeepPCGC (λ_g)	0.1	1	5	10
	DeepPCAC (λ_a)	100	600	1000	2000

* PQS-Position Quantization Scale; **QP-Quantization Parameter.

In both Table III and Table IV, almost 40% BD-BR gains are reported when comparing DeepPCAC with SparsePCAC, which is mainly due to the use of proposed NIA module and Attribute Filter.

Qualitative Comparison. Furthermore, we illustrate the qualitative results of different methods in Fig. 10. Clearly, the visualization of SparsePCAC looks very blurry and exhibits some color casting (*e.g.*, on the lady's face and feet). The reconstruction of G-PCC TMC13v19 contains severe blocking artifacts, whereas DeepPCAC provides a very smooth and clear reconstruction, which is visually more appealing than the other methods.

D. DeepPCC Performance

Overall Performance. We form a unified DeepPCC solution for the lossy compression of both geometry and attribute by connecting the DeepPCAC and DeepPCGC models. We further conduct a comprehensive study on the compression performance of DeepPCC, compared with the latest G-PCC (TMC13v19) which is also comprised of two modules, namely G-PCC (octree) and G-PCC (RAHT). To ensure a fair comparison, we try our best to set comparable geometry bit rates in DeepPCGC and G-PCC (octree), and also similar attribute bit rates in DeepPCAC and G-PCC (RAHT), which eventually leads to comparable total bit rates. Table V details the encoder settings in DeepPCC and G-PCC. However, it is important to note that such settings may not be optimal for these two

TABLE VI: Compression Gains of DeepPCC to G-PCC (TMC13v19)

Point Cloud	DeepPCGC vs. TMC13v19				DeepPCAC vs. TMC13v19				DeepPCC
	D1		D2		Y		YUV		PCQM
	BD-BR (%)	BD-PSNR (dB)	BD-BR (%)	BD-PSNR (dB)	BD-BR (%)	BD-PSNR (dB)	BD-BR (%)	BD-PSNR (dB)	BD-BR (%)
longdress	-79.65	10.50	-62.23	9.41	-31.12	0.60	-21.50	0.41	-92.37
loot	-83.68	10.11	-73.83	8.89	-28.36	0.78	7.66	-0.07	-43.36
queen	-87.64	11.64	-78.17	9.99	-17.12	0.49	-5.38	0.26	-72.59
redandblack	-79.35	9.35	-54.33	8.28	-32.16	0.87	-21.56	0.21	-50.58
soldier	-80.12	10.09	-68.73	8.86	-41.38	1.05	-29.99	0.63	-56.24
Average	**-82.09**	**10.34**	**-67.46**	**9.08**	**-30.03**	**0.76**	**-14.15**	**0.29**	**-63.03**

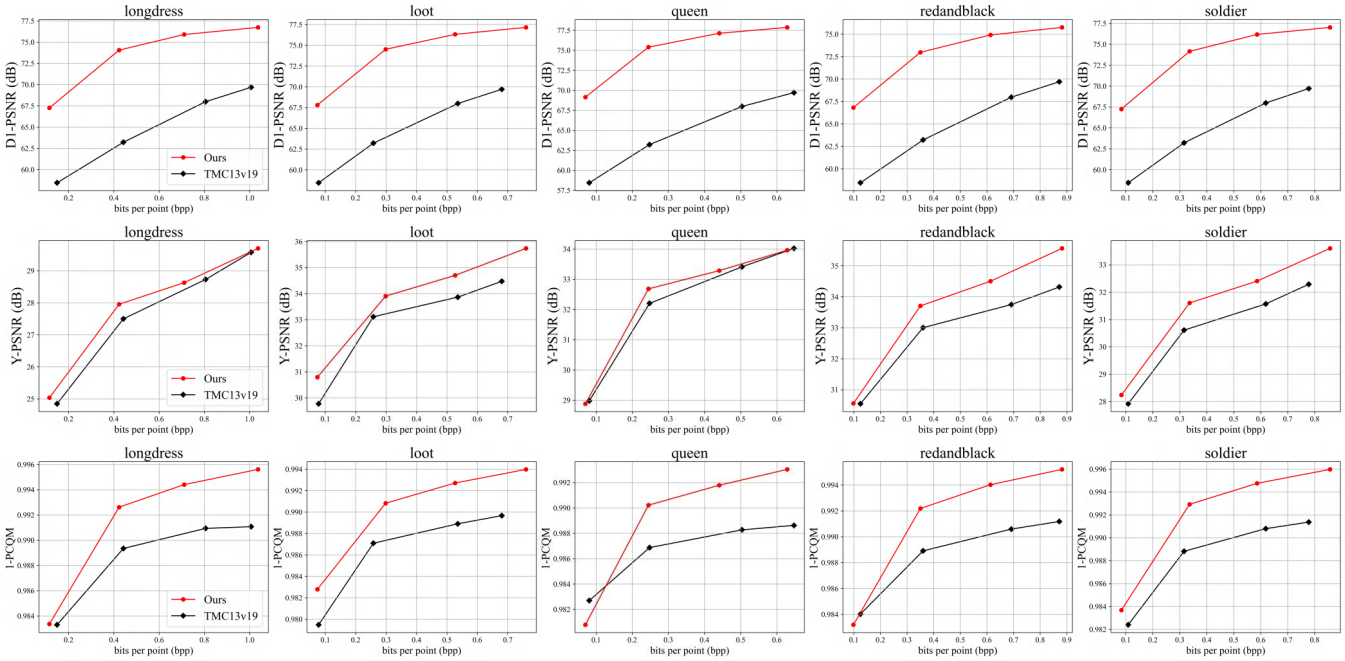


Fig. 11: **R-D Curves of DeepPCC and G-PCC (TMC13v19) for Unified Lossy Solution.** Distortion measures using D1 PSNR, Y PSNR, and PCQM are provided for comparison. The horizontal axis represents the total bit rates. It is worth noting that we ensure DeepPCC and G-PCC have comparable geometric bit rates and comparable attribute bit rates for a fair comparison.

methods but just serve as a reference to provide an intuitive understanding of their performance. On the other hand, since both G-PCC and DeepPCC currently have no accurate rate control technique, bitrate mismatch is inevitable.

We evaluate the restored geometry quality using D1 and D2 metrics and the restored attribute quality using Y and YUV PSNR. Furthermore, we employ the PCQM metric to measure the reconstructed quality of both geometry and attribute. As shown in Table VI, DeepPCC outperforms the G-PCC TMC13v19 by 82.09% (67.46%) BD-BR in D1 (D2) measurement, 30.03% (14.15%) BD-BR in Y (YUV) space, respectively. Regarding the overall PCQM that measures the superimposed distortion from both geometry and attributes, DeepPCC attains 63.03% BD-BR gains over the TMC13v19. Corresponding R-D curves are provided in Fig. 11.

The reconstructed point clouds from different methods are visualized in Fig. 1 for comparison. We observe that the point cloud reconstructed by G-PCC contains abundant compression noise, blocking artifacts, and black holes, which severely deteriorate the visual quality. The blocking artifacts are mainly triggered by the RAHT and quantization algorithms utilized in the G-PCC attribute compression process, while the black holes come from the absence of geometry coordinates after G-PCC geometry compression. In contrast, the proposed DeepPCC provides much better visual representation with smooth surfaces and clear edges, yielding more visually pleasing reconstructions. We observe that some regions in our reconstruction look smooth. This is because we employ the MSE loss to train DeepPCAC. This problem can be largely resolved through the use of a different loss such as MPED [58], which will be explored in our future work.

We further conduct experiments to evaluate the performance of DeepPCC on the Microsoft Voxelized Upper Bodies (MVUB) [59] dataset, which consists of five point clouds: *Andrew, David, Phil, Ricardo*, and *Sarah*. Corresponding R-D curves are presented in Fig. 12. It is shown that our DeepPCGC outperforms G-PCC by a large margin, providing a superior foundation for the subsequent PCAC processing. Benefiting from the well-reconstructed geometry, our DeepPCAC significantly outperforms G-PCC. As a result, the overall PCQM of DeepPCC, which jointly assesses the quality of geometry and attribute, surpasses that of the G-PCC remarkably.

The comparative study to the JPEG VM is supplemented in a companion material due to space limitation.

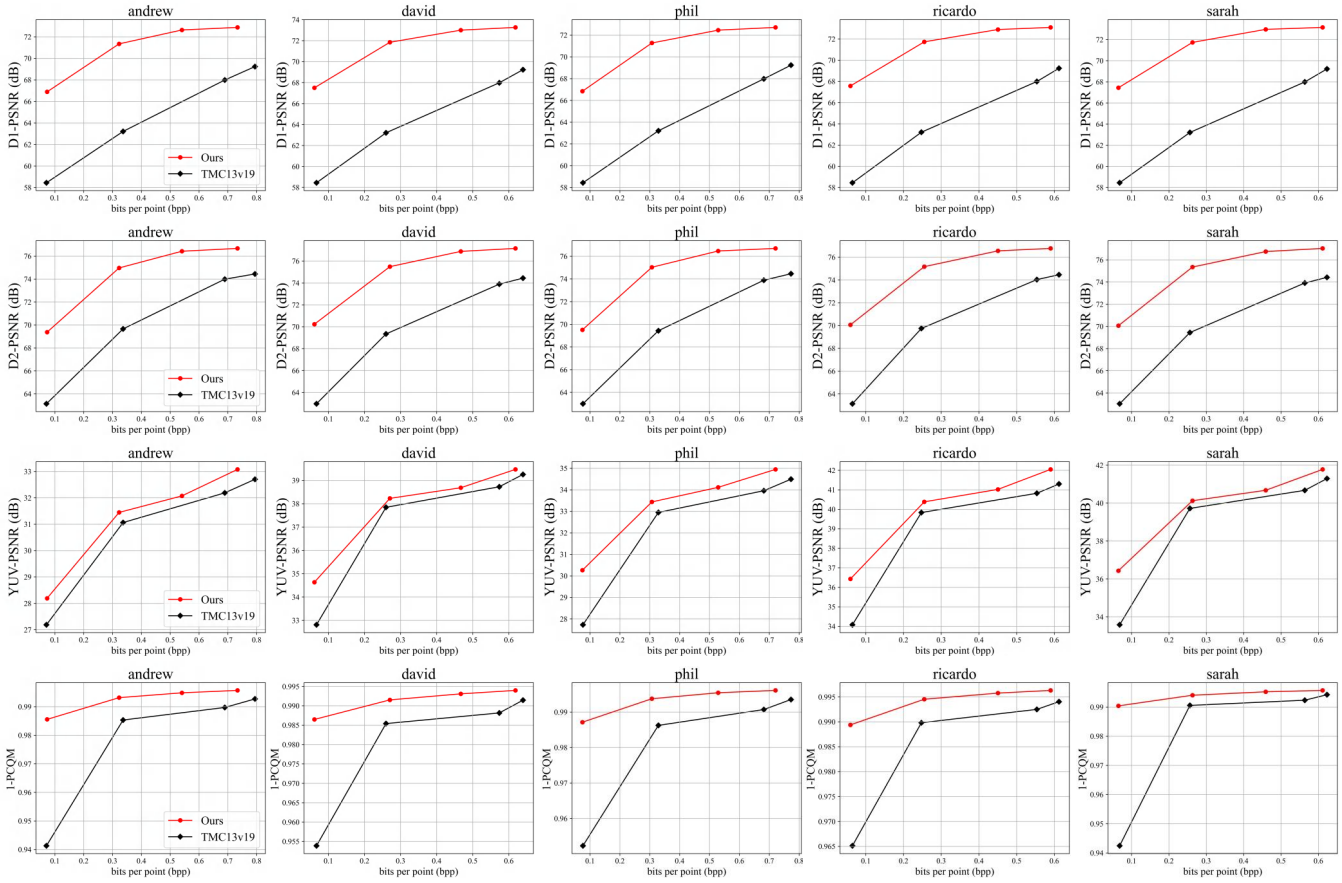


Fig. 12: **R-D Comparison using MVUB Dataset.** Our DeepPCC models are trained using ShapeNet. We ensure DeepPCC and G-PCC have comparable geometric bit rates and comparable attribute bit rates for a fair comparison.

E. Space and Time Complexity

Our DeepPCGC model uses 1.26 M parameters and its model size is 5.30 MB. Our DeepPCAC model has 25.63 M parameters, of which the Attribute Filter uses 1.51 M parameters. Its model size is 119.0 MB, of which the Attribute Filter costs 6 MB.

The runtime of DeepPCC is presented in Table VII. It is observed that its decoding time significantly exceeds its encoding time. This is attributed to the sequential decoding process involved in our entropy coding using the autogressive model. It is worth noting that using the entropy model with only hyperprior can reduce the runtime by approximately 90% but at the cost of a significant decline in performance, as we have demonstrated in our previous work [25]. Recently, a multistage context modeling method developed in [60] not only reports competitive performance to the context model using autoregressive neighbors but also demonstrates 6× to 8× decoding speedup, which can be borrowed to accelerate the DeepPCC entropy decoding in the future.

TABLE VII: DeepPCC Runtime Summary

Point Cloud	DeepPCGC		DeepPCAC		DeepPCC	
	Enc (s)	Dec (s)	Enc (s)	Dec (s)	Enc (s)	Dec (s)
longdress	2.15	16.68	24	216	23.15	220.68
loot	2.01	15.26	20	182	21.01	188.26
queen	2.49	15.66	22	195	29.4	223.44
redandblack	2.01	14.97	19	175	24.6	192.68
soldier	2.85	21.37	30	306	37.22	333.04
Average	**2.30**	**16.79**	**23.00**	**214.80**	**27.08**	**231.63**

V. Concluding Remarks

In this work, a learning-based lossy point cloud compression approach – DeepPCC was proposed and evaluated. Leveraging the powerful representation capability of deep learning techniques, DeepPCC employs a simple autoencoder architecture consisting of stacked neural network layers to compactly represent the geometry and attribute information of large-scale point clouds. To best exploit inter-point correlations for efficient compression, we develop a multiscale neighborhood information aggregation method that progressively downscales the input point cloud to generate multi-resolution scales, and at each resolution scale, the combination of sparse convolution and local self-attention is applied to effectively characterize and embed correlations across spatial neighbors. Extensive simulations demonstrate the superior performance of DeepPCC over existing methods, such as MPEG G-PCC and JPEG VM, in compressing individual components (*e.g.*, geometry or attribute) or color point cloud with both geometry and attribute information.

For future work, balancing the bitrates of geometry and attribute components for global optimal is an interesting

problem, which is largely overlooked in existing methods. In addition, it is important to develop a more accurate metric to measure the superimposed distortion of geometry and attributive information in practice. Another promising research direction is to extend the DeepPCC approach to compress dynamic object point clouds and other types of content, such as sparse LiDAR data.

REFERENCES

[1] G. Meynet, Y. Nehmé, J. Digne, and G. Lavoué, "PCQM: A full-reference quality metric for colored 3D point clouds," in *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE, 2020, pp. 1–6.

[2] C. Cao, M. Preda, and T. Zaharia, "3D point cloud compression: A survey," in *The 24th International Conference on 3D Web Technology*, 2019, pp. 1–9.

[3] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li *et al.*, "Emerging MPEG standards for point cloud compression," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 133–148, 2018.

[4] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE signal processing magazine*, vol. 15, no. 6, pp. 74–90, 1998.

[5] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[6] C. Cao, M. Preda, V. Zakharchenko, E. S. Jang, and T. Zaharia, "Compression of sparse and dense dynamic point clouds—methods and standards," *Proceedings of the IEEE*, vol. 109, no. 9, pp. 1537–1558, 2021.

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

[8] "ISO/IEC 23090-5 - Visual Volumetric Video-based Coding (V3C) and Video-based Point Cloud Compression (V-PCC)."

[9] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[10] B. Bross, J. Chen, J.-R. Ohm, G. J. Sullivan, and Y.-K. Wang, "Developments in international video coding standardization after AVC, with an overview of versatile video coding (VVC)," *Proceedings of the IEEE*, vol. 109, no. 9, pp. 1463–1493, 2021.

[11] R. L. De Queiroz and P. A. Chou, "Compression of 3D point clouds using a region-adaptive hierarchical transform," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3947–3956, 2016.

[12] M. Quach, J. Pang, D. Tian, G. Valenzise, and F. Dufaux, "Survey on deep learning-based point cloud compression," *Frontiers in Signal Processing*, 2022.

[13] WG7, MPEG 3D Graphics Coding, "Performance analysis of currently ai-based available solutions for PCC," *ISO/IEC JTC 1/SC 29/WG 7 N0174*, July 2021.

[14] J. Pleno, "JPEG pleno point cloud coding common test conditions v3.5," *ISO/IEC JTC1/SC29/WG1 N90053*, 2020.

[15] WG7, "Common test conditions for G-PCC," *ISO/IEC JTC1/SC29/WG11 N00106*, 2021.

[16] T. Huang and Y. Liu, "3D point cloud geometry compression on deep learning," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 890–898.

[17] L. Gao, T. Fan, J. Wang, Y. Xu, and Z. Ma, "Point cloud geometry compression via neural graph sampling," *IEEE ICIP*, 2021.

[18] J. Zhang, G. Liu, D. Ding, and Z. Ma, "Transformer and upsampling-based point cloud compression," in *Proceedings of the 1st International Workshop on Advances in Point Cloud Compression, Processing and Analysis*, ser. APCCPA '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 33–39.

[19] K. You and P. Gao, "Patch-based deep autoencoder for point cloud geometry compression," in *ACM Multimedia Asia*, 2021, pp. 1–7.

[20] K. You, P. Gao, and Q. Li, "IPDAE: Improved patch-based deep autoencoder for lossy point cloud geometry compression," in *Proceedings of the 1st International Workshop on Advances in Point Cloud Compression, Processing and Analysis*, 2022, pp. 1–10.

[21] T. Huang, J. Zhang, J. Chen, Z. Ding, Y. Tai, Z. Zhang, C. Wang, and Y. Liu, "3QNet: 3D point cloud geometry quantization compression network," *ACM Trans. Graph.*, vol. 41, no. 6, nov 2022.

[22] J. Wang, D. Ding, Z. Li, X. Feng, C. Cao, and Z. Ma, "Sparse tensor-based multiscale representation for point cloud geometry compression," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[23] J. Wang, H. Zhu, H. Liu, and Z. Ma, "Lossy point cloud geometry compression via end-to-end learning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 12, pp. 4909–4923, 2021.

[24] A. F. R. Guarda, N. M. M. Rodrigues, and F. Pereira, "Adaptive deep learning-based point cloud geometry coding," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, pp. 415–430, 2021.

[25] J. Wang, D. Ding, Z. Li, and Z. Ma, "Multiscale point cloud geometry compression," in *2021 Data Compression Conference (DCC)*. IEEE, 2021, pp. 73–82.

[26] C. Choy, J. Gwak, and S. Savarese, "4D spatio-temporal convnets: Minkowski convolutional neural networks," in *IEEE CVPR*, 2019.

[27] G. Liu, J. Wang, D. Ding, and Z. Ma, "PCGFormer: Lossy point cloud geometry compression via local self-attention," in *IEEE VCIP*, 2022.

[28] W. Yan, S. Liu, T. H. Li, Z. Li, G. Li *et al.*, "Deep autoencoder-based lossy geometry compression for point clouds," *arXiv preprint arXiv:1905.03691*, 2019.

[29] X. Wen, X. Wang, J. Hou, L. Ma, Y. Zhou, and J. Jiang, "Lossy geometry compression of 3D point cloud data via an adaptive octree-guided network," in *2020 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2020, pp. 1–6.

[30] L. Gao, T. Fan, J. Wan, Y. Xu, J. Sun, and Z. Ma, "Point cloud geometry compression via neural graph sampling," in *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2021, pp. 3373–3377.

[31] M. Quach, G. Valenzise, and F. Dufaux, "Learning convolutional transforms for lossy point cloud geometry compression," in *IEEE ICIP*, 2019, pp. 4320–4324.

[32] Z. Que, G. Lu, and D. Xu, "VoxelContext-Net: An octree based framework for point cloud compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6042–6051.

[33] L. Huang, S. Wang, K. Wong, J. Liu, and R. Urtasun, "OctSqueeze: Octree-structured entropy model for LiDAR compression," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 1313–1323.

[34] D. T. Nguyen, M. Quach, G. Valenzise, and P. Duhamel, "Lossless coding of point cloud geometry using a deep generative model," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 12, pp. 4617–4629, 2021.

[35] D. T. Nguyen and A. Kaup, "Lossless point cloud geometry and attribute compression using a learned conditional probability model," *IEEE Transactions on Circuits and Systems for Video Technology*, 2023.

[36] C. Fu, G. Li, R. Song, W. Gao, and S. Liu, "OctAttention: Octree-based large-scale context model for point cloud compression," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2022.

[37] J. Pang, M. A. Lodhi, and D. Tian, "GRASP-Net: Geometric residual analysis and synthesis for point cloud compression," in *Proceedings of the 1st International Workshop on Advances in Point Cloud Compression, Processing and Analysis*, 2022, pp. 11–19.

[38] X. Fan, G. Li, D. Li, Y. Ren, W. Gao, and T. H. Li, "Deep geometry post-processing for decompressed point clouds," in *2022 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2022, pp. 1–6.

[39] D. Ding, G. Liu, R. Xue, J. Li, and Z. Ma, "GRNet: Geometry restoration for compressed point clouds using auxiliary density signaling," *submitted to IEEE TVCG*, 2022.

[40] C. Zhang, D. Florencio, and C. Loop, "Point cloud attribute compression with graph transform," in *2014 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2014, pp. 2066–2070.

[41] R. A. Cohen, D. Tian, and A. Vetro, "Point cloud attribute compression using 3-D intra prediction and shape-adaptive transforms," in *2016 Data Compression Conference (DCC)*. IEEE, 2016, pp. 141–150.

[42] X. Sheng, L. Li, D. Liu, Z. Xiong, Z. Li, and F. Wu, "Deep-PCAC: An end-to-end deep lossy compression framework for point cloud attributes," *IEEE Transactions on Multimedia*, vol. 24, pp. 2617–2632, 2021.

[43] J. Wang and Z. Ma, "Sparse tensor-based point cloud attribute compression," in *2022 IEEE 5th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, 2022, pp. 59–64.

[44] J. Wang, D. Ding, and Z. Ma, "Lossless point cloud attribute compression using cross-scale, cross-group, and cross-color prediction," in *Data Compression Conference (DCC)*. IEEE, 2023.

[45] G. Fang, Q. Hu, H. Wang, Y. Xu, and Y. Guo, "3DAC: Learning attribute compression for point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 14 819–14 828.

[46] D. Ding, J. Zhang, J. Wang, and Z. Ma, "CARNet: Compression artifact reduction for point cloud attribute," *arXiv preprint arXiv:2209.08276*, 2022.

[47] D. Minnen, J. Ballé, and G. D. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," *Advances in neural information processing systems*, vol. 31, 2018.

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

[49] T. Chen, H. Liu, Z. Ma, Q. Shen, X. Cao, and Y. Wang, "End-to-end learnt image compression via non-local attention optimization and improved context modeling," *IEEE Transactions on Image Processing*, vol. 30, pp. 3179–3191, 2021.

[50] L. Kong, D. Ding, F. Liu, D. Mukherjee, U. Joshi, and Y. Chen, "Guided CNN restoration with explicitly signaled linear combination," in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 3379–3383.

[51] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su *et al.*, "ShapeNet: An information-rich 3D model repository," *arXiv preprint arXiv:1512.03012*, 2015.

[52] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[53] Z. Alexandre, D. Graziosi, and T. Ali, "Dataset split for AI-PCC," *ISO/IEC JTC 1/SC 29/WG 7 m58754*, Jan. 2022.

[54] WG7, MPEG 3D Graphics Coding, "Description of exploration experiment 5.3 on AI-based dynamic pc coding," *ISO/IEC JTC 1/SC 29/WG 7 N00386*, July 2022.

[55] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, "8i voxelized full bodies - a voxelized point cloud dataset," *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) m38673/M72012*, May 2016.

[56] X. Yi, L. Yao, and W. Ziyu, "Owlii dynamic human mesh sequence dataset," *ISO/IEC JTC1/SC29/WG11 (MPEG/JPEG) m41658*, October, 2017.

[57] C. Choy, J. Gwak, and S. Savarese, "4D spatio-temporal ConvNets: Minkowski convolutional neural networks," *2019 IEEE/CVF CVPR*, pp. 3070–3079, 2019.

[58] Q. Yang, Y. Zhang, S. Chen, Y. Xu, J. Sun, and Z. Ma, "MPED: Quantifying point cloud distortion based on multiscale potential energy discrepancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[59] C. Loop, Q. Cai, S. O. Escolano, and P. A. Chou, "Microsoft voxelized upper bodies-a voxelized point cloud dataset," *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document m38673 M*, vol. 72012, p. 2016, 2016.

[60] M. Lu and Z. Ma, "High-efficiency lossy image coding through adaptive neighborhood information aggregation," *arXiv preprint arXiv:2204.11448*, 2022.

[61] W. Zhu, Y. Xu, D. Ding, Z. Ma, and M. Nilsson, "Lossy point cloud geometry compression via region-wise processing," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 12, pp. 4575–4589, 2021.

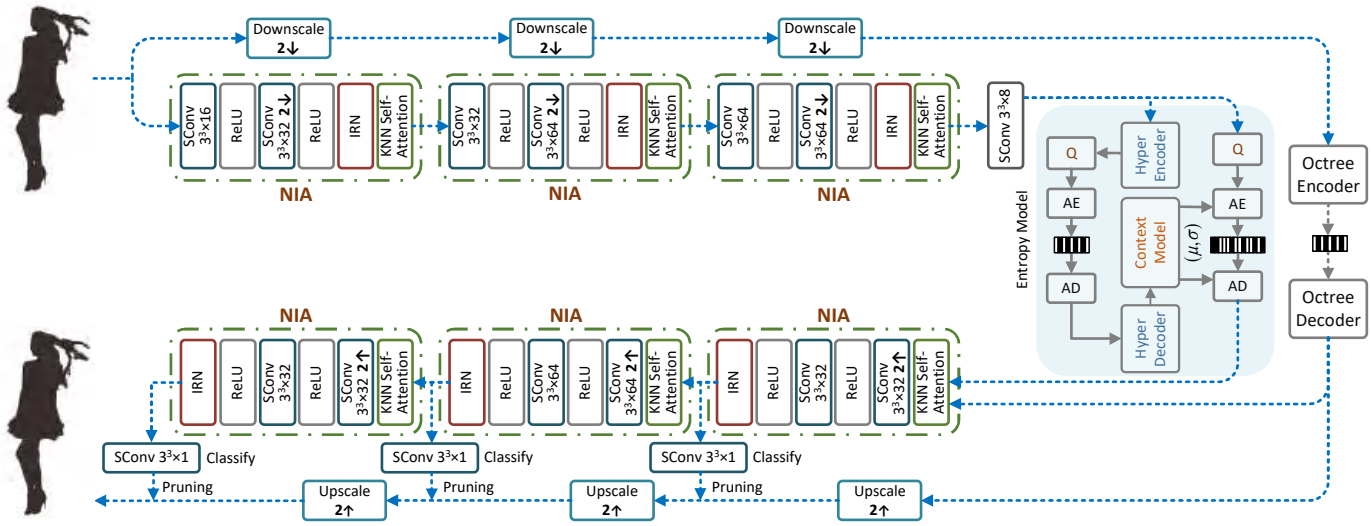
[62] S. Perry, "JPEG pleno point cloud coding common test conditions v3.2," in *87th JPEG Meeting, WG1N87037*, 2020.

[63] A. F. Guarda, N. M. Rodrigues, M. Ruivo, L. Coelho, A. Seleem, and F. Pereira, "IT/IST/IPLeiria response to the call for proposals on jpeg pleno point cloud coding," *arXiv preprint arXiv:2208.02716*, 2022.

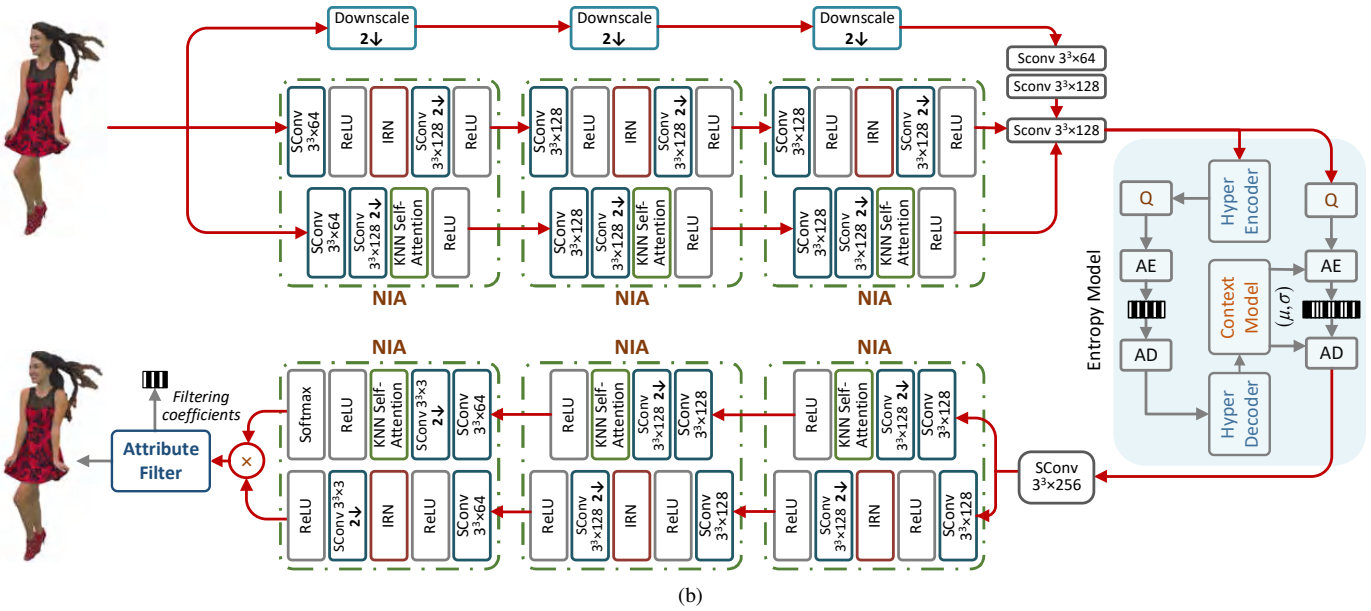
SUPPLEMENTARY MATERIAL

A. Detailed Network Architecture of DeepPCGC and DeepPCAC

The network layers of DeepPCC for geometry and attribute compression are detailed in Fig. 1a and Fig. 1b, respectively.



(a)



(b)

Fig. 1: **Network Architecture for Point Cloud Geometry Compression (PCGC) and Point Cloud Attribute Compression (PCAC) in DeepPCC**. (a) DeepPCGC. (b) DeepPCAC.

B. NIA Module

Previous discussions have demonstrated that the incorporation of NIA modules significantly contributes to the performance of DeepPCC. In this section, we will provide more details on the NIA which consists of IRN and KNN Slef-Attention.

IRN. The IRN module is essentially an inception-residual network. As depicted in Fig. 3a of our main manuscript, besides the skip connection branch, the IRN has another two parallel branches: one stacks three SConvs with the kernel size of $1 \times 1 \times 1$, $3 \times 3 \times 3$, and $1 \times 1 \times 1$, and the other stacks two $3 \times 3 \times 3$ SConvs. As such, the IRN is capable of capturing both long-range and short-range reception fields, facilitating multiple-level information collection. The skip connection additionally helps the network to converge.

Local Self-Attention. In DeepPCC, the quantized latent feature vectors obtained at the bottleneck, which contain the scale-wise aggregated information, are critical to compression performance as they are used to hierarchically reconstruct a point cloud at the decoder. Previously approaches, such as stacked sparse convolutions were mainly used for feature analysis and

aggregation. However, their information aggregation efficiency was often limited because preset fixed convolutions are unable to effectively capture sufficient points, particularly in geometry regions with low point density, and thus fail to characterize the dynamic content (*e.g.*, regions with unevenly distributed points) [61].

The DeepPCC hence introduces the local Self-Attention mechanism to tackle the above issues. It basically comprises two key components: 1) dynamic local neighborhood construction for each point using KNN search, and 2) adaptive information aggregation using Transformer-based Self-Attention over the K nearest neighbors.

Local Neighborhood Construction Using KNN Search. We introduce a KNN module to instantaneously collect K nearest neighbors of each point for local neighborhood generation. As seen, this local neighborhood is dynamically conditioned on each individual point.

As presented in Fig. 3b in our main manuscript, at the beginning of each Transformer module, K neighboring points of the current point along with their coordinates and features are gathered and combined as F_X'' of $N \times K \times d$ for further processing. Here, d denotes the number of output feature maps. Compared to previous works which input the entire point cloud into networks for processing, our KNN-based method greatly reduces the memory requirement and computational complexity. On the other hand, the nearest neighbors generally exhibit stronger correlations with the current point, whereas points located at a greater distance may present less correlation or even negative contributions. As a result, the selection of K will impact the coding performance and complexity. In this study, we set $k = 16$.

Local Information Aggregation Using Self-Attention. Subsequently, latent features including F_X and F_X'' are both sent into linear layers for computing the Q, K, and V as the conventional Transformer does, shown in Fig. 3c in our main manuscript. For every point, an attention map will be calculated. By applying the attention map to K local neighbor features, we attain the attention-weighted feature at a size of $1 \times d$ of each point. As such, the information from valid K nearest neighbors is aggregated and embedded compactly as latent features of a point. The feature vector of each point describes its spatial neighborhood embedding. For all points of the input PCG, we finally obtain attentive features with the size of $N \times d$. As illustrated in Fig. 1, features are progressively aggregated from one scale to another and eventually encoded at the bottleneck.

C. Experimental Settings of Fig. 1

Our DeepPCC is a unified lossy solution including DeepPCGC for geometry and DeepPCAC for attribute compression, which is trained using the ShapeNet dataset. The models of JPEG VM are from its opensource website. The latest G-PCC version (TMC13v19) including G-PCC (octree) for geometry compression and G-PCC (RAHT) for attribute compression is used for comparison. To ensure a fair comparison, we keep comparable bit rates between the DeepPCGC and G-PCC (octree) and comparable bit rates between the DeepPCAC and G-PCC (RAHT). Since JPEG VM devises an end-to-end model for monolithic geometry and attribute compression, the respective bit rates for geometry and attribute are unavailable. Specifically, for Fig. 1 in our main manuscript, we detail the bit rate allocation results of G-PCC and DeepPCC in Table I.

TABLE I: Bit Rate Allocation for Geometry and Attribute in G-PCC and DeepPCC

Point Cloud	Method	Geometry bit rate (bpp)	Attribute bit rate (bpp)	Total bit rate (bpp)
redandblack	G-PCC TMC13v6	0.5482	0.0921	0.6403
	G-PCC TMC13v19	0.2518	0.1084	0.3602
	DeepPCC	0.2300	0.1201	0.3501
solider	G-PCC TMC13v6	0.5464	0.0637	0.6103
	G-PCC TMC13v19	0.2311	0.0846	0.3157
	DeepPCC	0.2320	0.1043	0.3363

D. Comparison with JPEG VM

We also conduct a comparison between our proposed DeepPCC and the JPEG VM. We additionally select four point clouds sequences from JPEG Pleno Point Cloud Test database [62], including *RWT136, RWT395, RWT503b, and RWT529* (as visualized in Fig. 2).

This comparison is only intended as a reference since the JPEG VM is still under development. We limit our comparison to the JPEG test sequences since the 8iVFB sequences are used as the training dataset of JPEG VM. The comparison results, as shown in Fig. 3, reveal that our DeepPCC outperforms the JPEG VM in various distortion measurements, such as D1, YUV PSNR, and PCQM.

E. Open Source

DeepPCC is built upon several pioneering explorations as listed in Table II, which will be made publicly accessible soon at https://github.com/3dpcc.

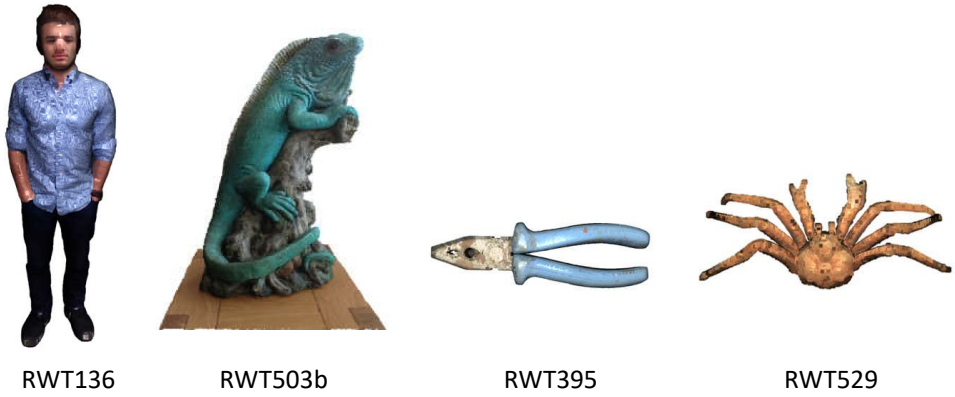


Fig. 2: **Test Sequences from JPEG Pleno Point Cloud Test database**. Four point clouds are randomly selected for testing.

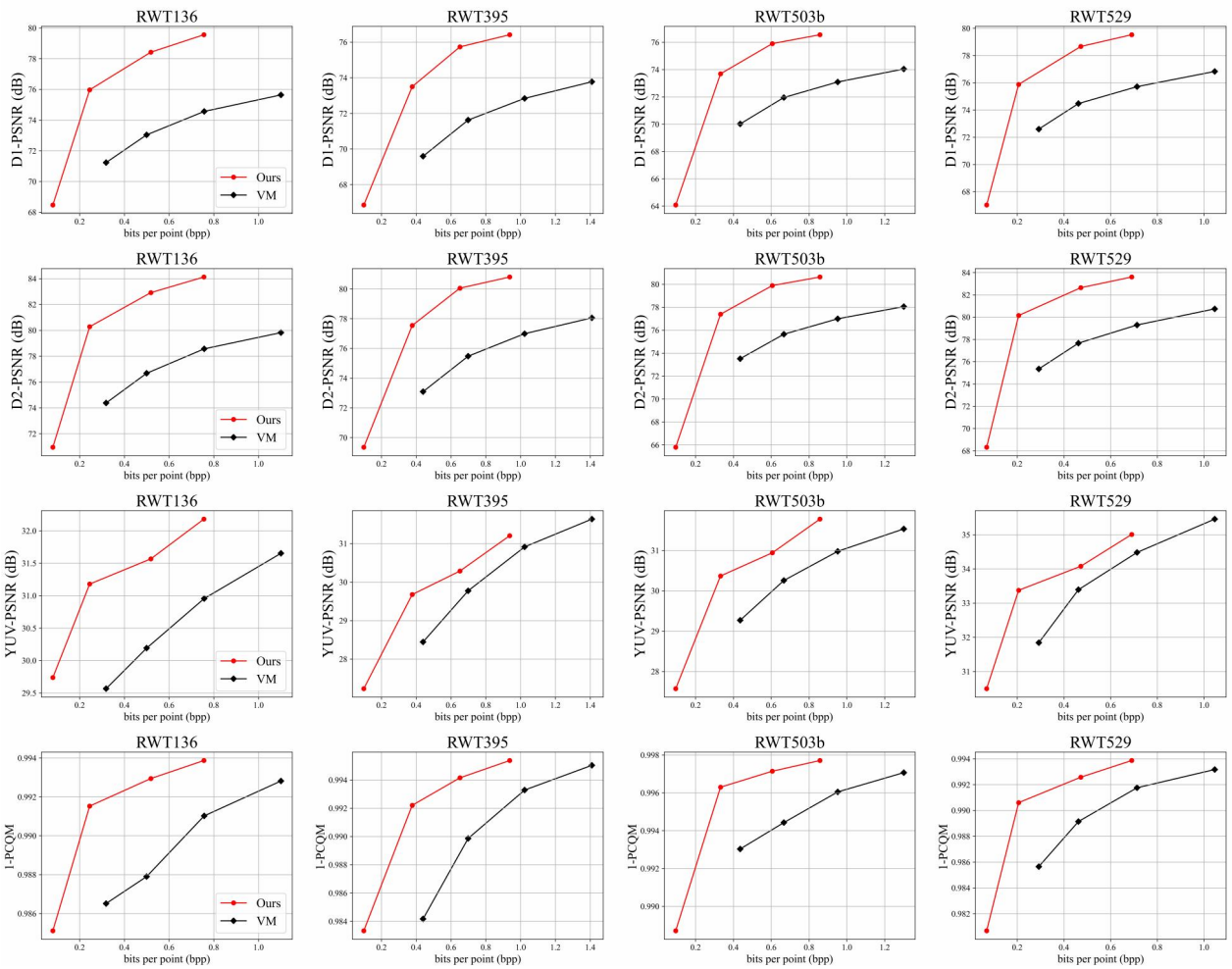


Fig. 3: **R-D Curve Comparison with JPEG VM in JPEG Test Sequences.** The JPEG VM models are from the publicly accessible website and they are pretrained following JPEG training conditions [62]. Our DeepPCC models are trained using ShapeNet.

TABLE II: Open-source Examples of Point Cloud Compression Implementations

Codec	Component	Description	Open Source
PCGCv1 [23]	Geometry	End-to-End Learning	https://njuvision.github.io/PCGCv1/
PCGCv2 [25]	Geometry	End-to-End Learning	https://njuvision.github.io/PCGCv2/
SparsePCGC [22]	Geometry	End-to-End Learning	https://github.com/NJUVISION/SparsePCGC
JPEG Pleno VM [63]	Geometry and Attribute	End-to-End Learning	https://github.com/aguarda/IT-DL-PCC
G-PCC	Geometry and Attribute	Classical Rules-based	https://github.com/MPEGGroup/mpeg-pcc-tmc13
DeepPCC	Geometry and Attribute	End-to-End Learning	https://github.com/3dpcc