Pointsoup: High-Performance and Extremely Low-Decoding-Latency Learned Geometry Codec for Large-Scale Point Cloud Scenes

Kang You[1] , **Kai Liu**[1] , **Li Yu**[2] , **Pan Gao**[1*] and **Dandan Ding**[3]
[1]Nanjing University of Aeronautics and Astronautics
[2]Nanjing University of Information Science and Technology
[3]Hangzhou Normal University
{youkang, liu-kai}@nuaa.edu.cn, li.yu@nuist.edu.cn, pan.gao@nuaa.edu.cn, DandanDing@hznu.edu.cn

Abstract

Despite considerable progress being achieved in point cloud geometry compression, there still remains a challenge in effectively compressing large-scale scenes with sparse surfaces. Another key challenge lies in reducing decoding latency, a crucial requirement in real-world application. In this paper, we propose Pointsoup, an efficient learning-based geometry codec that attains high-performance and extremely low-decoding-latency simultaneously. Inspired by conventional Trisoup codec, a point model-based strategy is devised to characterize local surfaces. Specifically, skin features are embedded from local windows via an attention-based encoder, and dilated windows are introduced as cross-scale priors to infer the distribution of quantized features in parallel. During decoding, features undergo fast refinement, followed by a folding-based point generator that reconstructs point coordinates with fairly fast speed. Experiments show that Pointsoup achieves state-of-the-art performance on multiple benchmarks with significantly lower decoding complexity, i.e., up to 90$\sim$160$\times$ faster than the G-PCCv23 Trisoup decoder on a comparatively low-end platform (e.g., one RTX 2080Ti). Furthermore, it offers variable-rate control with a single neural model (2.9MB), which is attractive for industrial practitioners.

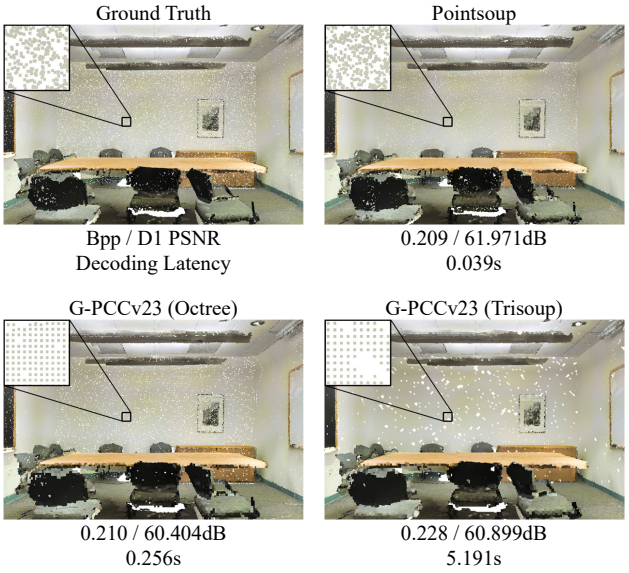

Figure 1: Quantitative compression results of proposed Pointsoup and G-PCCv23. Colors are rendered by nearest mapping. The "conferenceRoom_1" in S3DIS Area 6 is used as an example, which has 1,067,709 points. Our method allows for the decoding of a million-scale point cloud geometry in 39 ms with only one **RTX 2080Ti GPU** while guaranteeing superior visual quality.

1 Introduction

Large-scale point clouds are widely used in numerous 3D applications, such as Augmented Reality/Virtual Reality (AR/VR), autonomous driving, robotics, etc., owing to their capacity to realistically represent objects and scenes [Quach *et al.*, 2022; Abbasi *et al.*, 2023]. A large-scale point cloud typically consists of millions of sparse points, making it challenging to store and transmit, which urges the development of Point Cloud Compression (PCC) techniques.

Background. Two international PCC standards, i.e., Video-based PCC (V-PCC) and Geometry-based PCC (G-PCC), were developed under the Moving Picture Experts Group (MPEG) and released as part of ISO/IEC 23090-5 and 23090-9 [Chen *et al.*, 2023; WG 7, 2020; WG 7 and Coding, 2023]. The octree representation [Schnabel and Klein, 2006] is adopted in G-PCC to efficiently encode the geometry information. Alternatively, the Trisoup geometry codec is a viable option in G-PCC to perform lossy geometry compression more effectively by estimating local point cloud surface as triangle meshes.

Recently, owing to the significant gains obtained by learning-based approaches, both MPEG and Joint Photographic Experts Group (JPEG) committees have launched explorations on Artificial Intelligence (AI) based PCC solutions. Despite the powerful performance demonstrated in the compression of dense point cloud objects, learning-based methods [Wiesmann *et al.*, 2021; Fu *et al.*, 2022; You *et al.*, 2022; Huang *et al.*, 2022; Wang *et al.*, 2022; Zhang *et al.*, 2023a; Song *et al.*, 2023] still face the follow-

*Corresponding author

ing two major challenges in compressing sparse point cloud scenes: 1) Unsatisfactory compression performance. The variability of sparse surfaces can make the neural model easily collapse; 2) High decoding power demands that restrict the application scope. We argue that the computational complexity for decoding should be rather low, in order to adapt to different computing-power client devices as well as live-streaming applications.

Our Approach. To address the above issues, we propose Pointsoup, an efficient learning-based geometry codec that attains high-performance and extremely low-decoding-latency simultaneously. Specifically, we first design the Aligned Window-based Down-Sampling (AWDS) module, which allows for the learned embedding of local surfaces, leveraging an effective attention-based aggregation. We then devise the Dilated Window-based Entropy Modeling (DWEM) module to aggregate dilated windows, which are built upon down-sampled bones, to estimate the distribution of quantized features in parallel. Finally, a fast feature refinement block is intergrated with an efficient folding-based point generator, in the Dilated Window-based Up-Sampling (DWUS) module, to reconstruct the local surface with fast speed. Moreover, the Pointsoup provides variable-rate control with a single neural network model, by fully exploiting the flexibility of the point-based pipeline.

Contribution. Main contributions can be summarized as:

- Leveraging a point model that harnesses an attention-based encoder and the dilated window-based entropy modeling, our method achieves state-of-the-art efficiency on multiple large-scale benchmarks.
- By designing a fast feature refinement block followed by an efficient folding-based point generator, our method achieves extremely low-decoding latency. It enables nearly real-time decoding for million-scale point clouds, with up to 90∼160× faster than the G-PCCv23 Trisoup decoder on only one RTX 2080Ti GPU.
- Our method shows strong generalization capability and offers flexible bitrate control with a lightweight neural model, which is beneficial for practical applications.

2 Related Work

Numerous works have contributed to the Point Cloud Geometry Compression (PCGC) task, which can be roughly divided into two categories: voxel models and point models.

Voxel Model. Considering the sparsity of the point cloud geometry, original Point Cloud Geometry (PCG) can be reorganized to octree structure [Schnabel and Klein, 2006; Fu *et al.*, 2022; Song *et al.*, 2023] or multi-scale sparse representation [Wang *et al.*, 2021a; Wang *et al.*, 2022; Zhang *et al.*, 2023a]. The octree iteratively divides the occupied space to produce an efficient tree-structured format, which is adopted by the well-known MPEG G-PCC standard [WG 7 and Coding, 2023] for its effectiveness and scalability. As another optional codec, Trisoup models the surface of the point cloud as a series of triangle meshes, which yields superior compression performance, but at the expense of high computational cost. The sparse tensor-based approach [Wang *et al.*, 2022; Zhang *et al.*, 2023a; Fan *et al.*, 2023] utilizes multi-scale sparse representation and delivers significant compression gains. However, stacked convolutional layers still impose substantial computational demands, limiting their application scenarios.

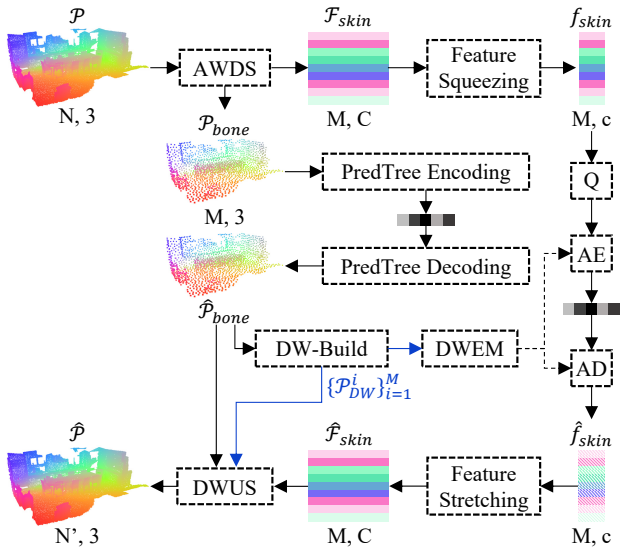

Figure 2: Pointsoup workflow. AWDS refers to the Aligned Window-based Down-Sampling module; DWEM denotes the Dilated Window-based Entropy Modeling module; DWUS represents the Dilated Window-based Up-Sampling module; AE and AD are for arithmetic encoding and decoding; Q denotes quantization.

Point Model. The past several years have witnessed the emergence of point-based techniques [Guo *et al.*, 2021; Wu *et al.*, 2022; Vinodkumar *et al.*, 2023], which promoted the development of point models for learned PCGC tasks. Some works explored small-scale PCC techniques, but generally lack the applicability to large-scale point clouds [You and Gao, 2021; Zhang *et al.*, 2022]. Other point models for large-scale point clouds, on the other hand, struggle to achieve a competitive trade-off between compression efficiency and computing overhead [He *et al.*, 2022; You *et al.*, 2022; Huang *et al.*, 2022; Huang and Wang, 2023]. For instance, 3QNet [Huang *et al.*, 2022] devised the Model Breaking Strategy (MBS) to divide point cloud into blocks first, but the MBS can lead to significant gaps between blocks, which significantly diminishes visual quality. As a real-time codec designed for dense point maps, Depoco [Wiesmann *et al.*, 2021] reports a low coding complexity, but it suffers from severe quality degradation.

3 Methodology

3.1 Framework

The overall workflow of our proposed Pointsoup is shown in Fig. 2. Specifically, the surface of the input point cloud $\mathcal{P} \in \mathbb{R}^{N\times 3}$ is embedded to skin features $\mathcal{F}_{skin} \in \mathbb{R}^{M\times C}$ by the Aligned Window-based Down-Sampling (AWDS) module, and the down-sampled bones are then instantly encoded and decoded, through the predictive tree of G-PCC. The Dilated Window-based Entropy Modeling (DWEM) module is

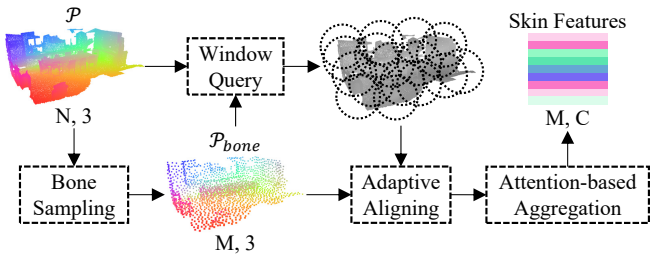

Figure 3: Aligned Window-based Down-Sampling (AWDS) module.

used to estimate the distribution of compacted skin features, by the dilated windows $\{\mathcal{P}^i_{DW}\}^M_{i=1}$ that derived from the decoded bones $\hat{\mathcal{P}}_{bone} \in \mathbb{R}^{M\times 3}$. A Dilated Window-based Up-Sampling (DWUS) module is devised to reconstruct the local surface from decoded skin features and bones. The next few subsections will detail the above-mentioned modules.

3.2 Aligned Window-based Down-Sampling (AWDS)

The AWDS module is devised to identify a well-spread skeleton and characterize the local surface into skin features, as shown in Fig. 3.

Bone Sampling and Window Query

The sampling and querying methods in the Pointsoup inherit the widely used aggregation basis in point cloud analysis tasks [Zhao *et al.*, 2021; Zhang *et al.*, 2023b; Li *et al.*, 2023a], which first use the Farthest Point Sampling (FPS) to derive the skeleton $\mathcal{P}_{bone} \in \mathbb{R}^{M\times 3}$ and then build a K-Nearest Neighbor (KNN) graph to formulate overlapping local windows. Particularly, due to the high computational cost of FPS, we first use the Random Point Sampling (RPS) to obtain a subset with no more than $M \times 16$ points, and then apply FPS on the subset to derive the result $\mathcal{P}_{bone}$ with M points.

Adaptive Aligning

We align the obtained overlapping local windows to facilitate network learning and enhance density adaptability. Each window is first shifted to the coordinate origin and then rescaled according to the skeleton density. Mathematically,

$$d = \frac{1}{|\mathcal{P}_{bone}|} \sum_{p_i \in \mathcal{P}_{bone}} \min_{p_j \in \mathcal{P}_{bone}} \left\{ \|p_i - p_j\|_2 : p_i \neq p_j \right\} \quad (1)$$

$$\mathcal{P}^i_{AW} = \left\{ \frac{p - p_i}{d} : p \in \mathcal{N}(p_i, \mathcal{P}, K) \right\}, \forall p_i \in \mathcal{P}_{bone} \quad (2)$$

where $\mathcal{P}^i_{AW}$ refers to the aligned local window, and $\mathcal{N}(p_i, \mathcal{P}, K)$ represents finding K nearest neighbors of the point p_i on the input point cloud $\mathcal{P}$.

Attention-based Aggregation

Given an aligned window $\mathcal{P}^i_{AW} \in \mathbb{R}^{K\times 3}$, we embed the local surface into a high dimensional feature vector $\mathcal{F}^i_{skin} \in \mathbb{R}^{1\times C}$, by an effective attention-based neural network, as illustrated in Fig. 4.

Specifically, we first perform a mini embedding on each point within the window to produce feature $\mathcal{F}^i_{(0)} \in \mathbb{R}^{K\times C}$

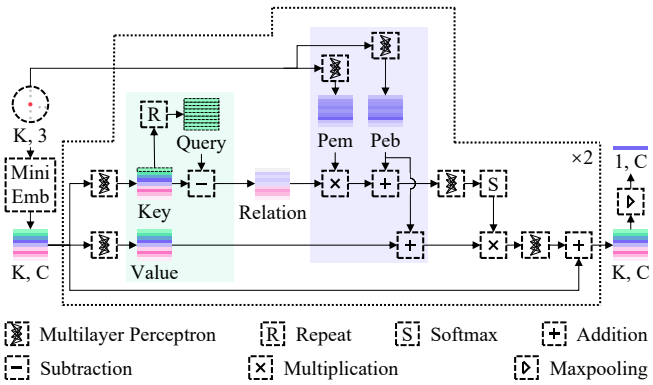

Figure 4: Attention-based aggregation of AWDS module. The self-attention block is presented in the dotted line.

for the local details, based on a pervasive graph convolution (GraphConv) operation:

$$\mathcal{F}^i_{(0)}[j] = \text{GraphConv}\left(\mathcal{N}\left(p^i_j, \mathcal{P}^i_{AW}, k_m\right)\right), \forall p^i_j \in \mathcal{P}^i_{AW} \quad (3)$$

where $\mathcal{N}(p_j, \mathcal{P}^i_{AW}, k_m)$ represents finding k_m nearest neighbors of the point p_j on the aligned window $\mathcal{P}^i_{AW}$; GraphConv is defined as GraphConv(·)=MaxPool(MLP(·)); $\mathcal{F}^i_{(0)}[j] \in \mathbb{R}^{1\times C}$ refers to the jth feature vector (corresponding to the point p^i_j) in the feature matrix $\mathcal{F}^i_{(0)} \in \mathbb{R}^{K\times C}$.

Then, self-attention blocks are stacked following the subtraction vector attention [Wu *et al.*, 2022]. It should be noted that since the local window is generated by KNN query, the first row of the Key matrix always represents the feature that attached to the center point of the window. Therefore, the $Query$ matrix is constructed by repeating the first row of the Key, producing the subtraction relation anchored at the window center. Then, the process of the lth attention block can be described as follows:

$$\mathcal{S}^i_{(l)} = \varrho\left(\text{MLP}\left(\left(\mathcal{K}^i_{(l)} - \mathcal{Q}^i_{(l)}\right) \times Pem^i_{(l)} + Peb^i_{(l)}\right)\right) \quad (4)$$

$$\mathcal{F}^i_{(l+1)} = \mathcal{F}^i_{(l)} + \text{MLP}\left(\left(\mathcal{V}^i_{(l)} + Peb^i_{(l)}\right) \times \mathcal{S}^i_{(l)}\right) \quad (5)$$

where ϱ means the Softmax operation; Pem and Peb refer to the positional encoding multiplier and bias; $Query$, Key, and $Value$ are abbreviated as $\mathcal{Q}$, $\mathcal{K}$, and $\mathcal{V}$, for a concise explanation.

Finally, the feature $\mathcal{F}^i_{(L)} \in \mathbb{R}^{K\times C}$ output by the final self-attention block is aggregated to the skin feature $\mathcal{F}^i_{skin} \in \mathbb{R}^{1\times C}$, by a max-pooling operation:

$$\mathcal{F}^i_{skin} = \text{MaxPool}\left(\mathcal{F}^i_{(L)}\right) \quad (6)$$

We set k_m=16 and L=2 in our experiment.

3.3 Dilated Window-based Entropy Modeling (DWEM)

Considering the significant dependency among the target local window and the nearby area of the down-sampled skeleton, dilated window is introduced as the cross-scale prior to approximate the distribution of the skin feature. Meanwhile, the skin feature is further squeezed, yielding a compact representation for fast arithmetic coding.

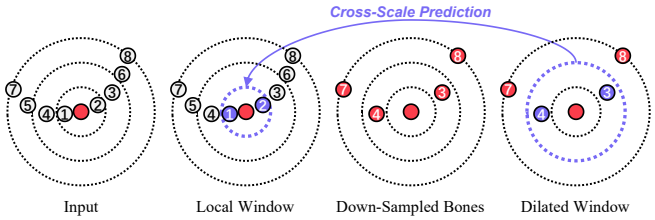

Figure 5: Dilated window-based entropy modeling. The dilated window, obtained by computing the k nearest neighbors upon down-sampled bones, is introduced as a cross-scale prior.

Dilated Window Construction

As shown in Fig. 5, a local window is dilated by employing the KNN graph on the down-sampled bones. Recall that the bones are handled by G-PCC at the base level, which makes dilated neighborhoods readily accessible to serve as the cross-scale prior. Mathematically, a dilated window $\mathcal{P}^i_{DW}$ is defined as:

$$\mathcal{P}^i_{DW} = \mathcal{N}(p_i, \hat{\mathcal{P}}_{bone}, k), \quad \forall p_i \in \hat{\mathcal{P}}_{bone} \tag{7}$$

where $\mathcal{P}^i_{DW} \in \mathbb{R}^{k\times 3}$, $\mathcal{N}$ represents the k nearest neighbors of the down-sampled point p_i on the decoded skeleton $\hat{\mathcal{P}}_{bone}$. k is set to 8 in our experiments.

Feature Compaction

Higher dimensional features lead to an increase in arithmetic coding complexity, due to the presence of longer symbol sequence to be coded. Thus, a squeezing operation is used by leveraging a simple fully-connected layer. Mathematically,

$$f^i_{skin} = \text{Linear}\left(\mathcal{F}^i_{skin}\right) \tag{8}$$

where $\mathcal{F}^i_{skin} \in \mathbb{R}^{1\times C}$, $f^i_{skin} \in \mathbb{R}^{1\times c}$. Note that the skin features are stretched back by another Linear layer after arithmetic decoding. We set C=128 and c=16 in our experiments.

Cross-Scale Entropy Modeling

A uniform scalar quantizer is used in this work, which is replaced with an additive uniform noise during training [Ballé *et al.*, 2016; Jamil *et al.*, 2023]. Let the quantized skin features be $\tilde{f}_{skin} = Q\left(f_{skin}\right)$, then it is further modeled as:

$$P_\theta\left(\tilde{f}_{skin}\right) = \prod_{i=1}^{M}\left(\mathcal{L}\left(\Phi^i\right) * \mathcal{U}\left(-\frac{1}{2}, \frac{1}{2}\right)\right)\left(\tilde{f}^i_{skin}\right) \tag{9}$$

where P_θ represents the entropy model parameterized by θ, $\mathcal{L}\left(\Phi^i\right)$ refers to the Laplacian distribution of quantized feature $\tilde{f}_{skin}$ with parameter $\Phi^i = (\mu^i, \sigma^i)$, and $\mathcal{U}\left(-\frac{1}{2}, \frac{1}{2}\right)$ denotes the uniform distribution ranging from $\left[-\frac{1}{2}, \frac{1}{2}\right]$. Here, the parameter Φ^i can be estimated from dilated window by a network that contains a GraphConv layer and a regression head MLP:

$$\Phi^i = \left(\mu^i, \sigma^i\right) = \text{MLP}\left(\text{GraphConv}\left(\mathcal{P}^i_{DW}\right)\right) \tag{10}$$

Finally, the expected bit rate for the skin features can be written as:

$$\mathcal{R}_{skin} = -\frac{1}{N}\log_2 P_\theta\left(\tilde{f}_{skin}\right) \tag{11}$$

where N denotes the number of points of the input point cloud.

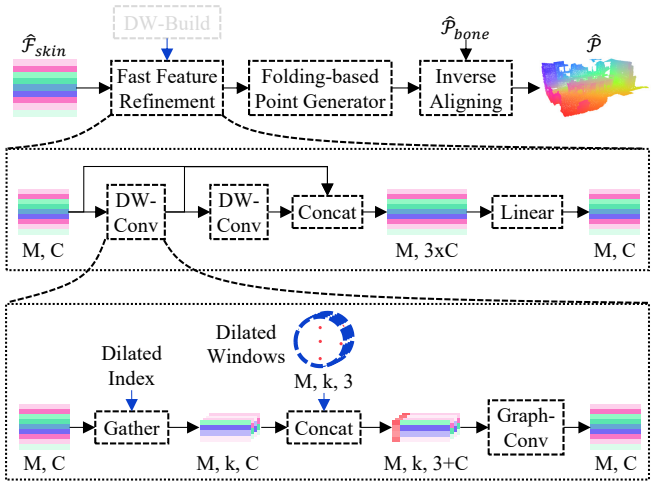

Figure 6: Dilated Window-based Up-Sampling (DWUS) module.

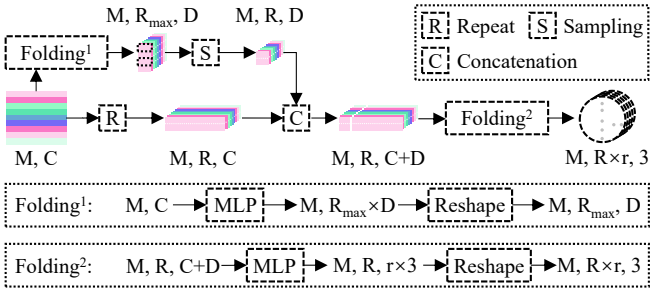

Figure 7: The proposed folding-based point generator.

3.4 Dilated Window-based Up-Sampling (DWUS)

Unlike the progressive multi-scale fashion used in the previous point models [Huang *et al.*, 2022; He *et al.*, 2022], we fully exploit the *single-scale* strategy to reduce the complexity: First, the skin features that are attached to the skeleton undergo fast refinement, owing to the small quantity of features M, which is orders of magnitude less than the original input scale N; Then, lightweight Folding operation is devised to generate point coordinates based on shallow MLPs.

Fast Feature Refinement

Figure 6 details the fast feature refinement block, where the Dilated Window-based Convolution (DWConv) is introduced to integrate information from dilated windows. The reuse of the built dilated window avoids the recomputation of the spatial graph [Wang *et al.*, 2019; Li *et al.*, 2023b], thus provides an economized graph structure for feature convolution. To be specific, features of the points within the corresponding dilated window are gathered into groups based on the provided dilated index. Then, GraphConv is applied to aggregate the gathered groups to produce refined features.

Folding-based Point Generator

Backed by the refined features, a lightweight folding-based point generator is devised to generate point coordinates with fast speed, as shown in Fig. 7. To be specific, we first define the Folding operation as a combination of MLP and re-

shape, where MLP is used to upscale the input features and reshape operation is used to fold the output dimensions. The first Folding transforms each skin feature from a $1 \times C$ vector to a grid matrix of $R_{max} \times D$. Then, the rows of the grid will be randomly sampled to the shape of $R \times D$, where $R \in [1, R_{max}]$. The sampling technique allows an adjustable number of generated points, which is crucial for the variable-rate control mechanism, as will be detailed in Sec. 3.5. Then, the down-sampled feature grid is concatenated with the input skin features, followed by another Folding function to generate point coordinates.

Inverse Aligning

The inverse aligning operation mirrors the adaptive aligning used in the encoder. Each reconstructed window $\hat{\mathcal{P}}^i_{AW}$ is shifted to the original position and rescaled back, to assemble into a completed reconstructed result $\hat{\mathcal{P}}$. Mathematically,

$$\hat{\mathcal{P}} = \bigcup_{\hat{p}_i \in \hat{\mathcal{P}}_{bone}} \left\{ \left(\hat{p} \times \hat{d} \right) + \hat{p}_i : \hat{p} \in \hat{\mathcal{P}}^i_{AW} \right\} \tag{12}$$

where $\hat{d}$ represents the scale factor recalculated by $\hat{\mathcal{P}}_{bone}$, as described in Eq. 1.

3.5 Variable-Rate Control

A single-model-variable-rate solution is suggested based on the local window size that can be flexibly handled by the point model. Similar to the node-size adjustment of the G-PCC Trisoup codec, we modulate the size K of the queried local windows to adapt to different bit rates. Inspired by [You *et al.*, 2022], we set $M = \lfloor \frac{2N}{K} \rfloor$, i.e., a denser skeleton is grown for smaller windows to capture finer details at higher bitrate budget, while a sparser skeleton is presented with larger windows to naturally reduce the bit rate. At the decoder, we employ the feature sampling technique to reconstruct the window under the given size K, by adapting the parameter R of the point generator to $\lfloor \frac{K}{r} \rfloor$, where r is fix to 4 in our implementation.

3.6 Loss Function

We follow the conventional rate-distortion trade-off as our loss function:

$$\mathcal{L} = \mathcal{D}_{CD}(\mathcal{P}, \hat{\mathcal{P}}) + \lambda \mathcal{R}_{skin} \tag{13}$$

where $\mathcal{D}_{CD}(\mathcal{P}, \hat{\mathcal{P}})$ refers to the Chamfer Distance between input point cloud $\mathcal{P}$ and reconstructed point cloud $\hat{\mathcal{P}}$, $\mathcal{R}_{skin}$ refers to the bit rate as described in Eq. 11.

4 Experiments

4.1 Experimental Setup

Training Dataset. We limit the training process on the ShapeNet [Chang *et al.*, 2015] training set, which consists of 35,708 point clouds, each generated by uniformly sampling 8k points from a CAD model.

Test Dataset. Both large-scale indoor scenes and outdoor maps are considered for testing. The indoor point cloud scenes includes Stanford Large-Scale 3D Indoor Spaces Dataset (S3DIS) [Armeni *et al.*, 2016] and ScanNet [Dai *et al.*, 2017]. The outdoor point cloud maps are generated from KITTI [Behley *et al.*, 2019], following [Wiesmann *et al.*, 2021]. Details of used test sets are provided in Tab. 1.

Dataset	Test Split	#Point Cloud	#Points per Point Cloud
S3DIS	Area 6	48	3.2M∼0.3M
ScanNet	Official test set	100	553k∼32k
KITTI	Sequence 08	186	554k∼214k

Table 1: Details for test data set used in this paper.

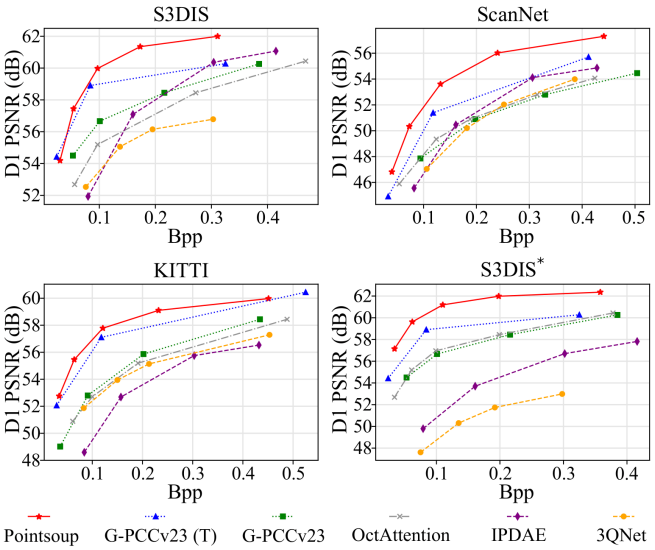

Figure 8: Rate-distortion performance comparison on S3DIS, ScanNet, and KITTI. "S3DIS*" refers to the evaluation of the models that are trained on Area 1∼5, instead of ShapeNet, and tested on Area 6.

Settings. We implement our model using Python 3.10 and Pytorch 2.0. Adam optimizer is used with an initial learning rate of 0.0005 and a batch size of 1. We train our model only once for 140,000 steps under the local window size of 128. The λ that balances the rate and distortion is set to 10^{-4}. Down-sampled bones are compressed losslessly by G-PCC predictive tree. All experiments are conducted on an Intel Core i9-9900K CPU and one RTX 2080Ti GPU.

Benchmarking Baselines. We compare our method with state-of-the-art rules-based methods: the default Octree codec and improved Trisoup codec of the lastest G-PCCv23 [WG 7 and Coding, 2023], which are denoted as "G-PCCv23" and "G-PCCv23 (T)", respectively; and learning-based methods: OctAttention [Fu *et al.*, 2022], IPDAE [You *et al.*, 2022], and 3QNet [Huang *et al.*, 2022]. All learning-based methods are retrained on the same dataset as our method, and all test samples are normalized to the coordinate range of [0, 1023] (a.k.a., 10-bit precision) for ease of fair comparison. In addition, we compare Depoco [Wiesmann *et al.*, 2021] and D-PCC [He *et al.*, 2022] following their recommended test conditions in Sec. 4.4.

4.2 Quantitative Comparison

Rate-Distortion Performance. Figure 8 shows the rate-distortion curves of different methods and Tab. 2 demonstrates the quantitative results using BD-PSNR and BD-Rate

Dataset	Metric	G-PCCv23	G-PCCv23 (T)	OctAttention	IPDAE	3QNet	Pointsoup
S3DIS	BD-PSNR (dB)	-	+2.256	-1.085	-0.745	-2.461	**+3.229**
	BD-Rate (%)	-	**-67.712**	+38.385	+25.838	+138.569	-60.067
ScanNet	BD-PSNR (dB)	-	+2.477	+0.234	+0.613	-0.233	**+4.195**
	BD-Rate (%))	-	-46.310	-6.038	-14.023	+4.122	**-60.302**
KITTI	BD-PSNR (dB)	-	+3.093	-0.430	-2.222	-0.891	**+3.392**
	BD-Rate (%)	-	-61.517	+12.344	+87.928	+27.486	**-64.105**
Avg. Time (s/frame)		Enc / Dec	Enc / Dec	Enc / Dec	Enc / Dec	Enc / Dec	Enc / Dec
S3DIS		**0.334** / 0.126	11.477 / 2.143	0.672 / 92.055	20.846 / 1.224	24.248 / 0.401	8.149 / **0.022**
ScanNet		**0.060** / 0.027	7.365 / 1.122	0.109 / 12.807	4.641 / 0.237	2.255 / 0.156	2.833 / **0.006**
KITTI		**0.110** / 0.048	8.508 / 1.819	0.231 / 26.658	6.944 / 0.459	5.865 / 0.258	3.236 / **0.011**

Table 2: Quantitative results using BD-PSNR and BD-Rate metrics. G-PCCv23 serves as the anchor. The **best** and second-best results are highlighted in bold and underlined, respectively.

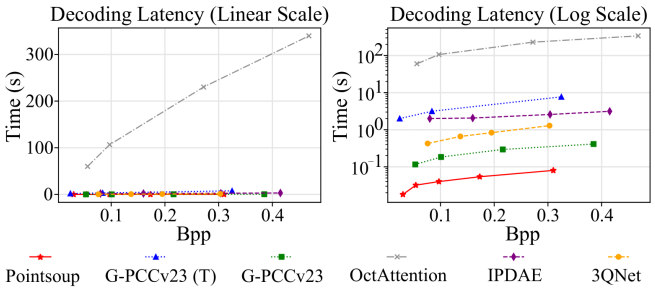

Figure 9: Decoding time comparison. We illustrate the time consumption at different bit rates, where each data point refers to the average decoding time for all test scenes in S3DIS Area 6.

Metric	Conditions of Depoco		Conditions of D-PCC	
	Pointsoup	Depoco	Pointsoup	D-PCC
BD-PSNR (dB)	**+3.392**	-2.061	**+4.180**	-5.480
BD-Rate (%)	**-64.105**	+54.037	**-65.682**	+165.138
Enc. Time (s)	3.234	**0.131**	1.480	**0.646**
Dec. Time (s)	0.011	**0.002**	**0.006**	0.165

Table 3: Compression efficiency comparison on the test conditions of Depoco and D-PCC. G-PCCv23 serves as the anchor.

metrics. It can be seen that the proposed Pointsoup achieves the best rate-distortion performance, providing 60%∼64% bitrate reduction over the G-PCCv23 anchor.

Subjective Visual Quality. Figure 1 shows visualized results for an example indoor scene, where our method reconstructs the point cloud surface with a uniform point distribution that shares the same nature as the ground truth. An example of an outdoor point cloud map is presented in Fig. 10, in which finer details can be observed in our output (e.g., higher fidelity vehicle shapes and vegetation outlines).

Computational Complexity. As can be seen from Fig. 9 and Tab. 2, our method reports the lowest decoding latency, e.g., 90∼160× faster than the G-PCCv23 Trisoup decoder and 3∼5× faster than the Octree decoder. Moreover, the encoding time of the proposed Pointsoup is also significantly faster than the Trisoup, presenting a manageable encoding complexity. In addition, our network is fairly small with 761k parameters (about 2.9MB), which is much lighter than other point models such as 3QNet (85MB) and IPDAE (68∼516MB for each bitrate point).

4.3 Customized Training Domain

The training process in the above section is limited to the ShapeNet dataset. Intuitively, it is worth considering a training domain that closely resembles the test scenario to enhance reconstruction accuracy. Therefore, we retrain each model on S3DIS Area 1∼5 and test them on Area 6 again. As seen from Fig. 8 (S3DIS*), the performances of Pointsoup and OctAttention are significantly improved, due to the similar patterns shared between training and test samples. Unexpectedly, both IPDAE and 3QNet exhibit a severe degradation, possibly indicating their inadequate capacity in handling complex training samples [Wang *et al.*, 2021b].

4.4 Scenario Dependent Comparison

Depoco [Wiesmann *et al.*, 2021] and D-PCC [He *et al.*, 2022] are representative point models for large-scale point cloud compression. However, they do not support the generalization from a small-scale training set (e.g., ShapeNet) to large-scale test frames. In this subsection, we follow their training/testing conditions for a comparative study, i.e., we verify our model using the respective training and test datasets as suggested in their papers. As seen from Tab. 3, our method significantly outperforms Depoco and D-PCC in terms of rate-distortion performance. It is worth noting that, despite the faster coding speed of Depoco, it comes at the expense of severely compromised reconstruction quality.

4.5 Ablation Study

Adaptive Aligning. It is imperative to consider the adaptation of the point densities as they vary with the number of points and the size of the scenes. The quantity-based aligning [You *et al.*, 2022] is another reasonable way of density adaptation, but it only considers the influence of the number of points, while neglecting the impact of the spatial volume of the point cloud. For instance, they do not adapt well to narrow hallways, as evidenced in Fig. 11. On the contrary, we

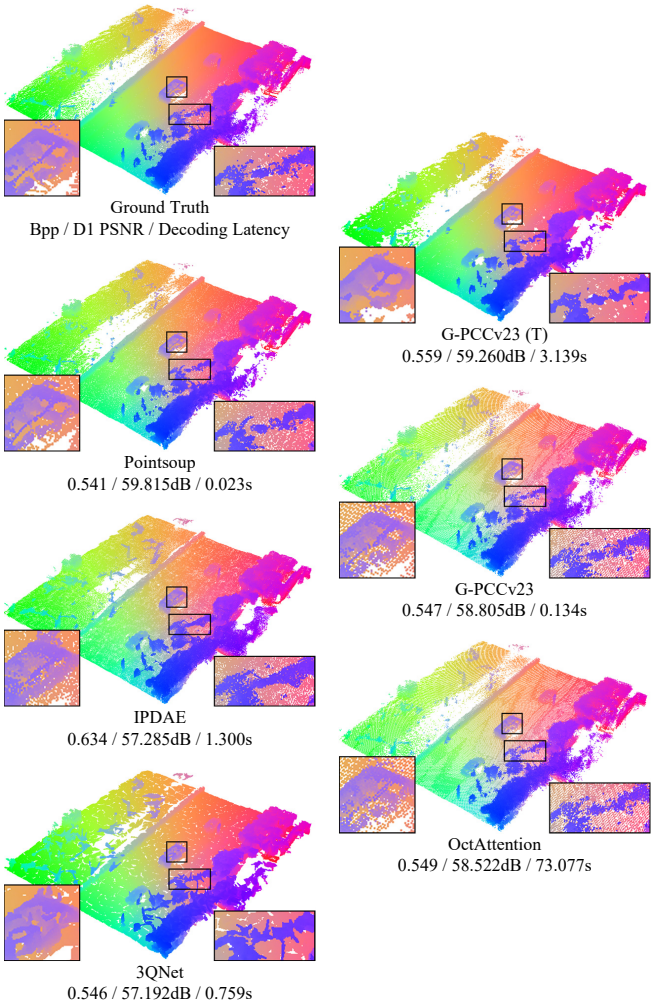

Figure 10: Reconstruction visualization of an example dense point cloud map in KITTI sequence 08.

ATTN	DWEM	FFR	BD-PSNR (dB)	BD-Rate (%)
✗	✓	✓	-0.385	+14.647
✓	✗	✓	-1.128	+47.243
✓	✓	✗	-0.210	+7.588

Table 4: Ablation study for network components. "ATTN" refers to the attention block used in the AWDS module. "DWEM" refers to the dilated window-based entropy modeling. "FFR" refers to the fast feature refinement block of the DWUS module. Fully armed Pointsoup serves as the anchor. Models are tested on S3DIS.

	BD-Rate (%)	Dec (ms)	AD (ms)
w/o Feature Compaction	-54.820	42	22
Pointsoup	**-60.067**	**22**	**4**

Table 5: Ablation study on feature compaction. S3DIS Area 6 is used for test. G-PCCv23 serves as the anchor. AD refers to the time of arithmetic decoding.

infer local densities from down-sampled bones, which provides better aligned results by exploiting cross-scale priors.

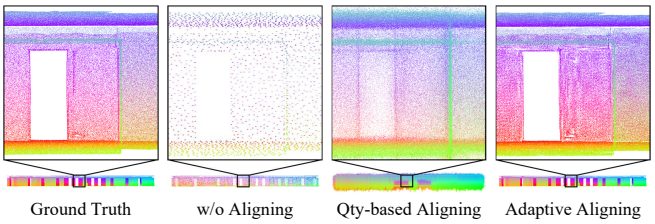

Figure 11: Comparison of proposed adaptive aligning and quantity (Qty)-based aligning.

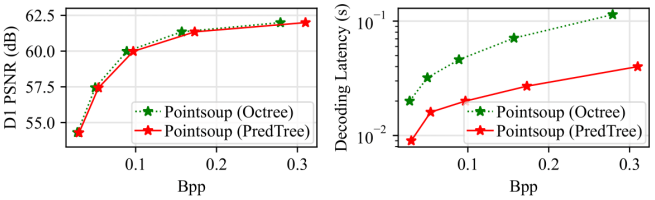

Figure 12: Comparison between using predictive tree and octree codec for bones. S3DIS Area 6 is used for test.

Predictive Tree vs. Octree. Octree is another optional codec to compress down-sampled bone points. However, since predictive tree is particularly designed for low complexity decoding, using the octree will decelerate the decoding speed (about 2∼3× slower), as shown in Fig. 12, despite some rate-distortion benefits (7.979% BD-BR gain).

Network Components. As shown in Tab. 4, several key modules are disabled individually to examine the validity of the components. Note that the attention block is changed to MLP, and the DWEM is replaced by the basic factorized prior model [Ballé *et al.*, 2016] during study. Results show that disabling the DWEM module will lead to 47.243% BD-Rate loss relative to the original Pointsoup, which demonstrates the significant efficiency of the cross-scale entropy modeling.

Feature Compaction. We use a linear layer to squeeze skin feature to a compact representation, which speed up the arithmetic coding of the features. As shown in Tab. 5, the feature compaction operation greatly improves the speed of the decoding, reducing the arithmetic decoding time from 22 ms to 4 ms. In addition, the compaction yields a more efficient representation, leads to a slight bitrate reduction.

Please refer to our supplementary material for more comparisons and ablation studies.

5 Conclusion

This paper proposes an efficient learning-based geometry codec, dubbed Pointsoup, aiming at large-scale point cloud scenes. The Pointsoup demonstrates state-of-the-art compression efficiency with significantly low decoding latency and variable-rate controllability, making it a promising option for AI-based PCC solutions. Downstream tasks (e.g., object detection) may consider working directly on compressed domain without the need of a complete reconstruction. Source code and supplementary material are available at https://github.com/I2-Multimedia-Lab/Pointsoup.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 62272227).

References

[Abbasi *et al.*, 2023] Rashid Abbasi, Ali Kashif Bashir, Hasan J. Alyamani, Farhan Amin, Jaehyeok Doh, and Jianwen Chen. Lidar point cloud compression, processing and learning for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 24(1):962–979, 2023.

[Armeni *et al.*, 2016] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2016.

[Ballé *et al.*, 2016] Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016.

[Behley *et al.*, 2019] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.

[Chang *et al.*, 2015] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[Chen *et al.*, 2023] Anthony Chen, Shiwen Mao, Zhu Li, Minrui Xu, Hongliang Zhang, Dusit Niyato, and Zhu Han. An introduction to point cloud compression standards. *GetMobile: Mobile Computing and Communications*, 27(1):11–17, 2023.

[Dai *et al.*, 2017] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[Fan *et al.*, 2023] Tingyu Fan, Linyao Gao, Yiling Xu, Dong Wang, and Zhu Li. Multiscale latent-guided entropy model for lidar point cloud compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 33(12):7857–7869, 2023.

[Fu *et al.*, 2022] 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*, volume 36, pages 625–633, 2022.

[Guo *et al.*, 2021] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7:187–199, 2021.

[He *et al.*, 2022] Yun He, Xinlin Ren, Danhang Tang, Yinda Zhang, Xiangyang Xue, and Yanwei Fu. Density-preserving deep point cloud compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2333–2342, 2022.

[Huang and Wang, 2023] Runnan Huang and Miaohui Wang. Patch-wise lidar point cloud geometry compression based on autoencoder. In *International Conference on Image and Graphics*, pages 299–310. Springer, 2023.

[Huang *et al.*, 2022] Tianxin Huang, Jiangning Zhang, Jun Chen, Zhonggan Ding, Ying Tai, Zhenyu Zhang, Chengjie Wang, and Yong Liu. 3qnet: 3d point cloud geometry quantization compression network. *ACM Transactions on Graphics (TOG)*, 41(6):1–13, 2022.

[Jamil *et al.*, 2023] Sonain Jamil, Md Jalil Piran, MuhibUr Rahman, and Oh-Jin Kwon. Learning-driven lossy image compression: A comprehensive survey. *Engineering Applications of Artificial Intelligence*, 123:106361, 2023.

[Li *et al.*, 2023a] Shanshan Li, Pan Gao, Xiaoyang Tan, and Mingqiang Wei. Proxyformer: Proxy alignment assisted point cloud completion with missing part sensitive transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9466–9475, 2023.

[Li *et al.*, 2023b] Zihao Li, Pan Gao, Hui Yuan, and Ran Wei. Dynamic local feature aggregation for learning on point clouds. *arXiv preprint arXiv:2301.02836*, 2023.

[Quach *et al.*, 2022] Maurice Quach, Jiahao Pang, Dong Tian, Giuseppe Valenzise, and Frédéric Dufaux. Survey on deep learning-based point cloud compression. *Frontiers in Signal Processing*, 2:846972, 2022.

[Schnabel and Klein, 2006] Ruwen Schnabel and Reinhard Klein. Octree-based point-cloud compression. *PBG@ SIGGRAPH*, 3, 2006.

[Song *et al.*, 2023] 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.

[Vinodkumar *et al.*, 2023] Prasoon Kumar Vinodkumar, Dogus Karabulut, Egils Avots, Cagri Ozcinar, and Gholamreza Anbarjafari. A survey on deep learning based segmentation, detection and classification for 3d point clouds. *Entropy*, 25(4):635, 2023.

[Wang *et al.*, 2019] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (tog)*, 38(5):1–12, 2019.

[Wang *et al.*, 2021a] Jianqiang Wang, Dandan Ding, Zhu Li, and Zhan Ma. Multiscale point cloud geometry compression. In *2021 Data Compression Conference (DCC)*, pages 73–82. IEEE, 2021.

[Wang *et al.*, 2021b] Xin Wang, Yudong Chen, and Wenwu Zhu. A survey on curriculum learning. *IEEE Trans-*

actions on Pattern Analysis and Machine Intelligence, 44(9):4555–4576, 2021.

[Wang *et al.*, 2022] 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 Intelligence*, 2022.

[WG 7 and Coding, 2023] MPEG 3D Graphics WG 7 and Haptics Coding. G-pcc 2nd edition codec description. *ISO/IEC JTC 1/SC 29/WG 7*, 2023.

[WG 7, 2020] MPEG 3D Graphics Coding WG 7. V-pcc codec description. *ISO/IEC JTC 1/SC 29/WG 7*, 2020.

[Wiesmann *et al.*, 2021] Louis Wiesmann, Andres Milioto, Xieyuanli Chen, Cyrill Stachniss, and Jens Behley. Deep compression for dense point cloud maps. *IEEE Robotics and Automation Letters*, 6(2):2060–2067, 2021.

[Wu *et al.*, 2022] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. *Advances in Neural Information Processing Systems*, 35:33330–33342, 2022.

[You and Gao, 2021] Kang You and Pan Gao. Patch-based deep autoencoder for point cloud geometry compression. In *ACM Multimedia Asia*, pages 1–7. 2021.

[You *et al.*, 2022] Kang You, Pan Gao, and Qing 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*, pages 1–10, 2022.

[Zhang *et al.*, 2022] Junteng Zhang, Gexin Liu, Dandan Ding, and Zhan Ma. Transformer and upsampling-based point cloud compression. In *Proceedings of the 1st International Workshop on Advances in Point Cloud Compression, Processing and Analysis*, pages 33–39, 2022.

[Zhang *et al.*, 2023a] Junteng Zhang, Tong Chen, Dandan Ding, and Zhan Ma. Yoga: Yet another geometry-based point cloud compressor. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 9070–9081, 2023.

[Zhang *et al.*, 2023b] Renrui Zhang, Liuhui Wang, Yali Wang, Peng Gao, Hongsheng Li, and Jianbo Shi. Parameter is not all you need: Starting from non-parametric networks for 3d point cloud analysis. *arXiv preprint arXiv:2303.08134*, 2023.

[Zhao *et al.*, 2021] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 16259–16268, 2021.