Efficient Hierarchical Entropy Model for Learned Point Cloud Compression

Rui Song[1], Chunyang Fu[1], Shan Liu[2], Ge Li[1]*
School of Electronic and Computer Engineering, Shenzhen Graduate Scool, Peking University[1]
Tencent America[2]
rsong@stu.pku.edu.cn, fuchy@stu.pku.edu.cn, shanl@tencent.com, geli@ece.pku.edu.cn

Abstract

Learning an accurate entropy model is a fundamental way to remove the redundancy in point cloud compression. Recently, the octree-based auto-regressive entropy model which adopts the self-attention mechanism to explore dependencies in a large-scale context is proved to be promising. However, heavy global attention computations and auto-regressive contexts are inefficient for practical applications. To improve the efficiency of the attention model, we propose a hierarchical attention structure that has a linear complexity to the context scale and maintains the global receptive field. Furthermore, we present a grouped context structure to address the serial decoding issue caused by the auto-regression while preserving the compression performance. Experiments demonstrate that the proposed entropy model achieves superior rate-distortion performance and significant decoding latency reduction compared with the state-of-the-art large-scale auto-regressive entropy model.

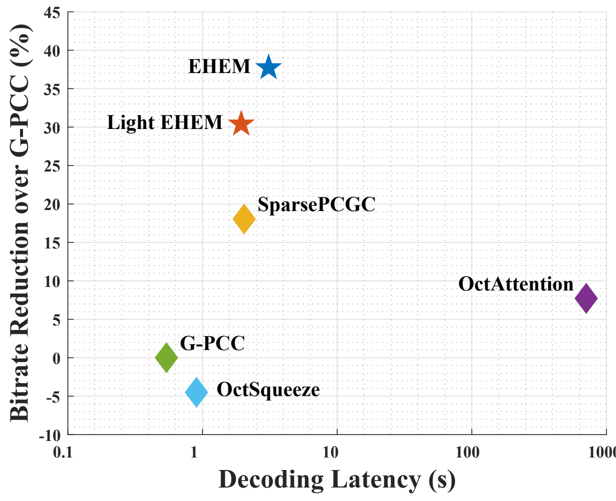

Figure 1. Bitrate and decoding speed in the log scale for lossless compression on 16-bit quantized SemanticKITTI. The proposed method EHEM yields state-of-the-art performance with a comparable decoding latency to the efficient traditional method G-PCC.

1. Introduction

Point cloud is a fundamental data structure to represent 3D scenes. It has been widely applied in 3D vision systems such as autonomous driving and immersive applications. The large-scale point cloud typically contains millions of points [36]. It is challenging to store and transmit such massive data. Hence, efficient point cloud compression that reduces memory footprints and transmission bandwidth is necessary to develop practical point cloud applications.

Recently, deep learning methods have promoted the development of point cloud compression [4, 9, 10, 16, 31, 36, 40, 43]. It is a common pipeline to learn an octree-based entropy model to estimate octree node symbol (*i.e*., occupancy symbol) distributions. The point cloud is first organized as an octree, and occupancy symbols are then encoded into the bitstream losslessly by an entropy coder (*e.g*., arithmetic coder [47]). An accurate entropy model is required since it reduces the cross entropy between the estimated distribution and ground truth, which is corresponding to actual bitrates. Various attempts have been made to improve the accuracy by designing different context structures [4, 10, 16, 37]. The key of these advances is to increase the context capacity and introduce references from high-resolution octree representations. For example, the context in OctAttention [10] includes hundreds of previously decoded siblings (*i.e*., nodes at the same octree level). The large-scale context incorporates more references for the entropy coder, and the high-resolution context preserves detailed features of the point cloud. Both of them contribute to building informative contexts and effective entropy models.

However, large-scale context requires heavy computations to model dependencies among numerous references. The previous work [10] uses the global self-attention mechanism to model long-range dependencies within the context, and its complexity is quadratic to the length of the

*Corresponding author.

large-scale context. Furthermore, considering the efficiency issue, it is infeasible to build a deeper entropy model or extend the context scale to enhance the modeling capability based on global attention. Another concern is the serial decoding process caused by the inclusion of previously decoded siblings. This auto-regressive context structure incurs a practically unacceptable decoding latency (*e.g.*, around 700 seconds per frame).

To address these issues, we build an entropy model with an efficient hierarchical attention model and a parallel-friendly grouped context structure. The hierarchical attention model partitions the context into local windows and computes attention within these windows independently. Therefore, the complexity is linear to the context scale, which allows the further extension of network depth and context capacity to improve performance. Since the receptive field of the localized attention is limited, we adopt a multi-scale network structure to query features across different windows. The context is progressively downsampled by merging neighboring nodes to generate a new token. Then, cross-window dependencies can be captured by incorporating these new tokens in the same window. The grouped context divides the occupancy symbol sequence into two groups. Each group is conditioned on ancestral features and previously decoded groups, and hence nodes in the same group can be coded in parallel. Furthermore, in contrast to the previous auto-regressive context that only exploits causal parts of the ancestral context [10], the grouped context allows to make use of a complete ancestral context.

The proposed efficient hierarchical entropy model called EHEM is evaluated on SemanticKITTI [3] and Ford [32] benchmarks. It surpasses state-of-the-art methods in terms of both compression performance and efficiency. Contributions of this work can be summarized from the following perspectives:

- We propose a hierarchical attention model, which yields improved compression performance by extending model and context while keeping the efficiency.
- We design a grouped context structure that enables parallel decoding. It adapts the entropy model using high-resolution references to practical applications.
- The proposed method achieves state-of-the-art RD performance with a practically applicable coding speed.

2. Related Work

2.1. Learned Point Cloud Compression

Point cloud compression is an active research area in both industry and academia. Traditional point cloud compression codecs are generally interpretable, robust, and efficient [11, 25, 39, 41]. However, neural codecs have shown better rate-distortion performance recently.

Various data structures and models have been developed to boost the rate-distortion performance for learned point cloud compression. The pioneering works [15,17,36,44,48] construct analysis and synthesis transforms to compress point clouds to latent representations. These transforms are learned on point sets with the PointNet backbone [33, 34] or on the voxel structure using 3D convolution. However, distortions are inevitable in these transforms. It is therefore difficult to preserve high-frequency information in latent vectors. Hence, these structures are preferable to lossy compression at low bitrates.

Learned entropy model is another effective pipeline to reduce bitrates without introducing distortion. Since the inference of the entropy model depends on the context, it is important to build an informative context. A typical entropy model predicts the binary occupancy status of the voxel based on the context of several adjacent voxels [30, 31]. Similarly, entropy model can be established on the octree by estimating occupancy symbol distributions. Early tree-structured entropy models exploit references from ancestor nodes [4,16,37]. Later works introduce adjacent voxels that have the same resolution as the currently coded octree node to provide high-resolution references [18]. A recent work replaces neighboring voxels with decoded siblings to extend the context [10], and it yields state-of-the-art performance. However, its efficiency is limited by the expensive global attention computation and auto-regressive processing.

2.2. Learned Image Compression

The typical learned image compression pipeline is based on non-linear transforms and hyperprior-based entropy models [1,2,5,19,21,26,27,35,49]. In addition, decoded latent elements also can be introduced into the context to generate parameters for entropy coding, leading to improved performance with an auto-regressive coding process [28]. To accelerate the coding, a few parallel solutions have been developed [13, 14, 29]. The key of these modifications is performing auto-regression at the level of groups instead of pixels to parallelize coding within each group. For example, the checkerboard context model divides latent elements into two groups in the spatial domain [14]. A checkerboard mask is employed to constrain that the inference of the second group only refers to the decoded first group. Despite the data structures of point clouds and images are different, these strategies are instructive to solve the serial coding problem in point cloud compression.

3. Preliminary

3.1. Octree Structure

The octree [24] provides a progressive representation of the point cloud, where each level represents the point cloud with a certain resolution. An octree recursively divides occupied voxels into 8 equal-sized subvoxels until the required resolution is reached. The occupancy symbol is composed of 8 bits (1 to 255 in decimal), where each bit indicates the occupancy status of the corresponding subvoxel. It is an effective data structure to represent large regions, since neighboring nodes in the breadth-first traversal order might locate at distant locations.

The octree-based point cloud compression transmits an octree to the receiver instead of point coordinates [38, 39]. The octree is represented by a breadth-first traversed occupancy symbol sequence. An entropy coder is adopted to encode these symbols into the bitstream losslessly. At the receiver, an identical octree is reconstructed, which restores the geometry structure of the original point cloud.

3.2. Large-scale Auto-regressive Entropy Model

The entropy coder encodes the occupancy symbol sequence $\mathbf{x} = \{x_1, \dots, x_n\}$ according to the distribution $\tilde{p}(\mathbf{x})$ estimated by a learned entropy model. The joint distribution is factorized as the product of probability of each node:

$$\tilde{p}(\mathbf{x}) = \prod_i \tilde{p}_i(x_i \mid \mathbf{C}_i), \tag{1}$$

where $\mathbf{C}_i$ is the contextual information for predicting x_i. Here, nodes are considered to be conditionally independent. Consumed bitrates are given by the cross entropy between the estimated distribution $\tilde{p}(\mathbf{x})$ and ground truth $p(\mathbf{x})$, formulated as $\mathbb{E}_{\mathbf{x}\sim p}\left[-\log_2 \tilde{p}(\mathbf{x})\right]$. An accurate entropy model that minimizes this entropy is helpful to reduce bitrates.

To formulate the context structure, we characterize each octree node n_i by features $\mathbf{u}_i$ including its occupancy symbol x_i, level index, octant index, and parent bounding box coordinates. The ancestral context collects features from the currently coded node and its K ancestors as:

$$\mathbf{a}_i = \left\{\mathbf{u}_i^{\varnothing}, \mathbf{u}_{\text{anc}(i)}, \dots, \mathbf{u}_{\text{anc}(\dots\text{anc}(i))}\right\}, \tag{2}$$

where $\mathbf{u}_i^{\varnothing}$ excludes x_i from $\mathbf{u}_i$ since it is unknown when decoding x_i. The ancestor-dependent methods [4, 16] predict the occupancy distribution $\tilde{p}_i(x_i)$ based on the ancestral context $\mathbf{a}_i$.

The large-scale auto-regressive entropy model [10] constructs an extensive context to incorporate more information for inference. It combines the occupancy symbol of the previous sibling with the ancestral context as $\mathbf{v}_i = \{x_{i-1}, \mathbf{a}_i\}$. Besides, the context is extended by introducing features from $N-1$ previously decoded sibling nodes and their ancestors to build a context window with length N:

$$\mathbf{C}_i = \{\mathbf{v}_{i-N+1}, \dots, \mathbf{v}_i\}. \tag{3}$$

$\mathbf{C}_i$ is then embedded to features $\mathbf{F}_i = \{\mathbf{f}_{i-N+1}, \dots, \mathbf{f}_i\}$, and a self-attention module [42] is adopted to model dependencies among N embedded features. A mask is employed to constrain that only causal contexts $\mathbf{f}_{<i}$ are visible. Finally, attention-weighted features $\tilde{\mathbf{f}}_i$ are projected to the distribution $\tilde{p}_i(x_i)$, which is used for entropy coding.

Since the octree is coded in a breadth-first manner, ancestral contexts are completely known while coding any node. Therefore, the ancestor-dependent entropy model can decode multiple nodes at the same octree level in parallel. In contrast, the auto-regressive context model is conditioned on previously decoded siblings (*e.g.*, x_{i-1}). It hence requires a serial decoding process. The global attention computation among N elements in $\mathbf{F}_i$ also limits its efficiency. However, large-scale contexts and sibling references are still effective to improve performance. Therefore, we aim to build an efficient entropy model while preserving these advantages.

4. Efficient Hierarchical Entropy Model

4.1. Overall Architecture

The overview of the proposed entropy model is shown in Fig. 2. It is composed of a grouped context structure and a hierarchical attention model. The occupancy sequence $\mathbf{x}$ is divided into a series of non-overlapped subsequences $\mathbf{x}_i = \{x_{i-N+1}, \dots, x_i\}$ with length N. Contextual information for each subsequence is called a context window. The grouped context equally partitions $\mathbf{x}_i$ into two groups $\mathbf{x}_{i_1}$ and $\mathbf{x}_{i_2}$. The first group $\mathbf{x}_{i_1}$ is only conditioned on the ancestral context $\mathbf{A}_i = \{\mathbf{a}_{i-N+1}, \dots, \mathbf{a}_i\}$, where $\mathbf{a}_i$ is defined in Eq. (2). The second group $\mathbf{x}_{i_2}$ is conditioned on both siblings $\mathbf{x}_{i_1}$ and ancestors $\mathbf{A}_i$.

The hierarchical attention model computes dependencies within local windows to improve the efficiency. The self-attention model discovers dependencies among N ancestral features. The ancestral context $\mathbf{A}_i$ is partitioned into $\frac{N}{L}$ non-overlapped local windows with length L. Then we regard parent coordinates in the same window as a point cloud, and feed $\mathbf{A}_i$ to a DGCNN-based [46] feature extractor to generate C-dimensional geometry-aware features $\mathbf{F}_i^a$ from local windows separately. $\mathbf{F}_i^a$ is then passed to successive localized self-attention blocks. Weighted ancestral features of the first group are projected to occupancy distributions $\tilde{p}(\mathbf{x}_{i_1})$ using a multi-layer perceptron (MLP). Then $\mathbf{x}_{i_1}$ is coded and we predict the second group $\mathbf{x}_{i_2}$. $\mathbf{x}_{i_1}$ is embedded to $\mathbf{F}_{i_1}^s$ and concatenated to its ancestral features $\tilde{\mathbf{F}}_{i_1}^a$ as $\mathbf{F}_{i_1}$. Subsequently, we adopt a hierarchical cross-attention model to introduce sibling-involved features $\mathbf{F}_{i_1}$ to ancestral features $\tilde{\mathbf{F}}_{i_2}^a$. Finally, an MLP produces distributions $\tilde{p}(\mathbf{x}_{i_2})$ based on sibling features $\tilde{\mathbf{F}}_{i_2}^s$ and ancestral features $\tilde{\mathbf{F}}_{i_2}^a$.

4.2. Grouped Context

The auto-regressive context model requires serial decoding since it depends on the previous sibling x_{i-1}. To

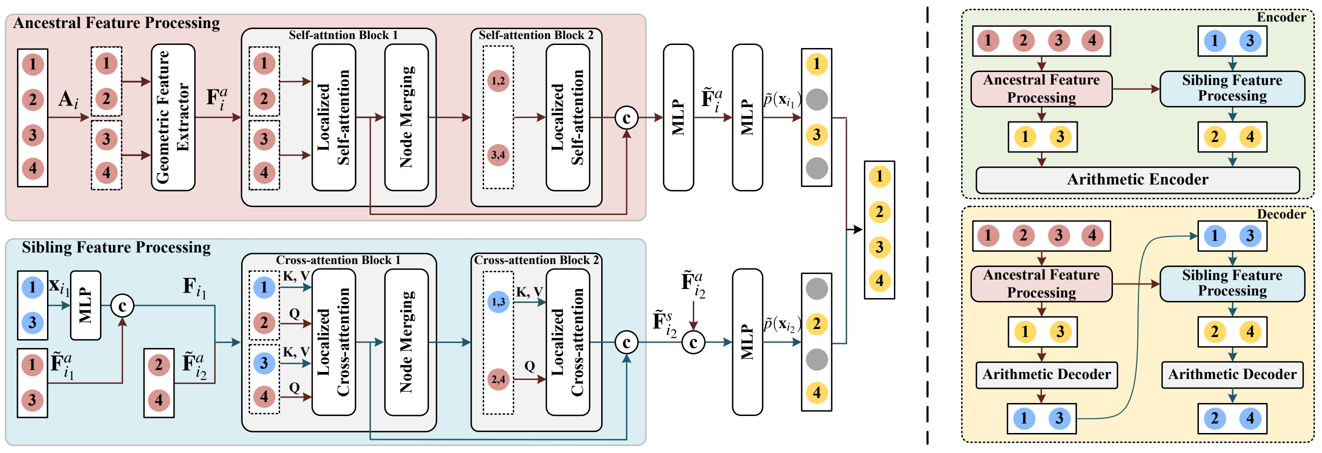

Figure 2. Left: The architecture of EHEM for context window length $N = 4$ and local window length $L = 2$. Red, blue, and yellow nodes denote ancestral features, sibling-involved features, and estimated distributions, respectively. Attention is computed within local windows (dashed boxes). Right: The encoding and decoding process following the grouped context structure. The occupancy sequence $\mathbf{x}_i$ is split into two disjoint groups and coded in turn.

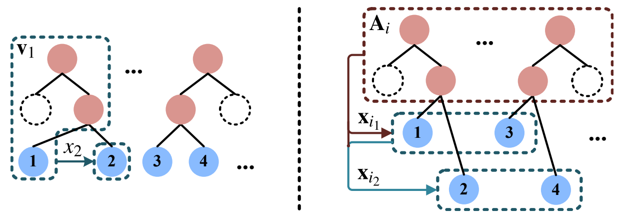

Figure 3. Illustration of auto-regressive (left) and grouped (right) context structures.

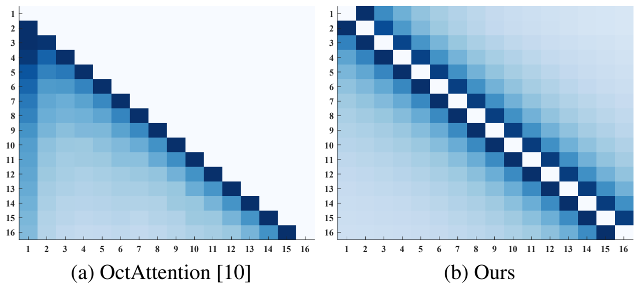

Figure 4. Visualization of dependencies discovered by different methods. Deep colors indicate strong dependencies.

address this issue, we decompose the occupancy subsequence into two groups and remove dependencies within each group. As shown in Fig. 3, $\mathbf{x}_{i_1}$ is first decoded based on the ancestral context $\mathbf{A}_i$, then it is introduced as sibling priors to predict $\mathbf{x}_{i_2}$. Since the required contexts are completely known before decoding each group, nodes in the same group can be decoded in parallel. The auto-regressive entropy model requires N sequential steps to decode the subsequence $\mathbf{x}_i$, while the grouped context only takes 2 steps to decode two groups in turn.

The prediction of $\mathbf{x}_{i_2}$ depends on $\mathbf{x}_{i_1}$, and hence it is important to design an effective decomposition pattern. We visualize ancestral dependencies among the first 16 siblings in the context window in Fig. 4b. Dependencies are represented by normalized attention scores computed on $\mathbf{F}_i^a$, and results are averaged over all self-attention layers from around 200 SemanticKITTI scans. Fig. 4b reveals that closer neighbors have stronger dependencies, and dependency activations at more distant locations are weak and uniform. This distribution is consistent with both intuitions and experiences in learned image compression [14]. We assume that dependencies computed on ancestral and sibling-involved contexts have similar distributions. With this property, we use a uniform decomposition pattern expressed as:

$$\begin{cases} \mathbf{x}_{i_1} = \{x_1, \ldots, x_{2j-1}, \ldots, x_{i-1}\}, \mathbf{C}_{i_1} = \{\mathbf{A}_i\}, \\ \mathbf{x}_{i_2} = \{x_2, \ldots, x_{2j}, \ldots, x_i\}, \ \mathbf{C}_{i_2} = \{\mathbf{A}_i, \mathbf{x}_{i_1}\}. \end{cases} \tag{4}$$

It preserves two most important neighbors x_{2j-1} and x_{2j+1} for x_{2j} and uniformly samples $\mathbf{x}_{i_1}$ at more distant locations.

In addition, the grouped context separates ancestral and sibling priors to utilize a complete ancestral context. For the auto-regressive context in OctAttention, ancestral and sibling features are concatenated when computing dependencies (see Eq. (3)). To keep the causality, both ancestral and sibling features from non-causal positions (*i.e.*, $\mathbf{f}_{>i}$) are discarded, as shown in Fig. 4a. However, these ancestral features are actually available. As illustrated in Fig. 4b, $\mathbf{f}^a_{>i}$ have similar dependency activations to $\mathbf{f}^a_{<i}$. Therefore, they are equivalently important to compute dependencies. The grouped context structure accesses to more ancestral features, which leads to better compression performance.

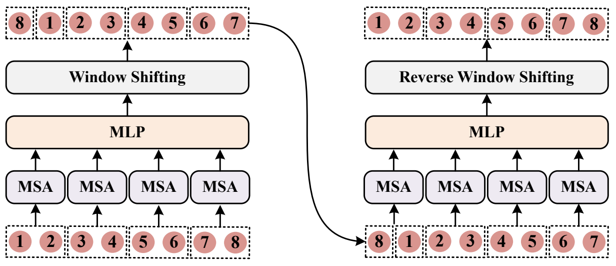

Figure 5. Illustration of two successive localized attention layers for the case of $N = 8$ and $L = 2$. MSA represents the multi-head self-attention.

From the perspective of context modeling, this grouped context can be interpreted as the space-channel context in image compression [13]. As shown in Eq. (3), N nodes constitute the spatial domain, and each spatial position has $K + 1$ channels, consisting of K ancestors and 1 sibling. The grouped context partitions on both channel and spatial dimensions. Available contexts for each element are decoded channels (ancestors) from all spatial positions, and elements in the same channel (siblings) at decoded spatial positions. Therefore, it is feasible to use similar context structures for point cloud and image compression, and both domains might benefit from advances of each other.

4.3. Hierarchical Attention

To reduce the complexity, we discover dependencies within local windows and model long-range dependencies with a multi-scale network structure. It is inspired by vision transformers designed for 2D images [8, 22, 23, 45]. Since the localized attention can not exploit long-range dependencies in the large-scale context, we gradually downsample the context to represent numerous nodes with L tokens. The downsampling is implemented by a node merging operation. Two neighboring nodes are merged by concatenating their features on the channel dimension. Then, $2C$-dimensional concatenated features are passed to an MLP to shrink the channel dimension to C. In this way, features from distant nodes can be computed in the same window and the receptive field is increased. Furthermore, dependencies are estimated at different scales, and closer neighbors are computed with fine-grained representations. It is reasonable since closer neighbors are more informative. In contrast, distant nodes can be coarsely characterized. We further introduce the shifted window strategy [23] to capture dependencies across neighboring local windows. Dependencies are computed with a window partition alternating between two patterns, as shown in Fig. 5.

A self-attention block is composed of various localized attention layers and one node merging module (except for the last block). The hierarchical attention model is a stack of $\log_2 \frac{N}{L} + 1$ blocks to capture all N nodes in the last block. The number of blocks can be further reduced by aggregating more than two nodes in a single node merging module. We produce a hierarchical representation $\tilde{\mathbf{F}}_i^a$ by aggregating features and dependencies from different scales. Therefore, we refer to this model as a hierarchical attention model.

By performing attention operations within local windows, the hierarchical attention model reduces the complexity from quadratic to linear scale. Complexities of attention computations in global and localized self-attention are:

$$\Omega(\text{Global}) = 2N^2C, \quad \Omega(\text{Localized}) = 2LNC. \tag{5}$$

It allows us to build a much deeper entropy model with reasonable computational costs to enhance the modeling capability, or substantially expand the context scale to introduce more potentially useful references.

The hierarchical cross-attention follows a similar localized attention practice. Since the decomposition pattern is uniform, each local window includes $\frac{L}{2}$ nodes from $\mathbf{x}_{i_1}$ and another $\frac{L}{2}$ nodes from $\mathbf{x}_{i_2}$. They are regarded as key and query features, respectively. We compute dependencies among keys and queries within each local window, and weight keys according to their dependencies to queries. To extend the receptive field, neighboring keys (and queries) are progressively merged into new key (and query) tokens. Finally, features from different scales are concatenated, which are then exploited to predict x_{i_2} with ancestral features $\tilde{\mathbf{F}}_{i_2}^a$ jointly.

4.4. Learning

The optimization objective of the octree-based entropy model is to minimize the total bitrates for communicating octree nodes. The bitrate is given by the cross entropy between the estimated distribution $\tilde{p}(\mathbf{x})$ and the ground truth $p(\mathbf{x})$ at each node:

$$\ell = -\sum_i \log \tilde{p}_i(x_i \mid \mathbf{C}_i), \tag{6}$$

where $\tilde{p}_i(x_i \mid \mathbf{C}_i)$ is the distribution estimated by the proposed entropy model based on the context $\mathbf{C}_i$.

5. Experiments

5.1. Experimental Settings

Datasets Experiments are conducted on two large-scale point cloud datasets SemanticKITTI [3] and Ford [32]. SemanticKITTI is composed of 43552 LiDAR scans acquired from autonomous driving scenes, which are divided into 22 sequences. We follow the default split to perform training on sequences 00 to 10 and evaluation on sequences 11 to 21. Ford is another LiDAR point cloud dataset used in MPEG point cloud compression standardization. It includes 3 sequences, each consists of 1500 scans. On the Ford dataset,

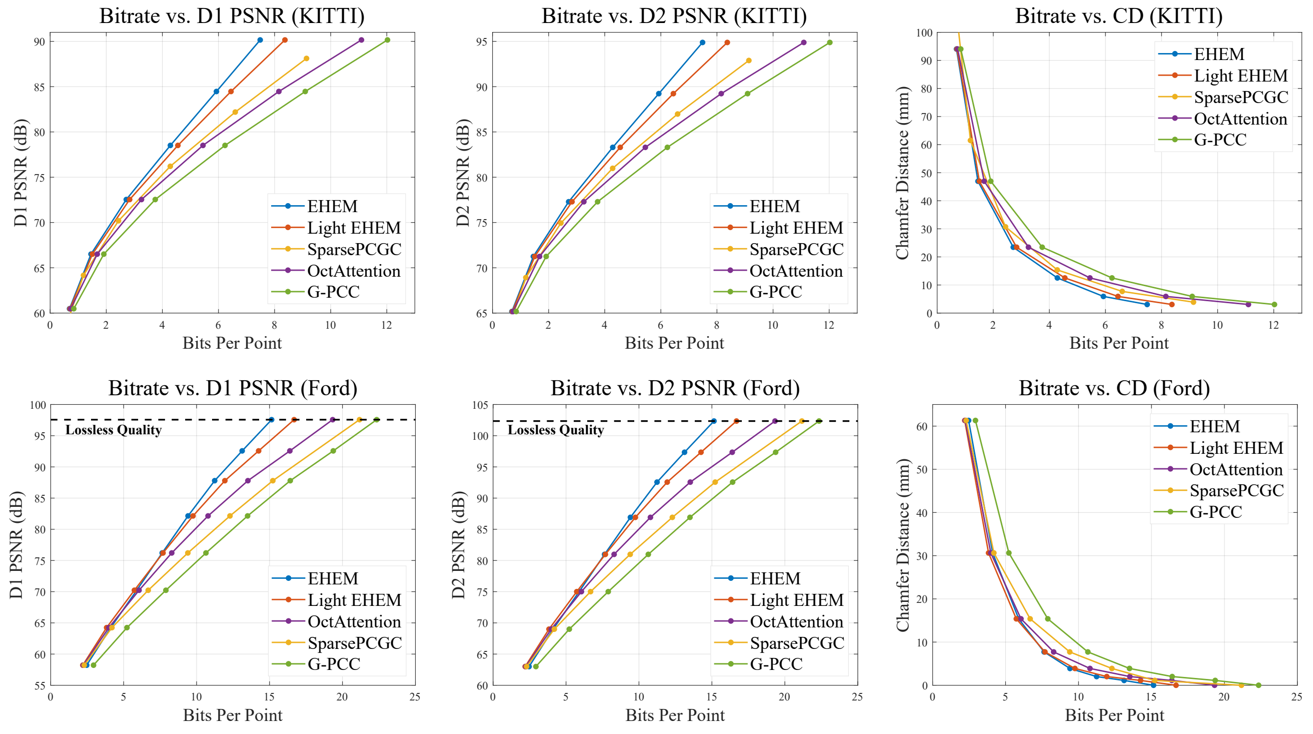

Figure 6. Rate-distortion performance of different methods on SemanticKITTI (top) and Ford (bottom).

we follow the suggested dataset partition in MPEG standardization [7], where sequence 01 is used for training and sequences 02 and 03 are used for evaluation.

Baselines To verify the effectiveness of the proposed model, we compare it with the state-of-the-art octree-based learned entropy model OctAttention [10] and the competitive voxel-based learned method SparsePCGC [43]. We also provide comparisons with the representative handcrafted compression method MPEG G-PCC [12].

Implementation settings Quantization is necessary to construct octrees from point clouds. For SemanticKITTI, we follow the quantization settings in OctSqueeze [16], where the quantization step is set to $\frac{400}{2^D-1}$ to build an octree with the depth of D. Octrees are truncated with a maximum depth of 16 for training and evaluation. We set the quantization step to 2^{18-D} and construct octrees with the maximum depth of 18 on Ford. Since the original Ford dataset has been quantized with 18-bit precision, compression is lossless at the highest octree level.

The hierarchical self-attention model has 5 blocks including $4, 4, 4, 4, 2$ layers, respectively. The hierarchical cross-attention model consists of 4 blocks with $2, 2, 1, 1$ layers. The head number is set to 4 for all attention layers. The channel dimension C is set to 256, and the ancestor depth K is set to 3. The context window length N is set to 8192, which is initially divided into 16 local windows with the length L of 512. To further achieve speed-accuracy trade-offs, we present a lightweight EHEM model called Light EHEM. It has the same network structure as EHEM except that its 5 self-attention blocks have $2, 2, 2, 2, 2$ layers and its channel dimension C is set to 192.

Table 1. Inference times (in seconds) for encoding/decoding a D-depth octree on SemanticKITTI. Runtimes for G-PCC are total times.

Method	D=12	D=14	D=16
G-PCC	0.25 / 0.12	0.66 / 0.32	1.07 / 0.54
SparsePCGC	1.14 / 0.86	1.76 / 1.43	2.43 / 2.04
OctAttention	0.08 / 83	0.31 / 321	0.66 / 708
EHEM	0.40 / 0.43	1.21 / 1.39	2.53 / 3.01
Light EHEM	0.29 / 0.33	0.79 / 0.92	1.63 / 1.94

We adopt an Adam optimizer [20] with a learning rate of 10^{-4} to train two models for 10 and 50 epochs on SemanticKITTI and Ford datasets, respectively. The evaluation is implemented on the NVIDIA V100 GPU.

Metrics We adopt the point-to-point PSNR (D1 PSNR), point-to-plane PSNR (D2 PSNR) [6], and Chamfer distance (CD) to measure the distortion. Experiments on SemanticKITTI follow the PSNR calculations in OctSqueeze and MuSCLE [4, 16], where the peak value is set to 59.70. On the Ford dataset, we set the peak value to 30000 [6]. Consumed bitrates are measured by bits per point (bpp).

Table 2. Complexity comparison on encoding/decoding time, theoretical computation cost (FLOPs), parameters, and memory usage among OctAttention, EHEM, and Light EHEM.

Method	Enc / Dec	FLOPs	#param.	Mem.
OctAttn	0.66s / 708s	124.3G	4.23M	1.3G
EHEM	2.53s / 3.01s	184.4G	13.01M	2.9G
L-EHEM	1.63s / 1.94s	102.9G	6.34M	2.6G

Table 3. Comparison of FLOPs among OctAttention, EHEM, and Light EHEM with respect to the context length N.

Method	512	1024	2048	4096	8192
OctAttn	6.0G	15.5G	45.0G	145.7G	514.0G
EHEM	9.0G	18.4G	38.1G	81.5G	184.4G
L-EHEM	5.2G	10.7G	21.9G	46.4G	102.9G

5.2. Performance Evaluation

The quantitative results of the rate-distortion performance are shown in Fig. 6. The proposed method EHEM achieves significant improvements over other baselines. For example, on SemanticKITTI, EHEM achieves an average gain of 28.89% to G-PCC, and 19.47% bitrate reduction on average compared with OctAttention. Furthermore, Light EHEM also preserves satisfactory performance and outperforms other baselines.

To evaluate the efficiency of EHEM, we report inference times at different quantization steps in Tab. 1. EHEM achieves notable decoding time reduction over the autoregressive context model OctAttention. The encoding time increases compared with OctAttention since the grouped context requires a two-step inference (for $\mathbf{x}_{i_1}$ and $\mathbf{x}_{i_2}$), while OctAttention only requires one step. However, this encoding time is still practically acceptable and the grouped context is effective to speed up decoding. We further represent both rate-distortion performance and decoding time in Fig. 1. It is shown that EHEM is effective regarding both compression performance and decoding speed.

We provide the complexity analysis of EHEM and OctAttention in Tab. 2 and Tab. 3. Here, we report required FLOPs to infer a single context window. Note that larger context predicts more nodes with one forward pass. For example, EHEM requires 81.5×2 and 184.4×1 GFLOPs to predict 8192 nodes when N is set to 4096 and 8192, respectively. It is shown that EHEM has comparable FLOPs to OctAttention when the context capacity is 1024, which is the default setting for OctAttention. However, EHEM contains 24 hierarchical attention layers while OctAttention only includes 2 global attention layers. It proves the superiority of the localized attention structure in terms of extending the network depth. Furthermore, localized attention is also effective to enlarge the context due to the linear complexity. FLOPs of EHEM increase 2.26 times as the context is enlarged from 4096 to 8192. In contrast, the complexity of OctAttention increases 3.53 times with the same extension. Besides, Light EHEM achieves lower FLOPs at all context window length settings compared to OctAttention. EHEM has slightly larger model size and memory usage compared with OctAttention. These increases are due to the extension of model capacity and context scale, but it is still reasonable for practical use.

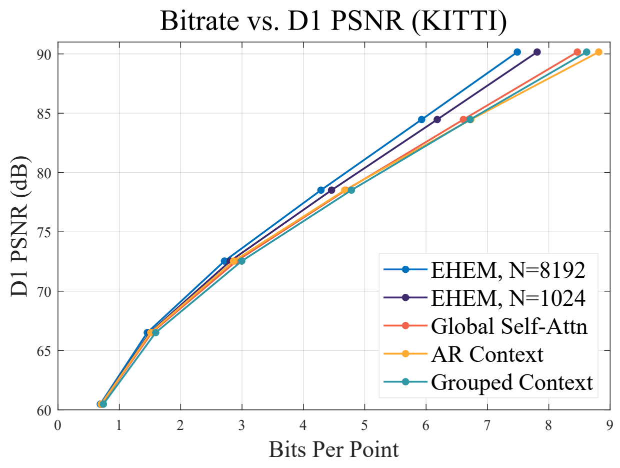

Figure 7. Ablation studies on attention models, context scales, and context structures.

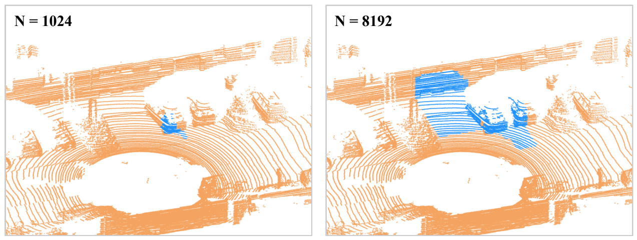

Figure 8. Contexts with different capacities N on 16-bit quantized SemanticKITTI. Points in the context are colored in blue.

5.3. Ablation Studies and Analysis

Hierarchical attention structure To verify the effectiveness of the hierarchical attention structure, we replace the hierarchical self-attention model with global self-attention layers to build a counterpart with similar FLOPs. Since it is impractical to compute global self-attention with $N = 8192$, we conduct experiments with $N = 1024$. Here, EHEM still has 18 localized self-attention layers, and the global-attention counterpart contains 5 layers. The comparison between EHEM ($N = 1024$) and Global Self-Attn in Fig. 7 proves that the hierarchical attention model with a much deeper network achieves considerable improvements.

Network depth With the same network structure, the comparison between EHEM and Light EHEM shows that the

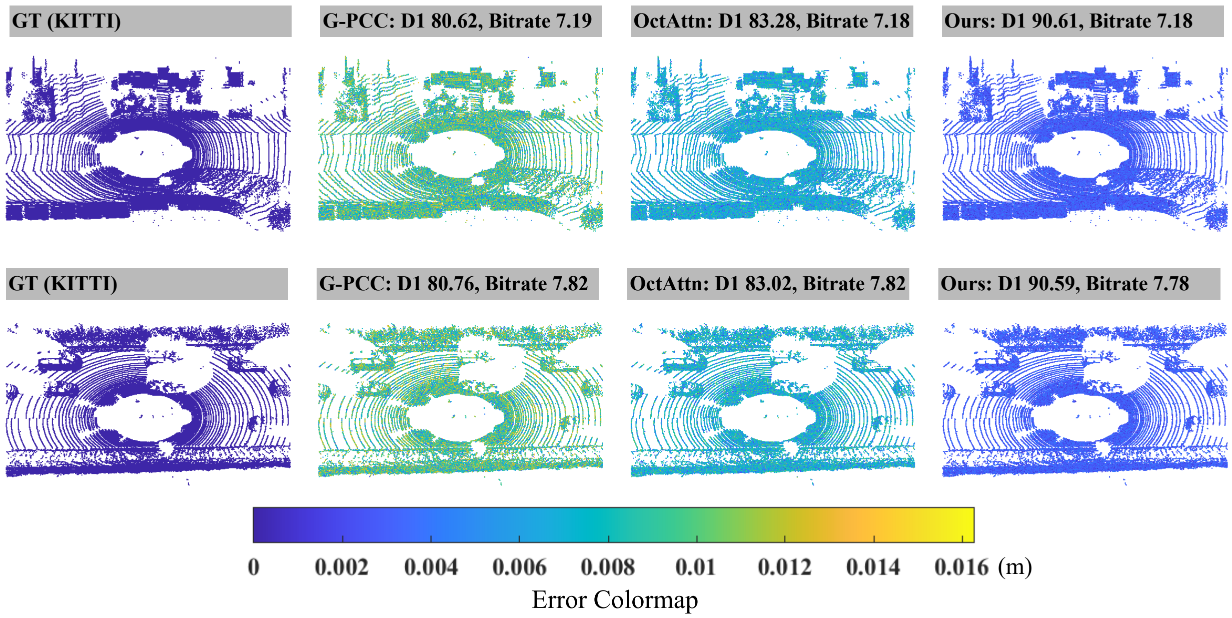
GT (KITTI)
G-PCC: D1 80.62, Bitrate 7.19
OctAttn: D1 83.28, Bitrate 7.18
Ours: D1 90.61, Bitrate 7.18
GT (KITTI)
G-PCC: D1 80.76, Bitrate 7.82
OctAttn: D1 83.02, Bitrate 7.82
Ours: D1 90.59, Bitrate 7.78
0
0.002
0.004
0.006
0.008
0.01
0.012
0.014
0.016 (m)
Error Colormap

Figure 9. Visualized compression results of G-PCC, OctAttention, and our method on the SemanticKITTI dataset.

deeper network leads to better performance. Therefore, it is helpful to enhance the expressiveness of the model by stacking more layers within a reasonable range.

Context scale The linear complexity of the hierarchical attention also allows the further extension of the context scale N. Contexts with different capacities are visualized in Fig. 8. It is shown that captured regions are significantly enlarged when N increases from 1024 to 8192. The extended context includes the complete geometry pattern of the vehicle and introduces references from the neighboring similar vehicle. This is beneficial to improve the rate-distortion performance, as shown in Fig. 7.

Grouped context structure The grouped context structure allows a two-step parallel decoding procedure. Here, we examine its influence on rate-distortion performance. To this end, we train a baseline with an auto-regressive context used in OctAttention. The hierarchical attention is unavailable to this context structure since the node merging operation uses non-causal ancestral references. For a fair comparison, we use global attention to compute dependencies for both grouped and auto-regressive context structures. The grouped context introduces features from the first group via global cross-attention layers. The comparison between AR Context and Grouped Context in Fig. 7 demonstrates that although only half of the nodes access to siblings, the grouped context structure achieves comparable performance. It is because the grouped context preserves ancestral features from non-causal references, as shown in Fig. 4b. Besides, it proves that the grouped context structure does not significantly improve compression performance. Therefore, improvements of EHEM are mainly due to stronger hierarchical attention blocks.

5.4. Qualitative Results

In Fig. 9, we show the visualized compression distortions of different methods at similar bitrates. It indicates that our method provides better reconstructions compared with other baselines. For example, EHEM outperforms G-PCC by roughly 10 dB D1 PSNR improvements.

6. Conclusion

We proposed an efficient learned entropy model for point cloud compression. It adopts a hierarchical attention structure, which allows us to substantially extend the network depth and context capacity to improve the rate-distortion performance with reasonable complexity. We further developed a grouped context structure to realize a parallel decoding procedure. Moreover, this context aligns the context modeling regimes of point cloud and image compression, and we hope it would be helpful to bridge the gap between two communities. The proposed model is proved to yield superior compression performance and significantly reduce the decoding time compared to the state-of-the-art auto-regressive method.

Acknowledgement

This work was supported by National Natural Science Foundation of China (No. 62172021).

References

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

[2] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *International Conference on Learning Representations*, 2018.

[3] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9297–9307, 2019.

[4] Sourav Biswas, Jerry Liu, Kelvin Wong, Shenlong Wang, and Raquel Urtasun. Muscle: Multi sweep compression of lidar using deep entropy models. In *Advances in Neural Information Processing Systems*, pages 22170–22181, 2020.

[5] Zhengxue Cheng, Heming Sun, Masaru Takeuchi, and Jiro Katto. Learned image compression with discretized gaussian mixture likelihoods and attention modules. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7939–7948, 2020.

[6] MPEG 3D Graphics Coding. Common test conditions for g-pcc. ISO/IEC JTC1/SC29/WG7 N00106, 2021.

[7] MPEG 3D Graphics Coding. Preliminary dataset for ai-based point cloud experiments. ISO/IEC JTC1/SC29/WG7 W21570, 2022.

[8] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6824–6835, 2021.

[9] Guangchi Fang, Qingyong Hu, Hanyun Wang, Yiling Xu, and Yulan Guo. 3dac: Learning attribute compression for point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14819–14828, 2022.

[10] 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*, 2022.

[11] Diogo C Garcia, Tiago A Fonseca, Renan U Ferreira, and Ricardo L de Queiroz. Geometry coding for dynamic voxelized point clouds using octrees and multiple contexts. *IEEE Transactions on Image Processing*, 29:313–322, 2019.

[12] MPEG Group. Mpeg g-pcc tmc13. `https://github.com/MPEGGroup/mpeg-pcc-tmc13`, 2021.

[13] Dailan He, Ziming Yang, Weikun Peng, Rui Ma, Hongwei Qin, and Yan Wang. Elic: Efficient learned image compression with unevenly grouped space-channel contextual adaptive coding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5718–5727, 2022.

[14] Dailan He, Yaoyan Zheng, Baocheng Sun, Yan Wang, and Hongwei Qin. Checkerboard context model for efficient learned image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14771–14780, 2021.

[15] 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.

[16] Lila Huang, Shenlong Wang, Kelvin Wong, Jerry Liu, and Raquel Urtasun. Octsqueeze: Octree-structured entropy model for lidar compression. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, pages 1313–1323, 2020.

[17] Tianxin Huang and Yong Liu. 3d point cloud geometry compression on deep learning. In *Proceedings of the 27th ACM international conference on multimedia*, pages 890–898, 2019.

[18] Emre Can Kaya and Ioan Tabus. Neural network modeling of probabilities for coding the octree representation of point clouds. In *2021 IEEE 23rd International Workshop on Multimedia Signal Processing*, 2021.

[19] Jun–Hyuk Kim, Byeongho Heo, and Jong–Seok Lee. Joint global and local hierarchical priors for learned image compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5982–5991, 2022.

[20] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[21] Jooyoung Lee, Seunghyun Cho, and Seung-Kwon Beack. Context-adaptive entropy model for end-to-end optimized image compression. In *International Conference on Learning Representations*, 2019.

[22] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12009–12019, 2022.

[23] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[24] Donald Meagher. Geometric modeling using octree encoding. *Computer graphics and image processing*, 19(2):129–147, 1982.

[25] Rufael Mekuria, Kees Blom, and Pablo Cesar. Design, implementation, and evaluation of a point cloud codec for tele-immersive video. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(4):828–842, 2016.

[26] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Conditional probability models for deep image compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4394–4402, 2018.

[27] Fabian Mentzer, George D Toderici, Michael Tschannen, and Eirikur Agustsson. High-fidelity generative image compression. In *Advances in Neural Information Processing Systems*, pages 11913–11924, 2020.

[28] David Minnen, Johannes Ballé, and George D Toderici. Joint autoregressive and hierarchical priors for learned image compression. *Advances in neural information processing systems*, 31, 2018.

[29] David Minnen and Saurabh Singh. Channel-wise autoregressive entropy models for learned image compression. In *2020 IEEE International Conference on Image Processing*, pages 3339–3343, 2020.

[30] Dat Thanh Nguyen and Andre Kaup. Learning-based lossless point cloud geometry coding using sparse representations. *arXiv preprint arXiv:2204.05043*, 2022.

[31] Dat Thanh Nguyen, Maurice Quach, Giuseppe Valenzise, and Pierre Duhamel. Lossless coding of point cloud geometry using a deep generative model. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(12):4617–4629, 2021.

[32] Gaurav Pandey, James R McBride, and Ryan M Eustice. Ford campus vision and lidar data set. *The International Journal of Robotics Research*, 30(13):1543–1552, 2011.

[33] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[34] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, 2017.

[35] Yichen Qian, Ming Lin, Xiuyu Sun, Zhiyu Tan, and Rong Jin. Entroformer: A transformer-based entropy model for learned image compression. In *International Conference on Learning Representations*, 2022.

[36] Maurice Quach, Giuseppe Valenzise, and Frederic Dufaux. Learning convolutional transforms for lossy point cloud geometry compression. In *IEEE International Conference on Image Processing*, pages 4320–4324, 2019.

[37] Zizheng Que, Guo Lu, and Dong Xu. Voxelcontext-net: An octree based framework for point cloud compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6042–6051, 2021.

[38] Ruwen Schnabel and Reinhard Klein. Octree-based point-cloud compression. In *Proceedings of the 3rd Eurographics / IEEE VGTC conference on Point-Based Graphics*, pages 111–120, 2006.

[39] Sebastian Schwarz, Marius Preda, Vittorio Baroncini, Madhukar Budagavi, Pablo Cesar, Philip A Chou, Robert A Cohen, Maja Krivokuća, Sébastien Lasserre, Zhu Li, et al. Emerging mpeg standards for point cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):133–148, 2018.

[40] Xihua Sheng, Li Li, Dong Liu, Zhiwei Xiong, Zhu Li, and Feng Wu. Deep-pcac: An end-to-end deep lossy compression framework for point cloud attributes. *IEEE Transactions on Multimedia*, 24:2617–2632, 2021.

[41] Fei Song, Yiting Shao, Wei Gao, Haiqiang Wang, and Thomas Li. Layer-wise geometry aggregation framework for lossless lidar point cloud compression. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(12):4603–4616, 2021.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, page 6000–6010, 2017.

[43] 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.

[44] Jianqiang Wang, Hao Zhu, Haojie Liu, and Zhan Ma. Lossy point cloud geometry compression via end-to-end learning. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(12):4909–4923, 2021.

[45] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 568–578, 2021.

[46] 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*, 38(5):1–12, 2019.

[47] Ian H Witten, Radford M Neal, and John G Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

[48] Wei Yan, Shan Liu, Thomas H Li, Zhu Li, Ge Li, et al. Deep autoencoder-based lossy geometry compression for point clouds. *arXiv preprint arXiv:1905.03691*, 2019.

[49] Yinhao Zhu, Yang Yang, and Taco Cohen. Transformer-based transform coding. In *International Conference on Learning Representations*, 2022.