OctAttention: Octree-based Large-scale Context Model for Point Cloud Compression

Chunyang Fu[1,2], Ge Li[1], Rui Song[1], Wei Gao[1,2*], Shan Liu[3]

[1]School of Electronic and Computer Engineering, Peking University Shenzhen Graduate School
[2]Peng Cheng Laboratory, [3]Tencent America
fuchy@stu.pku.edu.cn, geli@ece.pku.edu.cn, rsong@stu.pku.edu.cn, gaowei262@pku.edu.cn, shanl@tencent.com

Abstract

In point cloud compression, sufficient contexts are significant for modeling the point cloud distribution. However, the contexts gathered by the previous voxel-based methods decrease when handling sparse point clouds. To address this problem, we propose a multiple-contexts deep learning framework called *OctAttention* employing the octree structure, a memory-efficient representation for point clouds. Our approach encodes octree symbol sequences in a lossless way by gathering the information of sibling and ancestor nodes. Expressly, we first represent point clouds with octree to reduce spatial redundancy, which is robust for point clouds with different resolutions. We then design a conditional entropy model with a large receptive field that models the sibling and ancestor contexts to exploit the strong dependency among the neighboring nodes and employ an attention mechanism to emphasize the correlated nodes in the context. Furthermore, we introduce a *mask* operation during training and testing to make a trade-off between encoding time and performance. Compared to the previous state-of-the-art works, our approach obtains a 10%-35% BD -Rate gain on the Lidar benchmark (*e.g.* SemanticKITTI) and object point cloud dataset (*e.g.* MPEG 8i, MVUB), and saves 95% coding time compared to the voxel-based baseline. The code is available at https://github.com/zb12138/OctAttention.

Introduction

The point cloud is an essential data structure for 3D representation. It has been used in many fields such as virtual reality, smart city, robotics, and autonomous driving (Schwarz et al. 2018). Since massive point clouds are generated, efficient compression techniques are necessary for point cloud storage and transmission. However, point clouds are unordered and have various distributions; it is relatively difficult to compress point clouds compared with 2D images. Fortunately, several schemes for point cloud geometry and attribute compression such as voxel-based, image-based and tree-based algorithms have been proposed and applied in research works (Chou, Koroteev, and Krivokuća 2019; Shao et al. 2017, 2018) and standard specification (3DG 2021).

The MPEG point cloud geometry compression standard (G-PCC) (Schwarz et al. 2018) adopts a hand-crafted

context-adaptive arithmetic encoder for bit allocation, which can be seen as a prediction for the currently encoding node based on the coded information. Recently, entropy encoders based on deep learning have been shown to outperform hand-crafted ones on rate-distortion performance. Among them, some methods partition point clouds into voxels, then adopt 3D convolution to learn and predict the occupancy of each voxel (Wang et al. 2021; Nguyen et al. 2021a; Quach, Valenzise, and Dufaux 2019; Que, Lu, and Xu 2021). Voxel-based models are capable of exploiting the local geometric patterns (*e.g.*, planes, surfaces). However, they are not robust to point cloud resolution. These methods have to divide blocks into different scales for point clouds with varying point densities to find the optimal voxel size. Meanwhile, receptive fields are limited by the computational cost, *i.e.*, they only extract features from voxels within a narrow range. Other works encode the point cloud into an octree, then encode the occupancy of octree nodes based on their ancestor nodes (Huang et al. 2020; Biswas et al. 2020). The octree-based model is robust to resolution, and it also utilizes a broader range of contexts than voxel-based ones. However, prior methods ignore that sibling nodes (*i.e.*, nodes in the same octree level) provide low-level local geometry features, which are significant to exploit the geometry redundancy. In general, prior voxel-based and octree-based methods do not fully use much spatial context information.

In this work, we propose a point cloud compression method called *OctAttention* which generates and utilizes a large-scale context. Voxel is inefficient for representing sparse point clouds; thus, we encode the point cloud into an octree. Subsequently, we improve the predictability of the occupancy of each octree node based on large-scale context, which contains features from ancestor nodes of the current node, sibling nodes, and ancestors of sibling nodes. However, it should be noted that the side effect of expanding context is to introduce redundant and irrelevant information. For example, different sibling nodes may have the same ancestor, but they are repeated multiple times in the context; sibling nodes far from the current node may be worthless for prediction. To tackle this problem, we employ tree-structured attention to weight and explicitly express the contributions of different nodes in the prediction. The siblings in the context disable the parallelization strategy in prior works (Huang et al. 2020; Que, Lu, and Xu 2021), hence we pro-
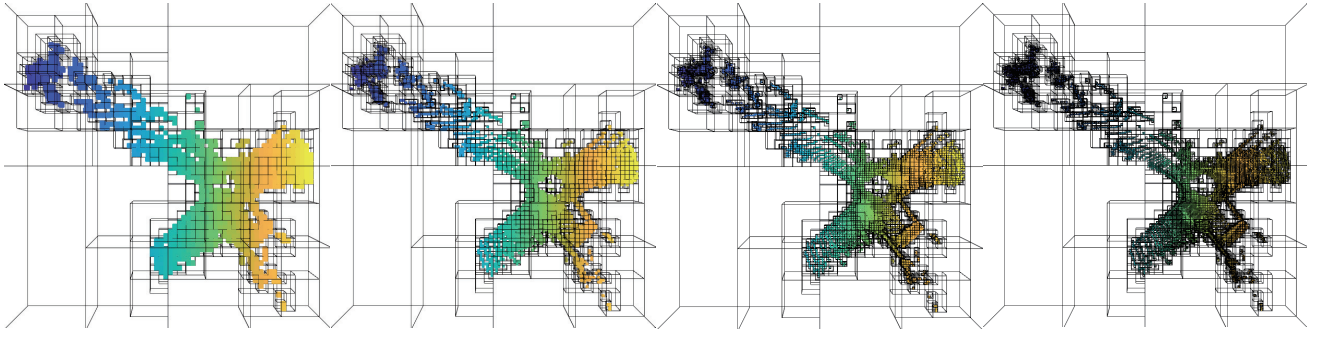

Figure 1: LiDAR Point Cloud quantized by octree with the max depth of 6, 7, 8, 9. Points in the octree leaf nodes decrease with increased depth. The receptive field of voxel-based approach is limited.

pose a *mask* operation to encode multiple nodes in parallel.

We compare the proposed model with state-of-the-art methods on 3D LiDAR dataset SemanticKITTI (Behley et al. 2019), object point cloud dataset MVUB (Charles et al. 2016) and MPEG 8i (Eugene et al. 2017). The experiments show that our method outperforms these state-of-the-art methods, which are only designed for a specific category of point clouds.

The contributions of our work can be summarized as:

- We propose a tree-structured attention mechanism to model the dependency of nodes in a large-scale context, which is achieved by extending the receptive field of context and exploiting features from sibling nodes and their ancestors.

- We employ a *mask* operation to encode octree in parallel to alleviate the drawbacks of introducing siblings in the large-scale context.

- Our generic model of point cloud geometry compression for both LiDAR and object point clouds achieves state-of-the-art performance on several large-scale datasets.

Related Work

Voxel-based Point Cloud Compression

Voxel-based methods quantize the point cloud and classify the voxel occupancy by neural networks. Voxel-based methods outperform G-PCC (3DG 2021) on lossless geometric compression (Nguyen et al. 2021a; Quach, Valenzise, and Dufaux 2019), lossy geometric compression (Quach, Valenzise, and Dufaux 2019, 2020; Wang et al. 2021) and progressive compression (Guarda, Rodrigues, and Pereira 2020). Compared to an octree, geometric patterns can be naturally preserved in the voxel representation. Yet, the side effect is Voxel-based network is sensitive to the density variation and may fail for the sparse point clouds. All of the above methods are applied to dense point clouds (*e.g.* MPEG 8i) and may suffer tremendous computing and memory costs on sparse LiDAR point clouds. The proposed method directly processes the octree occupancy code to overcome the density variation problem.

Tree-based Point Cloud Compression

Tree structures effectively reduce geometric redundancy by merging the common coordinates of point clouds. Numerous algorithms (Lasserre, Flynn, and Qu 2019; Zhang, Gao, and Liu 2020; Schwarz et al. 2018; Kammerl et al. 2012; Kathariya et al. 2018b; Gao et al. 2019) compress point cloud based on tree structures such as octree (Schnabel and Klein 2006), quadtree (Kathariya et al. 2018a), KD tree(Devillers and Gandoin 2000), prediction tree (Gumhold et al. 2005) etc. Recently, many works have focused on designing octree context for arithmetic coding to compress bitstream. (Song et al. 2021) aggregates voxels in reduced space by removing free regions and acquires a compact context model for LiDAR compression. (Garcia and de Queiroz 2018) reorder the node sequences to improve the intra-frames lossless geometric compression.

All of the above methods model the context by hand-crafted features. (Lei, Akhtar, and Mian 2019; Riegler, Osman Ulusoy, and Geiger 2017; Wang et al. 2017) mainly introduce new convolution methods under the octree framework. OctSqueeze (Huang et al. 2020) is the first octree-based deep learning entropy model by modeling the dependency among node and its multiple ancestor nodes. MuSCLE (Biswas et al. 2020) reduces the temporal redundancy by exploiting spatio-temporal relationships across LiDAR sweeps. Both methods avoid high computational complexity, yet the strong dependency among sibling nodes is ignored. VoxelContext-Net (Que, Lu, and Xu 2021) partly solves this problem by employing voxel and octree hybrid structure to learn the context in the previous octree depth. However, features from higher resolution (*i.e.*, from sibling nodes) are still ignored. Besides, as shown in Fig. 1, given a fixed-size voxel-based context, its receptive field shrinks with the increased octree depth. Due to the computational overhead, the receptive field of the voxel-based approach is limited, which restricts the ability to model the context. Our proposed method can acquire more than $500 \times 500 \times 500$ voxels receptive field ($9 \times 9 \times 9$ in VoxelContext-Net). Meanwhile, we introduce sibling nodes and their ancestors in the context. The extended context contains more potentially helpful information to model the distribution of octree nodes.
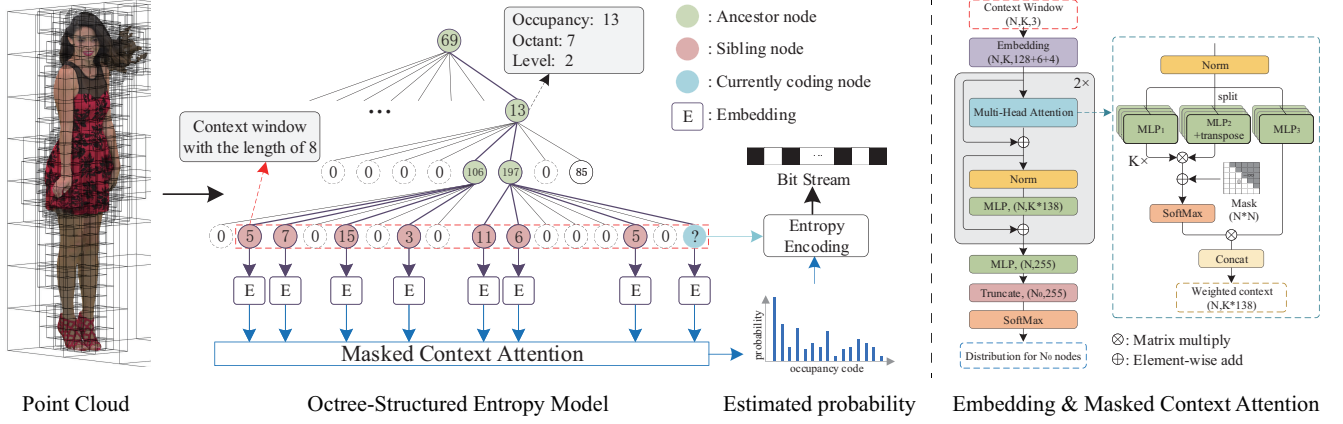
Figure 2: The overview architecture of OctAttention (Left). The number in the node indicates the corresponding occupancy code in decimal. The point cloud is first encoded into an octree, in which an octree node is characterized by its occupancy code, level, and octant. These 3 features are embedded to 128,6,4 dimensions vectors separately. For example, we construct a context window (red) with the level number $K = 4$ and length $N = 8$. The involvement of 3 layers of ancestors in the context is marked green. While encoding a node (blue), the context in the window is then fed into a masked context attention module (Right) and eventually utilized to model the occupancy code distribution by multi-layer perceptron (MLP). Finally, we use the estimated distribution to encode the serialized occupancy code into the final compressed bit stream by the arithmetic encoder.

Methodology

We propose an extended context and a tree-structured attention mechanism, which is shown in Fig. 2. Intuitively, nodes with similar ancestors and siblings tend to follow a similar distribution. Therefore, the proposed context exploits features from siblings and their ancestors, which are beneficial for inference. To achieve accurate and flexible prediction with a large-scale context, we employ a tree-structured attention mechanism to determine the importance of each node in the context. Finally, we infer the occupancy of each octree node based on the attention context. We further propose a *mask* operation to counteract the increased coding time caused by involving the siblings in the context.

Octree Structure

Octree is an efficient approach to organize point cloud (see Fig. 1). Given a point cloud P, we translate the point cloud by an offset $= (\min(P_x), \min(P_y), \min(P_z))$, and quantize it by quantization step qs:

$$P_Q = \text{round}\left(\frac{P - \text{offset}}{\text{qs}}\right) \tag{1}$$

$$\text{qs} \geq \frac{\max(P) - \min(P)}{2^L - 1} \tag{2}$$

An octree divides the cube space along the maximum side length of the bounding box of P_Q into 8 equal octants recursively. The occupancy status of 8 children cubes constitutes an 8-bit binary occupancy code. Only the nonempty child cubes will be flagged as 1 and be further subdivided. Other unoccupied cubes will be flagged as 0. The division will be terminated at the given depth L. At the leaf nodes, an 8-bit occupancy code represents 8 cubes whose

side length equals the quantization step qs. Points in the P are aligned and merged with the nearest corresponding cube. The point cloud is reconstructed by inverse quantization: $\hat{P} = P_Q * \text{qs} + \text{offset}$. Given the reconstructed point cloud $\hat{P}$, the reconstruction error will be controlled in:

$$e = \max_i ||\hat{P}_i - P_i||_\infty \leq \frac{\text{qs}}{2} \tag{3}$$

The geometric compression loss of our method only comes from quantization error in Eq. (3). Therefore, we are allowed to increase the octree depth L to achieve arbitrary required accuracy.

Context Model

According to Shannon's source coding theorem (Shannon 2001), given an octree in a breadth-first search fashion $\mathbf{x} = \{x_1, \ldots, x_i, \ldots\}$, its bitrate is lower bounded by its information entropy. Since the ground truth distribution $P(\mathbf{x})$ is high-dimensional and unknown, we estimate a probability $Q(\mathbf{x})$ by a network with prior information to guide the arithmetic coder. The closer the estimated probability is to ground truth, the closer the estimated bitrate $\mathbb{E}_{\mathbf{x} \sim P}[-\log_2 Q(\mathbf{x})]$ is to its lower bound $\mathbb{E}_{\mathbf{x} \sim P}[-\log_2 P(\mathbf{x})]$. Therefore, we aim to estimate accurate occupancy distribution to reduce the bit rate. We believe that exploiting more features in the context benefits this goal. Large-scale context is significant in extending the receptive field of context and searching for dependency in a wide range. Sibling nodes also serve as a complement of low-level local geometry features. For example, leaf sibling nodes are likely to distribute on the neighboring surface in the point cloud and have similar geometry patterns, which the sibling context can capture.

We propose an expanded large-scale context to achieve more accurate probability estimation. We first traverse the

octree in breadth-first order. Then for each currently encoding node n_i in the sequence, we construct a context window $\{n_{i-N+1}, \ldots, n_k, \ldots, n_{i-1}, n_i\}$ with the length of N (see Fig. 2 Left). The currently encoding node is at the end of the window, and the local context window slides forward with the currently encoding node moving forward. In this manner, we select $N - 1$ sibling nodes to exploit the strong dependency among the nodes at the same depth. Considering the dependency between the node and its ancestor nodes, we further introduce $K - 1$ ancestors of the N nodes in the context window, respectively. In summary, we integrate $N * K$ related nodes and greatly expand the contexts. Specifically, we factorize distribution $Q(\mathbf{x})$ into a product of conditional probabilities of each occupancy symbol x_i as:

$$Q(\mathbf{x}) = \prod_i q_i \left(x_i \mid \boldsymbol{f}_{i-N+1}, \ldots, \boldsymbol{f}_k, \ldots, \boldsymbol{f}_i; \mathbf{w} \right) \quad (4)$$

where $x_i \in \{1, 2, \ldots, 255\}$ is the occupancy symbol of currently encoding node n_i. High-dimensional vector $\boldsymbol{f}_k = [\boldsymbol{h}_k^{(0)}, \boldsymbol{h}_k^{(1)}, \boldsymbol{h}_k^{(2)}, \ldots, \boldsymbol{h}_k^{(K-1)}]$ denotes the concatenation of embedding features of nodes n_k and features of its $K - 1$ ancestors, which are defined as the embedding of their occupancy, depth and octant. $\mathbf{w}$ denotes the context model parameters.

Voxel-based methods naturally exploit the low-level geometry features preserved by a voxel representation. However, in an octree, these low-level features are hidden in the octree nodes. We employ an embedding for each node in the context window to reveal these features. Node information is embedded to a fixed-length vector respectively and then concatenated as $\boldsymbol{h}_k = [S_k W_1, L_k W_2, O_k W_3]$, where S_k, L_k, O_k are one-hot coded occupancy, level index and octant index, and W is their respective embedding matrix. Embedding also serves as normalization for the three different scale variables. It should be noted that the depth and octant of n_i are already available while decoding n_i, yet its occupancy code x_i is unknown, so we pad it with x_{i-1}.

In this manner, we utilize the features from sibling nodes and their ancestors, which are significant to prediction. Previous octree-based works (Huang et al. 2020; Biswas et al. 2020) ignored it. In these works, node n_i is assumed to be conditional independence under the condition $\boldsymbol{f}_i$. We strengthen this condition by expanding the context. Thus we make a more general assumption. Meanwhile, we avoid the inefficient sparse context and tremendous incremental computations caused by expanding the context in previous voxel-based works (Que, Lu, and Xu 2021). The receptive field of the proposed context can exceed 1000 octree nodes, which is extremely difficult to be achieved in voxel-based methods.

Tree-structured Attention

By expanding the context, more information is available for inference. However, a portion of nodes in the context is worthless. See Fig. 2 (Right). We adopt the self-attention mechanism (Vaswani et al. 2017) to discover similarity and strong dependency among the nodes, filter out irrelevant nodes, and emphasize useful ones. Since self-attention ef-
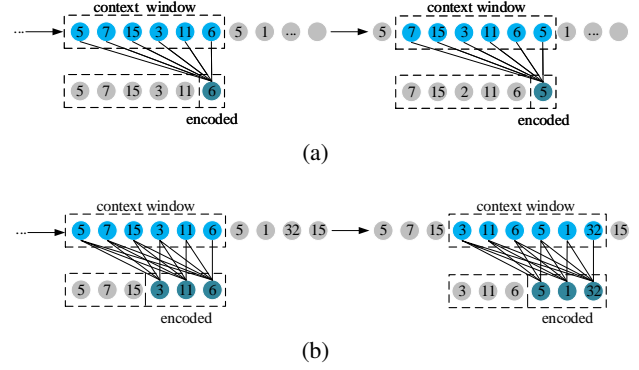

(a)

(b)

Figure 3: Illustration of *mask* operation, which allows us to encode N_0 nodes at the same time in one propagation. The vertical line represents the involvement of octant and level information. (a) $N_0 = 1$ and receptive field of encoded nodes is 6. (b) $N_0 = 3$ and receptive field of encoded nodes are 4,5,6. Meanwhile, it reduces encoding time by 3 times compared to (a).

fectively tackles long-range dependence, it is also appropriate in a large-scale context. Meanwhile, attention achieves flexible inference by evaluating weights varying with the input. Intuitively, nodes having similar ancestors and siblings tend to follow similar distribution, hence it is reasonable to estimate the occupancy based on similarity among siblings to the estimated node n_i. We omit shortcut and normalization for brief illustration. In the head t ($t = 0, 1, \ldots, K - 1$), the attention score scalar in attention map between the mth and n^{th} sibling node in the context window is defined as:

$$a_{m,n}^{(t)} = \frac{\exp(\text{MLP}_1(\boldsymbol{h}_m^{(t)}) \cdot \text{MLP}_2^{\text{T}}(\boldsymbol{h}_n^{(t)}))}{\sum_{k=i-N+1}^m \exp(\text{MLP}_1(\boldsymbol{h}_m^{(t)}) \cdot \text{MLP}_2^{\text{T}}(\boldsymbol{h}_k^{(t)}))} \quad (5)$$

where $n \leq m, m = i - N + 1, \ldots, i$. The summation for node n_m ends at m since a *mask* operation is applied to attention map to achieve fast encoding, which is discussed in the next section. With attention mechanism, we can draw weighted context as:

$$\boldsymbol{S}_i^{(t)} = \sum_{k=i-N+1}^i a_{i,k}^{(t)} \cdot \text{MLP}_3(\boldsymbol{h}_k^{(t)}) \quad (6)$$

In summary, the context is fed to 2 layers of multi-head self-attention $\text{MultiHead}(\boldsymbol{f}, i) = \text{MLP}([\boldsymbol{S}_i^{(0)}, \ldots, \boldsymbol{S}_i^{(K-1)}])$ and multi-layer perception (MLP) successively, and finally outputs a 255-dimensional probabilities for n_i:

$$q_i \left(\cdot \mid \boldsymbol{f}; \mathbf{w} \right) = \text{softmax}(\text{MLP}(\text{MultiHead}^{(2)}(\boldsymbol{f}, i))) \quad (7)$$

Mask operation

The estimated probability is adopted to guide the arithmetic coder, which codes the octree nodes sequentially in a lossless way. Previous methods (Huang et al. 2020; Que, Lu, and Xu 2021) exclude siblings from context and only depend on ancestors. Hence they can naturally parallelize encoding and decoding within each level. Without mask operation, as shown in Fig. 3(a), given a sliding window, we adopt

all $N-1$ siblings in the context to predict the last node in the context window. It is difficult to achieve the same parallelization since we can only decode the last node in one propagation. To reduce encoding time, we introduce a *mask* operation in Eq. (5) which assigns a varied-length receptive field for each node. Each node is restricted to the access of the previous nodes in the context window at training and testing. As shown in Fig. 3(b), in this way, the Eq. (7) can equally apply to the last N_0 nodes. Hence we are allowed to encode them simultaneously in one propagation. While encoding the jth node in the window, only j nodes are available for inference. Compared to maximum receptive field in Fig. 3(a), on average, receptive field of each node shrinks from N to $(2N - N_0 + 1)/2$. Nevertheless, the coding time reduces by N_0 times. Although the receptive field shrinks, we find there is negligible performance loss as the result of the *mask* operation during the training. The parameter N_0 balances the receptive field and coding time.

Learning

We optimize the cross entropy between the predicted occupancy code and ground-truth, which is defined as:

$$\ell = -\sum_i \log q_i \left(x_i \mid \boldsymbol{f}; \mathbf{w} \right) \tag{8}$$

Here, $q_i \left(x_i \mid \boldsymbol{f}; \mathbf{w} \right)$ is the estimated probability of occupancy code x_i at node n_i, which is defined in Eq. (7).

Experiments

Datasets

LiDAR Dataset SemanticKITTI (Behley et al. 2019) is a large public LiDAR point cloud dataset involving point clouds with different densities. It is collected from a Velodyne HDL-64E sensor and contains 43552 scans with 4549 million points. Following VoxelContext-Net (Que, Lu, and Xu 2021), we use the official training/testing split, where sequences 00 to 10 (include 23201 scans) are for training, and 11 to 21 (include 20351 scans) are for testing.

Object Dataset MPEG 8i (Eugene et al. 2017) includes sequences of smooth surface and complete human shape point clouds with 10 and 12-bit precision. Microsoft Voxelized Upper Bodies (Charles et al. 2016) is a dynamic voxelized point cloud dataset containing five half-body subjects sequences with 9 and 10-bit precision. Following the setting of VoxelDNN (Nguyen et al. 2021a), we use Soldier10 and Longdress10 sequences in MPEG 8i, Andrew10, David10, Sarah10 sequences in MVUB for training. We select several point cloud sequences with different resolutions in 8iVSLF and MVUB for testing. All testing point clouds were not used during training.

Experimental Details

Baseline In the static LiDAR lossy compression experiment, we quantize the point cloud P by Eq. (1), and set L from 8 to 12 to perform RD control. We compare our method against state-of-the-art methods VoxelContext-Net (Que, Lu, and Xu 2021) and Octsqueeze (Huang et al. 2020)

in LiDAR compression. Since the source code of the above methods is not publicly available, we keep our training/testing setting consistent with VoxelContext-Net and use the results in the paper.

In object point cloud compression, we set qs $= 1$ in Eq. (1). to perform lossless compression. We compare our method with the hand-crafted intra-frame octree-based contexts model P(full) from (Garcia et al. 2019), state-of-the-art compression method VoxelDNN and its fast version MSVoxelDNN (Nguyen et al. 2021b). We set the training condition following VoxelDNN and test the models on different depth data. In addition, we also compare against the most usual hand-crafted methods: G-PCC from MPEG standard in the latest version (TMC13 v14.0) (MPEG 2021).

Training and Testing Strategy In the training procedure for static LiDAR compression, we train a single model with the max octree depth (12-level) as our model can learn the distribution of all layers in one model. While testing, we truncate the octree over 8-12 levels to evaluate our model at different bitrates. For object dataset compression, we train one model using the octree sequence data converted from point clouds at depths 9 and 10. We also evaluate our model on data with different geometry precision to verify robustness. We implement our model in PyTorch and perform the training/testing with Xeon E5-2637 CPU and one NVIDIA TITAN Xp GPU (12G memory). We use batch sizes of 32, epochs of 8 and Adam optimizer with a learning rate of 1e-3. It takes 2 days to train our model in each experiment. Occupancy, level index, and octant index are embedded into 128, 6, and 4 dimensions, respectively. We set $K = 4$, $N = N_0 = 1024$ and use 2 layers and 4 heads in multi-head self-attention in experiments unless otherwise specified.

Evaluation Metrics It is important to adopt the same evaluation metrics to make a fair comparison. Following the MPEG standards (Schwarz et al. 2018), we use two standard metrics which measure geometry reconstruction quality named point to point PSNR (D1 PSNR) and point to plane PSNR (D2 PSNR) in lossy geometry compression. Both of them can be calculated by the MPEG tool *pc_error*. We normalize SemanticKITTI into $[-1, 1]^3$, and estimate the normal at each point using the MATLAB function *pcnormals*. We also report chamfer distance (CD). We set PSNR peak value $r = 1$ following VoxelContext-Net (Que, Lu, and Xu 2021). We correct its results from correspondence with the authors by eliminating inconsistencies in the PSNR formula. As for the object dataset, we adopt the official default configuration in G-PCC. We use bits per point (bpp) to measure the compression ratio. Unless otherwise specified, all distortion curves and bitrates are obtained by averaging over sequence.

Experiment Results

Results for Static LiDAR Compression The rate-distortion curves of LiDAR compression are shown in Fig. 4. We compare our method with VoxelContext-Net without coordinate refinement model (*i.e.*, VoxelContext-Net w/o CRM) for fairness, as post-processing is irrelevant to compression performance evaluation. Our method outperforms other baselines at all bitrates. On average, our approach
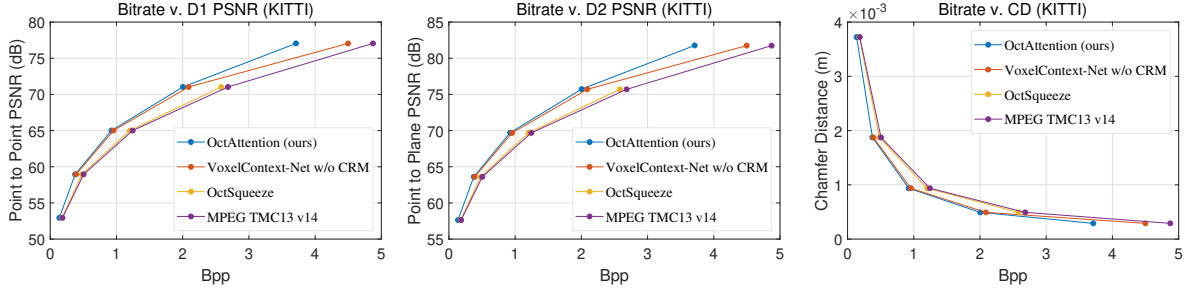
Figure 4: Experimental results of different methods on SemanticKITTI at different bitrates.

Table 1: Average bits per point (bpp) results of different methods on MPEG 8i and MVUB.

Point Cloud	Frames	P(full)	G-PCC	VoxelDNN	MSVoxelDNN	OctAttention (ours)	Gain over G-PCC
Microsoft Voxelized Upper Bodies (MVUB)							
Phil9	245	1.43	1.23	0.92	-	**0.83**	-32.60%
Phil10	245	-	1.07	0.83	1.02	**0.79**	-25.97%
Ricardo9	216	1.34	1.04	0.72	-	**0.72**	-31.20%
Ricardo10	216	-	1.07	0.75	0.95	**0.72**	-32.53%
Average	-	1.34	0.95	0.81	0.99	**0.76**	-30.58%
8i Voxelized Full Bodies (MPEG 8i)							
Loot10	300	1.03	0.95	0.64	0.73	**0.62**	-35.12%
Redandblack10	300	1.23	1.09	0.73	0.87	**0.73**	-32.84%
Boxer9/10	1	-	0.96/0.94	0.76/-	-/0.70	**0.60/0.59**	-37.94%
Thaidancer9/10	1	-	0.99/0.99	0.81/-	-/0.85	**0.64/0.65**	-34.51%
Average	-	1.13	0.99	0.73	0.79	**0.64**	-35.10%

(*i.e.*, OctAttention) saves 25.4% bitrates on SemanticKITTI compared with G-PCC, while OctSqueeze only saves less than 4% bitrates. Our method achieves more than 11% relative reduction in bitrate versus the state-of-the-art method VoxelContext-Net at high bitrates. It may be due to the voxel-based method failing in the sparse scenario that lacks occupied voxels. The experiment results demonstrate the effectiveness of our large receptive field context model.

Results for Object Point Cloud Compression In Table 1, we provide the bpp results for lossless compression on object point clouds. Our method outperforms VoxelDNN and achieves 32.8% gain over G-PCC on average.

Ablation Study and Analysis

Context Window Length We perform an ablation experiment on SemanticKITTI to demonstrate the effectiveness of large receptive field context. We set $N_0 = N$ so that each node is predicted only once and the average receptive filed is $(N+1)/2$. We then alter the context window size N, and the number of parameters in our model does not increase. As shown in Table 2, we can save 14% bitrates by enlarging the context window size from 8 to 1024. The encoding time decreases with increased context window size due to our *mask* operation, where we decrease I/O by N_0 times. Decoding time does not reduce significantly since the time consumption is primarily in I/O. To balance the decoding time and

performance, we set $N = 1024$.

Table 2: Performance and runtime when using various context window size N. 'D' stands for the max octree depth and 'time' is the duration for coding per 1000 octree nodes.

Size N	Bpp on SemanticKITTI			time / K nodes (s)	
	D=8	D=10	D=12	encode	decode
8	0.160	1.086	4.371	0.1628	1.5091
16	0.158	1.081	4.334	0.0828	1.5218
32	0.148	1.008	4.069	0.0387	1.5272
256	0.142	0.959	3.823	0.0055	1.5336
512	0.142	0.956	3.806	0.0028	1.5949
1024	**0.139**	**0.939**	**3.740**	**0.0015**	1.6093

Table 3: Performance ablation study on backbone and sibling context. Backbone 'MLP' stands for the shared multi-layer perception architecture like OctSqueeze.

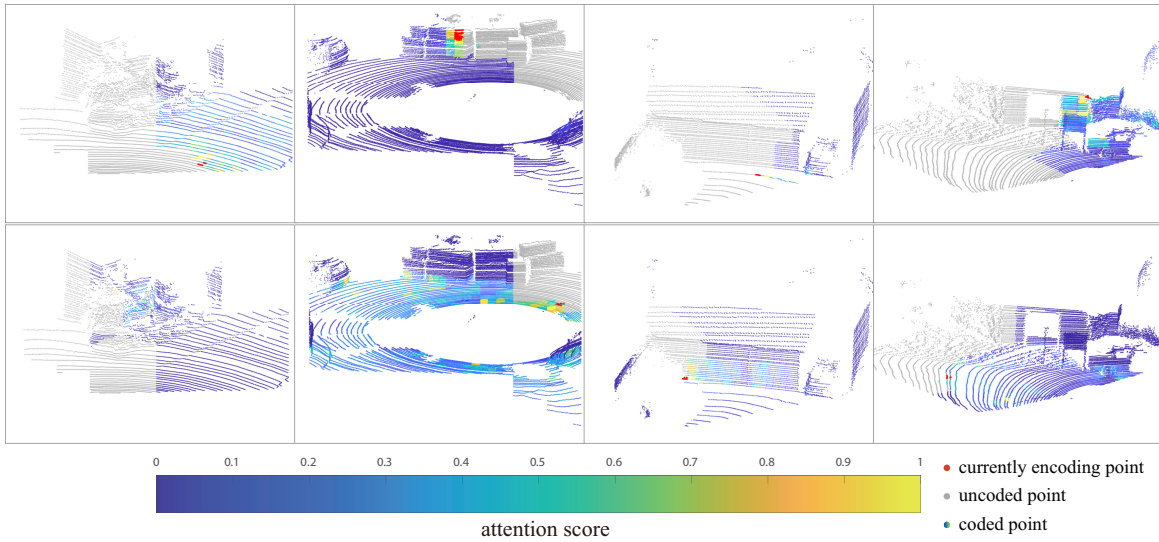| Backbone	Sibling	Bpp on SemanticKITTI				
		D=8	D=9	D=10	D=11	D=12
MLP		0.195	0.533	1.315	2.847	5.246
	✓	0.163	0.450	1.117	2.438	4.512
Attention		0.197	0.530	1.289	2.767	5.073
	✓	**0.142**	**0.391**	**0.959**	**2.074**	**3.823**

Figure 5: Attention score visualization. Each subfigure is a context window with size $N = 1024$. Points in the currently encoding octree node i voxel are colored in red, and other points in node k are colored as their normalized attention score $a_{i,k}$ to node i. Unavailable points (*i.e.*, they have not been coded yet) are colored in grey.

Effectiveness of Attention and Sibling Context As the visualization in Fig. 5, the attention mechanism discovers the similarity among points in a context window according to geometry patterns such as line, plane, surface, and curvature. The node with the highest attention score to the currently encoding nodes (red points) is colored in yellow. It confirms that the attention mechanism can predict occupancy by integrating similar features from sibling nodes in a large-scale context. We set $N = 256$ and removed the attention and sibling features for the ablation study. The result in Table 3 further illustrates the effectiveness of attention mechanism and sibling involved context.
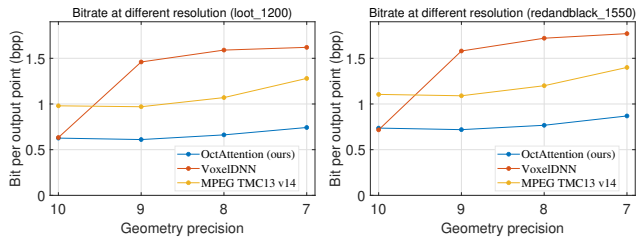

Figure 6: Performance at different geometry precision.

Robustness to Varying Point Density To evaluate OctAttention robustness to varying point density, we test point clouds with additional geometry precision and point density. See in Fig. 6. VoxelDNN only outperforms G-PCC on the geometry precision of 10. Its performance drops on sparse point clouds due to the lack of occupied voxel in the context. Since we adopt an octree structure with a fixed-length context window, our method's performance is shown to be stable with varying point densities.

Table 4: Coding time (in *seconds*) on object point clouds.

Average encoding time (s)				
Methods	G-PCC	VoxelDNN	MSVoxelDNN	Ours
MPEG 8i	0.35	2459	54	0.41
MVUB	0.31	4124	85	0.38
Average decoding time (s)				
Methods	G-PCC	VoxelDNN	MSVoxelDNN	Ours
MPEG 8i	0.19	6274	58	769
MVUB	0.16	10332	92	659

Runtime The number of parameters in our model is 2.67M. See table 4. Our method achieves better compression performance with 95% encoding time saving and 91% decoding time saving than VoxelDNN. Our approach can be applied to real-time point cloud encoding and offline point cloud compression. We believe it is possible to speed up the decoding by dividing the octree into disjoint subtrees and developing a GPU-based algorithm like an arithmetic encoder.

Conclusion

We proposed a novel octree-based entropy model called *OctAttention* for sparse and dense point cloud geometry compression by exploiting large-scale contexts. Specifically, we extend the context and introduce sibling nodes in the octree. We employ the attention mechanism to emphasize the significant nodes to utilize these abundant features. We further propose a *mask* operation to achieve parallel encoding under the condition of introducing siblings in the context. We evaluate our method on both the LiDAR and object point cloud datasets. The result demonstrates that the proposed method achieves state-of-the-art on both types of datasets.

References

3DG. 2021. G-PCC test model v14 user manual. *ISO/IEC JTC1/SC29/WG7 W20346 output document N00094*.

Behley, J.; Garbade, M.; Milioto, A.; Quenzel, J.; Behnke, S.; Stachniss, C.; and Gall, J. 2019. SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 9296–9306.

Biswas, S.; Liu, J.; Wong, K.; Wang, S.; and Urtasun, R. 2020. MuSCLE: Multi Sweep Compression of LiDAR using Deep Entropy Models. *Advances in Neural Information Processing Systems (NeurIPS)*, 33: 1–12.

Charles, L.; Qin, C.; Sergio, O. E.; and Philip, A. C. 2016. Microsoft voxelized upper bodies - a voxelized point cloud dataset. *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document m38673/M7201*.

Chou, P. A.; Koroteev, M.; and Krivokuća, M. 2019. A volumetric approach to point cloud compression—Part I: Attribute compression. *IEEE Transactions on Image Processing (TIP)*, 29: 2203–2216.

Devillers, O.; and Gandoin, P.-M. 2000. Geometric compression for interactive transmission. In *Proceedings Visualization 2000*, 319–326.

Eugene, d.; Bob, H.; Taos, M.; and Philip, A. C. 2017. 8i Voxelized Full Bodies - A Voxelized Point Cloud Dataset. *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006*.

Gao, Z.; Flynn, D.; Tourapis, A.; and Mammou, K. 2019. Predictive Geometry Coding. *ISO/IEC JTC1/SC29/WG11 M51012*.

Garcia, D. C.; and de Queiroz, R. L. 2018. Intra-Frame Context-Based Octree Coding for Point-Cloud Geometry. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, 1807–1811.

Garcia, D. C.; Fonseca, T. A.; Ferreira, R. U.; and de Queiroz, R. L. 2019. Geometry coding for dynamic voxelized point clouds using octrees and multiple contexts. *IEEE Transactions on Image Processing (TIP)*, 29: 313–322.

Guarda, A. F.; Rodrigues, N. M.; and Pereira, F. 2020. Deep Learning-based Point Cloud Geometry Coding with Resolution Scalability. In *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, 1–6. IEEE.

Gumhold, S.; Kami, Z.; Isenburg, M.; and Seidel, H.-P. 2005. Predictive point-cloud compression. In *ACM SIGGRAPH 2005 Sketches*, 137–es. Citeseer.

Huang, L.; Wang, S.; Wong, K.; Liu, J.; and Urtasun, R. 2020. OctSqueeze: Octree-Structured Entropy Model for LiDAR Compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 1310–1320.

Kammerl, J.; Blodow, N.; Rusu, R. B.; Gedikli, S.; Beetz, M.; and Steinbach, E. 2012. Real-time compression of point cloud streams. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, 778–785.

Kathariya, B.; Li, L.; Li, Z.; Alvarez, J.; and Chen, J. 2018a. Scalable point cloud geometry coding with binary tree embedded quadtree. In *2018 IEEE International Conference on Multimedia and Expo (ICME)*, 1–6. IEEE.

Kathariya, B.; Li, L.; Li, Z.; and Alvarez, J. R. 2018b. Lossless dynamic point cloud geometry compression with inter compensation and traveling salesman prediction. In *2018 Data Compression Conference (DCC)*, 414–414.

Lasserre, S.; Flynn, D.; and Qu, S. 2019. Using neighbouring nodes for the compression of octrees representing the geometry of point clouds. In *the 10th ACM Multimedia Systems Conference*, 145–153.

Lei, H.; Akhtar, N.; and Mian, A. 2019. Octree guided cnn with spherical kernels for 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 9631–9640.

MPEG. 2021. Mpeg g-pcc tmc13. https://github.com/MPEGGroup/mpeg-pcc-tmc13. Accessed: 2021.

Nguyen, D. T.; Quach, M.; Valenzise, G.; and Duhamel, P. 2021a. Learning-Based Lossless Compression of 3D Point Cloud Geometry. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4220–4224.

Nguyen, D. T.; Quach, M.; Valenzise, G.; and Duhamel, P. 2021b. Multiscale deep context modeling for lossless point cloud geometry compression. In *2021 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, 1–6. IEEE.

Quach, M.; Valenzise, G.; and Dufaux, F. 2019. Learning Convolutional Transforms for Lossy Point Cloud Geometry Compression. In *2019 IEEE International Conference on Image Processing (ICIP)*, 4320–4324.

Quach, M.; Valenzise, G.; and Dufaux, F. 2020. Improved Deep Point Cloud Geometry Compression. In *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, 1–6.

Que, Z.; Lu, G.; and Xu, D. 2021. VoxelContext-Net: An Octree based Framework for Point Cloud Compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 6042–6051.

Riegler, G.; Osman Ulusoy, A.; and Geiger, A. 2017. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 3577–3586.

Schnabel, R.; and Klein, R. 2006. Octree-based Point-Cloud Compression. In *Symposium on Point-Based Graphics*, 111–120.

Schwarz, S.; Preda, M.; Baroncini, V.; Budagavi, M.; Cesar, P.; Chou, P. A.; Cohen, R. A.; Krivokuća, M.; Lasserre, S.; Li, Z.; Llach, J.; Mammou, K.; Mekuria, R.; Nakagami, O.; Siahaan, E.; Tabatabai, A.; Tourapis, A. M.; and Zakharchenko, V. 2018. Emerging MPEG Standards for Point Cloud Compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1): 133–148.

Shannon, C. E. 2001. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5(1): 3–55.

Shao, Y.; Zhang, Q.; Li, G.; Li, Z.; and Li, L. 2018. Hybrid point cloud attribute compression using slice-based layered structure and block-based intra prediction. In *Proceedings of the 26th ACM international conference on Multimedia (ACM MM)*, 1199–1207.

Shao, Y.; Zhang, Z.; Li, Z.; Fan, K.; and Li, G. 2017. Attribute compression of 3D point clouds using Laplacian sparsity optimized graph transform. In *2017 IEEE Visual Communications and Image Processing (VCIP)*, 1–4. IEEE.

Song, F.; Shao, Y.; Gao, W.; Wang, H.; and Li, T. 2021. Layer-Wise Geometry Aggregation Framework for Lossless LiDAR Point Cloud Compression. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 31(12): 4603–4616.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*, 6000–6010.

Wang, J.; Zhu, H.; Liu, H.; and Ma, Z. 2021. Lossy Point Cloud Geometry Compression via End-to-End Learning. *IEEE Transactions on Circuits and Systems for Video Technology (TCSVT)*, 31(12): 4909–4923.

Wang, P.-S.; Liu, Y.; Guo, Y.-X.; Sun, C.-Y.; and Tong, X. 2017. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions On Graphics (TOG)*, 36(4): 1–11.

Zhang, X.; Gao, W.; and Liu, S. 2020. Implicit Geometry Partition for Point Cloud Compression. In *2020 Data Compression Conference (DCC)*, 73–82.