OctFormer: Efficient Octree-Based Transformer for Point Cloud Compression with Local Enhancement

Mingyue Cui[1], **Junhua Long**[1], **Mingjian Feng** [1], **Boyang Li** [1], **Kai Huang** [1,2*]

[1] School of Computer Science and Engineering, Sun Yat-sen University
[2] Shenzhen Institute, Sun Yat-sen University
{cuimy,longjh7,fengmj8,liby3}@mail2.sysu.edu.cn, huangk36@mail.sysu.edu.cn

Abstract

Point cloud compression with a higher compression ratio and tiny loss is essential for efficient data transportation. However, previous methods that depend on 3D convolution or frequent multi-head self-attention operations bring huge computations. To address this problem, we propose an octree-based Transformer compression method called OctFormer, which does not rely on the occupancy information of sibling nodes. Our method uses non-overlapped context windows to construct octree node sequences and share the result of a multi-head self-attention operation among a sequence of nodes. Besides, we introduce a locally-enhance module for exploiting the sibling features and a positional encoding generator for enhancing the translation invariance of the octree node sequence. Compared to the previous state-of-the-art works, our method obtains up to 17% Bpp savings compared to the voxel-context-based baseline and saves an overall 99% coding time compared to the attention-based baseline.

Introduction

In the past few decades, the LiDAR sensor has proven to be crucial for various types of intelligent robot applications (Li and Ibanez-Guzman 2020; Zou et al. 2022). It can accurately capture the 3D geometry information of scenes, which makes it widely used in the detection, segmentation, planning, and other downstream perception tasks. However, storing and transmitting point cloud data require high costs. For example, a single Velodyne LiDAR of HDL64 generates over 100,000 points per sweep, and about 2.88 million points are generated per second. Hence, it is necessary to develop an efficient point cloud compression method.

Due to the disorder and sparsity of the point cloud, compressing the point cloud into a tiny capacity is a tough challenge compared to its well-studied image and video counterparts. Fortunately, several schemes for point cloud geometry and attribute compression (Google 2017; Chou, Koroteev, and Krivokuća 2019; Guarda, Rodrigues, and Pereira 2020) have been explored in prior research works. (Limuti, Polo, and Milani 2018) presents a voxelized dynamic point cloud coding scheme that combines a cellular automata block reversible transform for geometric data with a region adaptive transform for color data. (Quach, Valenzise, and Dufaux 2019) designs a model for learning analysis and synthesis transforms suitable for voxel-based point cloud geometry compression. Further, (Nguyen et al. 2021) proposes a deep auto-regressive generative method to estimate the occupancy probability of each voxel. In general, these methods do not fully use spatial context information and also require a huge amount of computing power.

Compared with point cloud coding using voxel grids, octree has more advantages benefiting from its higher coding efficiency (Huang et al. 2006; Liu et al. 2019). An octree node uses an 8-bit occupancy to represent the distribution of children nodes. By combining the deep entropy model along with an arithmetic encoder, the octree encoded point cloud is further compressed into a compact bitstream. Actually, in octree encoding process, the main problem faced is how to efficiently extract strong prior information between spatial neighboring nodes, especially for the relationship between sibling nodes. Sibling nodes can provide low-level local geometry features, which are significant for exploiting geometry redundancy. Unlike ancestor nodes' relationships, it is difficult to obtain the neighbor geometric relationship of sibling nodes by traversing the octree directly. Therefore, a highly parallel sophisticated method for point cloud compression is needed, and spatial context information is fully considered.

In this paper, we propose a Transformer-based entropy model called OctFormer, which can compress the point cloud efficiently. OctFormer uses non-overlapped context windows to construct sequences and shares the results of the expensive multi-head self-attention (MSA) operation for a sequence of nodes, which significantly reduces time overhead. Subsequently, we propose an octree-based locally-enhanced feed-forward network (OctLeFF) and octree-based positional encoding generator (OctPEG) to help the model fit the octree node sequence data better. We use OctPEG instead of traditional absolute positional encoding (APE) to avoid unnecessary positional features, which enhances the translation-invariance of the octree node sequence. Our method only needs prior knowledge information of the traversal format from sibling and ancestor nodes, not additional coarse neighborhood information which is not readily available, such as the occupancy information of sibling nodes or local voxel context.

[*]Corresponding author.

We compare the proposed model with state-of-the-art methods (Graziosi et al. 2020; Que, Lu, and Xu 2021; Huang et al. 2020; Fu et al. 2022) on 3D point cloud datasets SemanticKITTI (Behley et al. 2019) and ScanNet (Dai et al. 2017). The experiments show that our method outperforms these methods, which are only designed for a specific category of point clouds. The contributions of our work can be summarized as:

- We propose a novel octree-based entropy model called OctFormer for point cloud geometry compression, which uses non-overlapped context windows to construct sequences.
- Our OctFormer shares the MSA results between a sequence of traversal node information and does not need to use the occupancies of sibling nodes, which can be applied to efficient point cloud encoding and decoding.
- We propose a local-enhancing module replacing the original feed-forward network in Transformer. This simple design exploits the local features and improves the compression performance.
- To avoid unnecessary positional features caused by traditional APE, we introduce the OctPEG module which enhances the translation-invariance of the octree nodes when generating the sequence data.

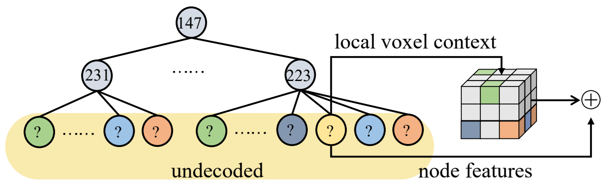
Related Work

Structured Point Cloud Compression

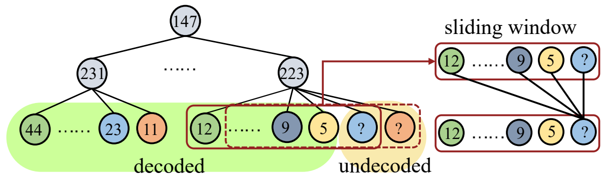
Structured data formats intuitively present the point cloud data and fit for convolution operations naturally. There are mainly voxel-based (Limuti, Polo, and Milani 2018; Quach, Valenzise, and Dufaux 2019; Kaya, Schwarz, and Tabus 2021) and image-based methods (Houshiar and Nüchter 2015; Sun et al. 2019; Feng, Liu, and Zhu 2020). The video point cloud compression standard (VPCC) (Schwarz et al. 2018) converts the 3D input point cloud into a set of 2D patches followed by a packing process. Such a way allows compressing the patches utilizing the existing video coding standards. (Houshiar and Nüchter 2015) proposes mapping the floating point measured ranges onto a three-channel image and then compressing it. The performance of image-based methods is limited due to the loss of spatial information inherent. (Wang et al. 2021) voxelizes point clouds into non-overlapped 3D cubes, and designs a 3D-convolution-based variational autoencoder to compress the point clouds. However, the voxel-based method is sensitive to the density variation and fails for the sparse point clouds. High computational costs are also prohibitive.

Unstructured Point Cloud Encoding

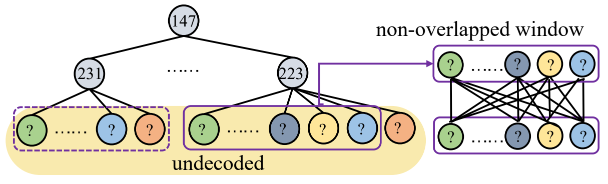
Octree-based methods have higher coding efficiency through hierarchical representation. The MPEG point cloud geometry compression standard (GPCC) (Graziosi et al. 2020) compresses the point cloud geometry information by exploiting an octree-based encoding strategy. It predicts attributes and encodes or transmits them in a scalable manner. (Dricot, Pereira, and Ascenso 2018) optimizes the octree partitioning decisions based on a rate-distortion optimization process to adapt the content. (Wen et al. 2020)



(a) VoxelContext-Net method



(b) OctAttention method



(c) Our proposed OctFormer

Figure 1: Comparison between different methods of extracting node features, where the occupancy is represented by decimal digits in the node. For a sequence of octree nodes, our method can share the MSA results.

adopts an adaptive patch generation module through adaptive octree-based decomposition and clustering processing to avoid the reconstruction error prorogation. (Garcia et al. 2020) develops a super-resolution technique to generate possible contexts based on octrees that can be arithmetically encoded. OctSqueeze (Huang et al. 2020) is the first octree-based deep learning entropy model by gathering context information about conditions on ancestor nodes. However, the prior octree-based methods seldom focus on the neighborhood contexts from the sibling nodes, which can not fully extract the local spatial information of the nodes.

Recently, VoxelContext-Net (Que, Lu, and Xu 2021) proposes introducing voxel context in the tree-structured deep model for a more precise prediction of the node's occupancy distribution. It employs 3D convolution on the generated local voxel context to encode the neighboring spatial information for each node in the constructed octree. However, this method has a limited receptive field of neighborhood information in fixed-size voxels and causes a lot of computations due to introducing voxel encoding, especially for higher resolution. Furthermore, OctAttention (Fu et al. 2022) designs a conditional entropy model with a large receptive field that models the sibling and ancestor contexts to exploit the strong dependencies among the neighboring nodes. Additionally, OctAttention uses the mask operation to encode multiple nodes in parallel. Although OctAttention obtains excellent compression performance, the resulting high time overhead is unacceptable. OctAttention directly uses the occupancy as features from the sibling nodes, so decoding the current

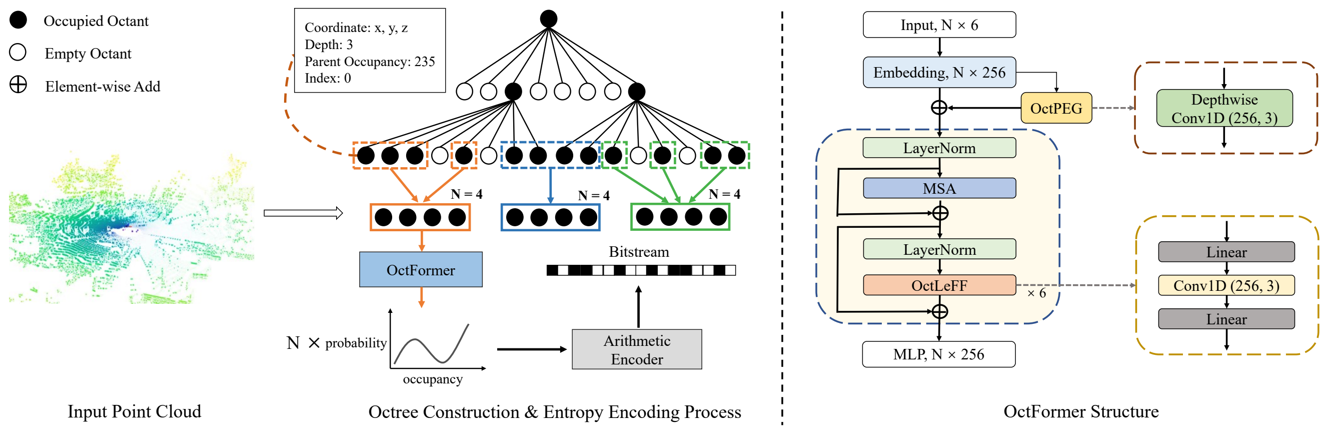


Figure 2: The overview of our proposed method. The input point cloud is first constructed as an octree. Each non-leaf node contains the feature of its xyz coordinate, depth, parent occupancy, and index. Non-overlapped context windows are constructed for generating input sequences. For example, we set the context window size $N = 4$ and take one sequence (orange) as input. The input features ($N * 6$) are firstly embedded into 256 dimension vectors ($N * 256$). After adding the positional encoding using OctPEG, the features ($N * 256$) are fed into Transformer blocks with our OctLeFF for feature extraction. Finally, a multi-layer perceptron (MLP) is employed to generate the occupancy distribution ($N * 256$) for N nodes in the context window.

octree node requires previous sibling nodes to be decoded. Frequent MSA operation for every prediction for one node brings expensive time costs.

As shown in Fig.1, we visualize the differences in feature extraction between different methods. Overall, our proposed OctFormer has the following advantages:

1. Compared with VoxelContext-Net, our proposed method considers higher resolution features (*i.e.*, from sibling nodes) and does not need to introduce computationally expensive 3D convolution on generated voxel grids.
2. Compared with OctAttention, our method does not introduce sibling occupancy as a feature. Our method eliminates the decoding dependencies between the nodes and shares the result of a MSA operation within the context window, which achieves several times to hundreds of times the time performance improvement.
3. We design OctLeFF and OctPEG modules to further improve the compression performance, which is also verified in both offline open source datasets and the practical downstream task (*i.e.*, segmentation).

The Proposed Method

As shown in Fig.2, we propose an octree-based Transformer with the local enhancement for point cloud compression called OctFormer. We organize point cloud data with octrees and then design a Transformer-based deep entropy model along with an arithmetic encoder to compress the octree node sequences. The octree node sequences are constructed by non-overlapped context windows. By not introducing sibling occupancy as features, our OctFormer eliminates the decoding dependencies between the nodes and shares the result of a multi-head self-attention operation (MSA) within the context window, which hugely reduces the computations. Considering that the sibling nodes tend to have similar occupancies, we introduce an octree locally-enhanced feed-forward network to better capture the local features. Besides, we propose octree-based positional encoding instead of absolute positional encoding to keep the translation invariance of the input sequence. Finally, we predict the distribution of the node's occupancy based on our proposed OctFormer.

Octree Building

The octree is an efficient representation structure, especially for the sparse point cloud where most of the space is empty. We construct an octree from an input point cloud by recursively partitioning the space into 8 cubes of the same size until the max depth is reached or the cube is empty. The octree node uses an 8-bit occupancy to indicate whether the child node is empty or not. We represent the octree by using these non-leaf nodes' occupancies. For example, for the octree built in Fig.2, there are 15 non-leaf octree nodes in total and each non-leaf node requires 8 bits to store the occupancy. Thus, it requires 120 bits to store the octree if there is no entropy model to compress further. Using a breadth-first traversal, we serialize the octree into a bitstream and reconstruct the octree structure from the bitstream losslessly. By taking each leaf node's center as a point, we reconstruct the point cloud from the octree. Octree construction introduces quantization error, which is a lossy stage for the octree-based methods. As shown in Fig.3, we visualize octree occupancies with different depths. A deeper octree obtains more fine-grained leaf nodes and discovers precise geometric information. This means that the quality of the reconstructed point cloud depends on the max depth of the octree.

Our Deep Entropy Model

Suppose a sequence of occupancies for octree nodes are represented as $o = [o_1, o_2, ..., o_i, ..., o_n]$, where o_i represents the occupancy of the i-th octree node O_i. o_i can be represented using an 8-bit code, which indicates whether a

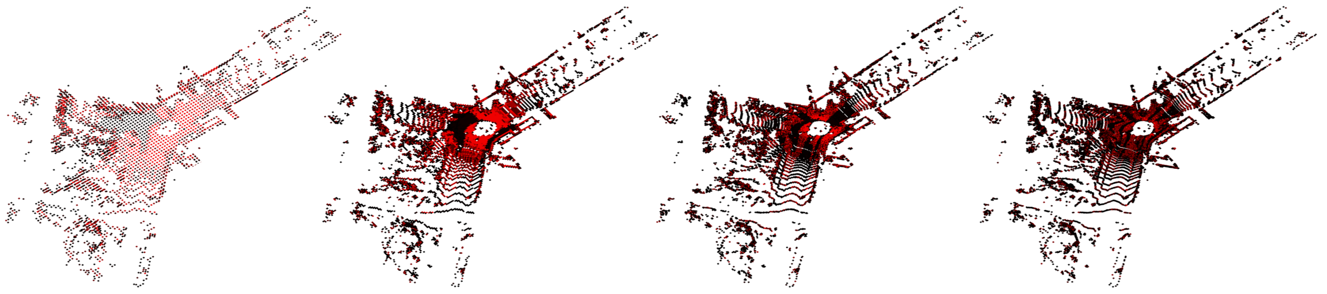


Figure 3: The visualization of octree occupancies at depths of 8, 9, 10, and 11. The color of each point is set to the value of the node's occupancy and similar color denotes similar occupancy. Points in the octree leaf nodes decrease with increased depth.

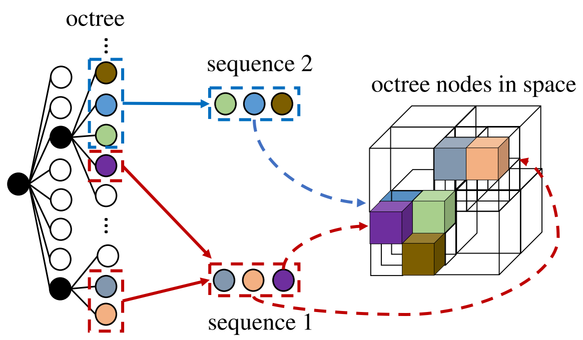


Figure 4: An illustration of the octree node sequences generating process. The nodes of a sequence come from different parts of the space, which is not friendly to the APE mechanism.

child node is empty or not. For example, suppose $o_i = [1,0,0,0,0,1,0,0]$, that means child node of O_i at index 0 and 5 is not empty, while the others are all empty.

Suppose the actual distribution of the sequence o is $P(o)$, and the predicted distribution is $Q(o)$. According to the information theory (Shannon 1948), the closer the estimated distribution $Q(o)$ is to the actual distribution $P(o)$, the fewer bits are needed to encode the sequence o. Thus the goal of the entropy model is to minimize the cross-entropy loss $\mathbb{E}_{o\sim P}[-log_2 Q(o)]$ between the $P(o)$ and $Q(o)$. We factorize $Q(o)$ into a product of predicted probability distributions of every octant occupancy o_i as follows:

$$Q(o) = \prod_i q_i(o_i|f_{i-j},...,f_i,...f_{i+k};w) \tag{1}$$

where $q_i(o_i|f_{i-j},...,f_i,...f_{i+k};w)$ is the estimated probability distribution of octree node occupancy o_i. w is the weight of the entropy model. f_i denotes the feature of the octree node O_i, which contains the xyz coordinates, index (0-7), depth (1-12), and parent occupancy (1-255). Assuming the distribution of the occupancy o_i only depends on the local context window of octree node O_i, we construct non-overlapped context window with size of N to form sequence, and use all the features from the sequence $(f_{i-j},...,f_i,...f_{i+k})$ to predict the distribution q_i, where $j+k-1=N$.

Octree-based Transformer Block

The structure of OctFormer is illustrated in Fig.2. We first employ an embedding layer to increase the feature dimension from 6 to 256. Then we introduce the octree positional encoding generator (OctPEG) to generate positional encoding for the input sequence. We use several Transformer blocks to extract features for every node, where an octree locally-enhanced feed-forward network (OctLeFF) is used to help capture the local features. Finally, a MLP is employed to generate the occupancy distribution of every node. The OctFormer block can be formally defined as:

$$\begin{aligned} X_0 &= OctPEG(Emb) + Emb, \\ X_l^{'} &= MSA(LN(X_{l-1})) + X_{l-1}, \\ X_l &= OctLeFF(LN(X_l^{'})) + X_l^{'} \end{aligned} \tag{2}$$

where Emb is the output of the Embedding layer. X_0 denotes the input before the first transformer block. X_l and $X_l^{'}$ are the outputs of the MSA module and OctLeFF module of the l-th transformer block, respectively. LN represents the layer normalization (Ba, Kiros, and Hinton 2016).

Computation Complexity Analysis Our OctFormer eliminates the decoding dependencies between the nodes and shares the results of the MSA operation within our constructed non-overlapped window, which hugely reduces the computation cost. Our encoding and decoding processes are identical. Suppose there are n nodes to be encoding/decoding, the context window size is N and the feature dimension is d. The computation complexity is $O(N^2 * d * \frac{n}{N})$ for us. OctAttention (Fu et al. 2022) can achieve $O(N^2 * d * \frac{n}{N})$ for encoding as well by applying their proposed masked operation and shrinking the average receptive field to $(N+1)/2$. However, because of taking the sibling occupancy as a feature, OctAttention is N times slower when decoding, which brings $O(N^2 * d * n)$ complexity. We can achieve N times less computation when decoding compared with OctAttention, which is a huge improvement because the context window size N is usually set to a big number such as 512 or 1024.

Octree Locally-enhanced Feed-Foward Network It has been proved that the vanilla Transformer can extract global features effectively but have limitations in capturing the local features (Li et al. 2021; Wu et al. 2021). As shown in

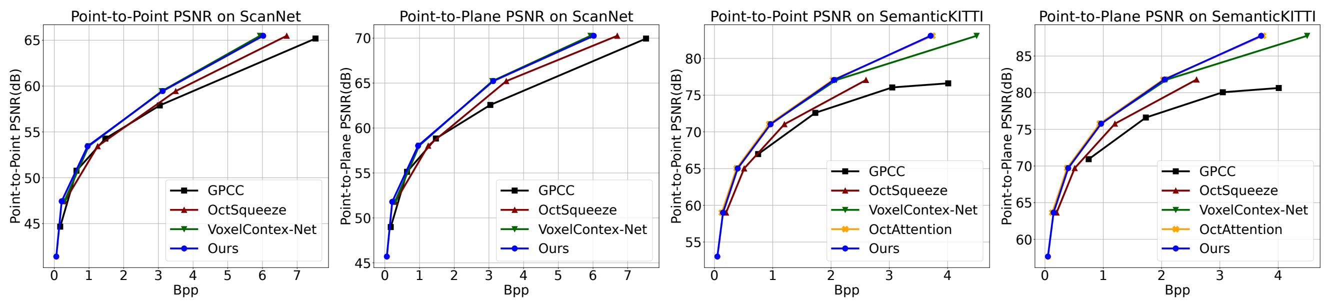
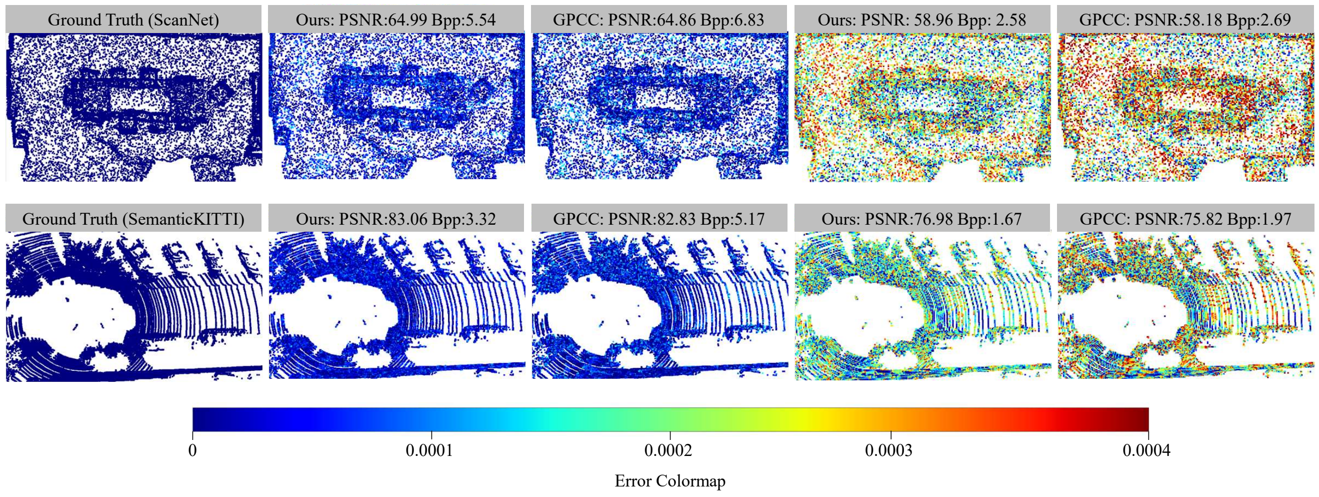


Figure 5: The quantitative results of different methods on ScanNet and SemanticKITTI, which is presented in the form of rate-distortion curves. It should be noted that we use the results of VoxelContext-Net not containing the coordinate refinement module, which is an algorithm to improve the PSNR value after point cloud reconstruction. Besides, for a fair comparison, we



Ground Truth (ScanNet)
Ours: PSNR:64.99 Bpp:5.54
GPCC: PSNR:64.86 Bpp:6.83
Ours: PSNR: 58.96 Bpp: 2.58
GPCC: PSNR:58.18 Bpp:2.69
Ground Truth (SemanticKITTI)
Ours: PSNR:83.06 Bpp:3.32
GPCC: PSNR:82.83 Bpp:5.17
Ours: PSNR:76.98 Bpp:1.67
GPCC: PSNR:75.82 Bpp:1.97
0
0.0001
0.0002
0.0003
0.0004
Error Colormap

Figure 6: The visualization of compression results of our method and GPCC on ScanNet (top) and SemanticKITTI (bottom) under different Bpps.

Fig.3, we visualize octree occupancies at different depths. We can observe that octree nodes tend to have similar occupancy among the siblings. Therefore, we propose using a 1D convolution layer to replace the feed-forward network in the vanilla Transformer to better capture the local features of the octree node. The structure of OctLeFF module is illustrated in Fig.2. We first use a linear projection layer to project the feature to another dimension and then apply a 1D convolution operation on the feature. Finally, we use another linear projection layer to project the feature back. Different from the original element-wise feed-forward network, the 1D convolution aggregates the features from the siblings, which enables our OctLeFF to capture the local features of the sibling nodes.

Octree Positional Encoding Generator The sequence generated by an octree should have translation-invariance. As shown in Fig.4, when generating the sequence data, the empty node is jumped over and the consequent N nodes are organized as a sequence. That means the sequence should have an order in space but the nodes at the same position of different sequences have totally different relative geometric locations in space. However, the absolute positional encoding (APE) scheme (Vaswani et al. 2017; Dosovitskiy et al. 2021) performs badly on the tasks which require translation-invariance because it adds unique positional encodings to each token. Inspired by (Chu et al. 2021), we propose using OctPEG to generate the positional encoding according to the local features. As shown in Fig.2, the key to our OctPEG is adopting a depth-wise convolution operation to extract the local feature before the transformer block. The OctPEG is applied in input embedding features $E \in \mathbb{R}^{N\times 256}$ to produce the position encodings $E' \in \mathbb{R}^{N\times 256}$. Compared with APE, the designed convolution introduces learnable parameters to enhance the translation-invariance for the node's embedding features and makes the position encodings the same size as the embedding features.

Learning

We optimize our deep entropy model using the cross entropy between the real and predicted occupancy of the non-leaf node. The loss function ℓ is defined as follows:

$$\ell = -\sum_{i} \log q_i(o_i | f_{i-j}, ..., f_i, ...f_{i+k}; w) \quad (3)$$

where $q_i(o_i | f_{i-j}, ..., f_i, ...f_{i+k}; w)$ is the estimated occupancy distribution of octree node occupancy o_i as defined in Eq.(1).

Experiments

Datasets

SemanticKITTI SemanticKITTI (Behley et al. 2019) is a large-scale point cloud dataset under different densities. It provides 22 sequences, 43,504 scans in total, with each scan containing over 120,000 points. We use the official train/test split, which is using sequences 00 to 10 for training and 11 to 21 for testing. Overall, we use 23,201 scans for training and 20,351 scans for testing.

Scannet ScanNet (Dai et al. 2017) is a popular indoor 3D point cloud dataset containing more than 1,500 scans. Train/test splits are the same as ScanNet's benchmark tasks, that is, we select 1,045 as the training set, 156 as the validation set, and 312 as the test set. We sample 50,000 points from each scan the same as (Que, Lu, and Xu 2021).

Experimental Details

Baseline Methods We compare our method against methods GPCC (Graziosi et al. 2020), OctSqueeze (Huang et al. 2020), VoxelContext-Net (Que, Lu, and Xu 2021), and OctAttention (Fu et al. 2022). They are also only designed for a specific category of point clouds. Since the source codes of some methods are not publicly available, we keep our training/testing setting consistent with them and use the results in their paper.

Implementation Details We set the maximum depth of the octree at 9 and 12 for ScanNet and SemanticKITTI respectively. At the training stage, we use all nodes of the octree to optimize the model. At the testing stage, we test the bitrate at a certain depth of the octree by directly truncating the octree at the depth and evaluate it.

Our model is implemented in Pytorch and trained/tested on a machine with Xeon Gold 6134 CPU and a single NVIDIA Tesla V100 GPU (32GB Memory). In the training procedure, the Adam optimizer is adopted and the learning rate is set to 1e-4 for the entropy model. It takes 2 days to train the model on ScanNet and 3 days on SemanticKITTI. We set the embedding size to 256 and the feed-forward dimension in the Transformer block to 1024. We use 6 layers and 8 heads for the MSA. By default, the context window size N is set to 1,024. We test the encoding/decoding time on 1,000 dummy octree nodes for the experiment below.

Evaluation Metrics We use bits per point (Bpp) as the compression ratio metric. As for evaluating the point cloud reconstruction quality, we use the point-to-point PSNR (D1 PSNR) and point-to-plane PSNR (D2 PSNR) proposed by the MPEG standards (Schwarz et al. 2018). Specifically, we use the official metric calculating tool *pc_error* provided by MPEG's GPCC. We set the PSNR peak value *r* = 1 following (Que, Lu, and Xu 2021) and (Fu et al. 2022). We normalize the point cloud data to $[0, 1]^3$. For a fair comparison, we correct other papers' results by eliminating inconsistencies in the PSNR formula.

Method	Bpp on SemanticKITTI			Time (s)
	D=8	D=10	D=12	En/Decode
VoxelContext-Net	-	0.951	4.497	0.426/0.419
OctAttention	**0.139**	**0.939**	3.740	**0.002**/2.060
Ours	0.146	0.963	**3.708**	0.005/**0.007**

Table 1: Comparison with prior state-of-the-art methods on SemanticKITTI dataset. 'D' is the max depth of octree. For each method, we choose the best compression results for fair comparison.

Experiment Results

Results on SemanticKITTI and ScanNet Datasets The rate-distortion curves of LiDAR compression are shown in Fig.5. Overall, one can observe that our method achieves better performance compared with other baselines, as expected. Specifically, compared with GPCC and OctSqueeze, we can achieve up to 54% and 21% bitrate savings on SemanticKITTI and 22% and 10% bitrate savings on ScanNet. The state-of-the-art method VoxelContext-Net is close to ours on ScanNet. The main reason is that the point cloud in ScanNet is much denser, which means there's much less information to discover in the octree. For sparse point cloud in SemanticKITTI, we achieve 17% Bpp savings compared with VoxelContext-Net. Considering that octree-based methods have the same reconstruction qualities under the same octree depth, we only visualize the quantitative results of ours and GPCC, shown in Fig.6. The visualization results also demonstrate the effectiveness of our method.

Comparison with Prior State-of-the-Art Methods As shown in Table.1, we further compare our method with state-of-the-art VoxelContext-Net (Que, Lu, and Xu 2021) and OctAttention (Fu et al. 2022). Compared with VoxelContext-Net, our method has up to 17% Bpp savings and achieves dozens of times performance improvement. The reason is that Voxelcontext-Net has a limited receptive field and expensive computation in higher resolution. Compared with OctAttention, we achieve about 300 *times* of time performance improvement, while maintaining similar compression effect. Although introducing sibling occupancy as the feature can greatly improve the compression performance, it will bring unacceptable time consumption because decoding the current octree node requires previous sibling nodes to be decoded. Our method eliminates the decoding dependencies between the nodes and shares the result of a MSA operation within the context window. Therefore, our decoding process is identical to the encoding process, which is hundreds of times faster than OctAttention.

Ablation Study and Analysis

Context Window Size As shown in Table.2, we perform ablation study on context window size. We set the different

Size N	Bpp on ScanNet				
	D=5	D=6	D=7	D=8	D=9
32	0.062	0.238	1.049	3.353	6.416
128	0.057	0.220	0.986	3.192	6.135
256	0.057	0.219	0.985	3.183	6.113
512	0.057	0.218	0.980	3.169	6.086
1024	0.056	0.211	0.958	3.123	6.021

Table 2: Performance of our methods when setting different context window sizes N.

APE	OctLeFF	OctPEG	Bpp	Params
✗	✗	✗	6.212	5.66 M
✓	✗	✗	6.213	5.66 M
✗	✓	✗	6.144	4.28 M
✗	✗	✓	6.161	5.66 M
✗	✓	✓	6.135	4.28 M

Table 3: Different compression performance when using different modules on ScanNet dataset with setting the depth to 9. ✗ denotes removing and ✓ denotes retaining.

context window sizes and train different models on ScanNet. From the table, we can observe that the method achieves up to 0.395 Bpp savings when setting the depth of the octree to 9. The model with larger context window size obtains better results, for larger window size has more context features. The experiment results demonstrate the effectiveness of large receptive field contexts.

Effectiveness of Proposed Modules As shown in Table.3, we perform ablation study on our proposed OctLeFF and OctPEG module. In the experiments, both the hidden size and channel size are set to 256. One can find that the original APE has little effect for the model, as we expected. The comparison shows that the OctLeFF and OctPEG modules effectively improve compression performance and parameters reduce 1.38M.

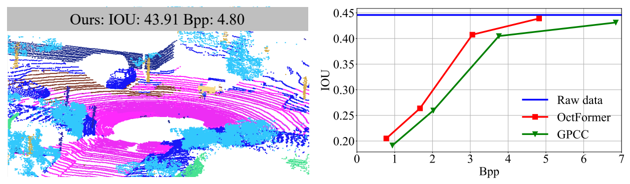
Runtime Analysis As shown in Table.4, we compare the runtime performance of different methods under different parameter settings. We can observe that both the encoding and decoding time in VoxelContext-Net increase with the voxel size, due to the more computationally expensive 3D convolution on voxel grids. But for transformer-based methods, when the context window size is larger, the efficiency of parallelism in the transformer is higher. OctAttention decoding one octree node needs the previous sibling nodes to be decoded, while our method can achieve much faster decoding since our encoding and decoding processes are the same for feature extraction and parallel computing. Experimental results show that our method achieves a significant improvement in runtime performance. We can observe that the method can be applied to efficient point cloud encoding and decoding.

Performance on Segmentation Application

The effect of the compression method on downstream tasks is another important metric. As shown in Fig.7, we evaluate the effectiveness of our proposed method on semantic segmentation, which is a basic task for point cloud (Nguyen and Le 2013; Hu et al. 2022). We use RandLA-Net (Hu et al. 2020) as the evaluative semantic segmentation method and intersection-over-union (IOU) as the metric. Experimental results show that our method can achieve segmentation performance close to the raw data when the Bpp is 4.8. At any given Bpp, we outperform GPCC and obtain higher IOU. Overall, experimental results demonstrate the effectiveness of our method on such downstream tasks (*i.e.*, semantic segmentation).

Methods	Time (s)	
	Encode	Decode
VoxelContext-Net (V=5)	0.3889	0.3741
VoxelContext-Net (V=7)	0.4010	0.4148
VoxelContext-Net (V=9)	0.4265	0.4199
VoxelContext-Net (V=11)	0.4523	0.4495
OctAttention (N=8)	0.1672	1.2710
OctAttention (N=32)	0.0476	1.3635
OctAttention (N=128)	0.0114	1.3737
OctAttention (N=512)	0.0030	1.4091
OctAttention (N=1024)	**0.0023**	2.0602
Ours (N=8)	0.4457	0.4561
Ours (N=32)	0.0765	0.0866
Ours (N=128)	0.0283	0.0369
Ours (N=512)	0.0096	0.0107
Ours (N=1024)	0.0052	**0.0073**

Table 4: The comparison of time performance of different methods. 'V' is the voxel size for VoxelContext-Net and 'N' represents the context window size for both our method and OctAttention.



(a) Segmentation visualization. (b) Comparison with others.

Figure 7: Quality performance of semantic segmentation of different point cloud compression methods on SemanticKITTI dataset.

Conclusion

We propose an octree-based transformer with local enhancement called OctFormer, which can compress point clouds efficiently. Specifically, we use non-overlapped context windows to construct sequences and share the results of the multi-head self-attention operation to reduce time overhead. We further design OctLeFF and OctPEG models to improve the compression performance. The experimental results on both datasets and the segmentation algorithm verify the effectiveness of the proposed method.

Acknowledgments

This work was supported by the Shenzhen Basic Research (JCYJ20180508152434975), the National Key Research and Development Program of China (2018YFB1802405), the Science and Technology Planning Project of Guangzhou (202007050004).

References

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9297–9307.

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*, 29: 2203–2216.

Chu, X.; Tian, Z.; Zhang, B.; Wang, X.; Wei, X.; Xia, H.; and Shen, C. 2021. Conditional positional encodings for vision transformers. *arXiv preprint arXiv:2102.10882*.

Dai, A.; Chang, A. X.; Savva, M.; Halber, M.; Funkhouser, T.; and Nießner, M. 2017. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5828–5839.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

Dricot, A.; Pereira, F.; and Ascenso, J. 2018. Rate-Distortion Driven Adaptive Partitioning for Octree-Based Point Cloud Geometry Coding. In *2018 25th IEEE International Conference on Image Processing*, 2969–2973.

Feng, Y.; Liu, S.; and Zhu, Y. 2020. Real-Time Spatio-Temporal LiDAR Point Cloud Compression. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 10766–10773.

Fu, C.; Li, G.; Song, R.; Gao, W.; and Liu, S. 2022. OctAttention: Octree-Based Large-Scale Contexts Model for Point Cloud Compression. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Garcia, D. C.; Fonseca, T. A.; Ferreira, R. U.; and de Queiroz, R. L. 2020. Geometry Coding for Dynamic Voxelized Point Clouds Using Octrees and Multiple Contexts. *IEEE Transactions on Image Processing*, 29: 313–322.

Google. 2017. Draco 3d data compresison. https://github.com/google/draco. Accessed: 2022-05-21.

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

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*, 1–6.

Houshiar, H.; and Nüchter, A. 2015. 3D point cloud compression using conventional image compression for efficient data transmission. In *2015 XXV International Conference on Information, Communication and Automation Technologies*, 1–8.

Hu, Q.; Yang, B.; Fang, G.; Guo, Y.; Leonardis, A.; Trigoni, N.; and Markham, A. 2022. Sqn: Weakly-supervised semantic segmentation of large-scale 3d point clouds. In *European Conference on Computer Vision*, 600–619. Springer.

Hu, Q.; Yang, B.; Xie, L.; Rosa, S.; Guo, Y.; Wang, Z.; Trigoni, N.; and Markham, A. 2020. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11108–11117.

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*, 1313–1323.

Huang, Y.; Peng, J.; Kuo, C.-C. J.; and Gopi, M. 2006. Octree-Based Progressive Geometry Coding of Point Clouds. In *PBG@ SIGGRAPH*, 103–110.

Kaya, E. C.; Schwarz, S.; and Tabus, I. 2021. Refining The Bounding Volumes for Lossless Compression of Voxelized Point Clouds Geometry. In *2021 IEEE International Conference on Image Processing*, 3408–3412.

Li, Y.; and Ibanez-Guzman, J. 2020. Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems. *IEEE Signal Processing Magazine*, 37(4): 50–61.

Li, Y.; Zhang, K.; Cao, J.; Timofte, R.; and Van Gool, L. 2021. Localvit: Bringing locality to vision transformers. *arXiv preprint arXiv:2104.05707*.

Limuti, S.; Polo, E.; and Milani, S. 2018. A Transform Coding Strategy for Voxelized Dynamic Point Clouds. In *2018 25th IEEE International Conference on Image Processing*, 2954–2958.

Liu, Z.; Tang, H.; Lin, Y.; and Han, S. 2019. Point-Voxel CNN for Efficient 3D Deep Learning. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Nguyen, A.; and Le, B. 2013. 3D point cloud segmentation: A survey. In *2013 6th IEEE conference on robotics, automation and mechatronics*, 225–230. IEEE.

Nguyen, D. T.; Quach, M.; Valenzise, G.; and Duhamel, P. 2021. Lossless coding of point cloud geometry using a deep generative model. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(12): 4617–4629.

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*, 4320–4324.

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*, 6042–6051.

Schwarz, S.; Preda, M.; Baroncini, V.; Budagavi, M.; Cesar, P.; Chou, P. A.; Cohen, R. A.; Krivokuća, M.; Lasserre, S.; Li, Z.; et al. 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. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3): 379–423.

Sun, X.; Ma, H.; Sun, Y.; and Liu, M. 2019. A Novel Point Cloud Compression Algorithm Based on Clustering. *IEEE Robotics and Automation Letters*, 4(2): 2132–2139.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

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*, 31(12): 4909–4923.

Wen, X.; Wang, X.; Hou, J.; Ma, L.; Zhou, Y.; and Jiang, J. 2020. Lossy Geometry Compression Of 3d Point Cloud Data Via An Adaptive Octree-Guided Network. In *2020 IEEE International Conference on Multimedia and Expo*, 1–6.

Wu, H.; Xiao, B.; Codella, N.; Liu, M.; Dai, X.; Yuan, L.; and Zhang, L. 2021. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 22–31.

Zou, Q.; Sun, Q.; Chen, L.; Nie, B.; and Li, Q. 2022. A Comparative Analysis of LiDAR SLAM-Based Indoor Navigation for Autonomous Vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 23(7): 6907–6921.