ECM-OPCC: EFFICIENT CONTEXT MODEL FOR OCTREE-BASED POINT CLOUD COMPRESSION

Yiqi Jin$^{1,2}$, Ziyu Zhu$^3$, Tongda Xu$^1$, Yuhuan Lin$^{1,2}$, Yan Wang$^1$

$^1$Institute for AI Industry Research (AIR), Tsinghua University
$^2$Department of Automation, Tsinghua University
$^3$Department of Computer Science, Tsinghua University

ABSTRACT

Recently, deep learning methods have shown promising results in point cloud compression. However, previous octree-based approaches either lack sufficient context or have high decoding complexity (e.g. > 900s). To address this problem, we propose a sufficient yet efficient context model and design an efficient deep learning codec for point clouds. Specifically, we first propose a segment-constrained multi-group coding strategy to exploit the autoregressive context while maintaining decoding efficiency. Then, we propose a dual transformer architecture to utilize the dependency of current node on its ancestors and siblings. We also propose a random-masking pre-train method to enhance our model. Experimental results show that our approach achieves state-of-the-art performance for both lossy and lossless point cloud compression, and saves a significant amount of decoding time compared with previous octree-based SOTA compression methods.

Index Terms— Point Cloud Compression, Efficient Context Model

1. INTRODUCTION

Inspired by learned image compression and traditional point cloud codec, deep learning community invents tree-based, voxel-based, projection-based and point-based Deep Point Cloud Compression (DPCC): Tree-based approaches [1, 2, 3] represent point cloud by octree and aggregate ancestor and sibling information to predict current node occupancy; Voxel-based approaches [4, 5, 6] quantize point cloud into voxel grid and use 3D convolution as learned transform; Projection-based approaches [7, 8] project 3D point cloud to 2D image and use image compression techniques to compress it; And point-based approaches [9, 10] utilize modules from point cloud understanding to transform point cloud and compress transformed points and features. Among above-mentioned methods, octree-based approach is commonly used for large scale sparse point cloud.

A major issue of octree-based methods is the decoding complexity due to the autoregressive context model, which hinders the practical deployment of such methods. For example, in [3], both ancestor and sibling information are introduced as autoregressive context models, which greatly increases the decoding complexity. On the other hand, advancements in parallelizable autoregressive model in density estimation [11] and image compression [12, 13] provide valuable insight for efficient context modeling in deep data compression. Inspired by this, we propose to solve DPCC problem by multi-group coding strategy. Specifically, we factorize the fully autoregressive context of octree into layer-wise autoregressive context, and slice the nodes within the same layers into parallel-decodable context segment. Within each context segment, we deploy group-wise autoregressive context. We propose a dual transformer architecture to process branches containing ancestor and sibling information. Besides, we introduce random masking pre-training strategy to boost the performance. Experimental results show our method outperforms previous deep point cloud compression methods re-
garding compression ratio and obtains very competitive de-
coding speed (Fig. 1).

Our main contributions are as follows:

- We propose a new segment-constrained multi-group coding strategy that enables parallel decoding of nodes inside each group, accelerating decoding process while maintaining compression performance.

- We propose a novel dual transformer architecture with level-parallel and group-parallel branch to better extract context information from ancestors and siblings.

- We introduce randomly masking input occupancy code as an efficient pre-training strategy to boost performance of our context model.

- Our proposed model achieves SOTA bitrate for lossless compression and R-D (Rate-Distortion) performance for lossy compression. Moreover, it saves significant decoding time compared with previous octree-based methods.

2. METHODOLOGY

2.1. Background: Octree-based Compression

Octree is a data structure describing 3D space, and the occu-
pancy code of each octree node is a binary flag indicating spatial occupancy of the eight cubic children. Denote the occu-
pancy code sequence of the whole octree as \( X \), we have

\[
X = \{x^1, ..., x^l\}, x^i = \{x^i_1, ..., x^i_{N^i}\} \tag{1}
\]

where \( l \) is the total level of octree, \( x^i \) is the occupancy code of level \( i \), \( N^i \) is the length of node in layer \( i \), \( x^i_k \) is the \( k^{th} \) node in \( i^{th} \) layer. Octree-based DPCV aims to losslessly compress \( X \) using a model parameterized by \( \theta \) to construct a parametric probability distribution \( P_\theta(X) \) approximating the true density \( P(X) \). The optimization for bitrate is Eq. 2.

\[
\theta^* \leftarrow \arg\min_\theta \mathbb{E}_{P(X)}[−\log P_\theta(X)] \tag{2}
\]

Due to the extremely high dimension of \( X \), the key to octree compression is to construct a tractable factorization to \( P_\theta(X) \) by utilizing the domain knowledge of dependency, such as the layer-wise autoregressive method:

\[
P_\theta(X) = P_\theta(x^1) \prod_{i=2}^l P_\theta(x^i|x^{<i}) \tag{3}
\]

Furthermore, we can use auxiliary information such as node level and octant to aid the prediction of occupancy codes. The auxiliary information in current level \( i \) can be obtained from the occupancy code of previous level \( i - 1 \) by \( y^i = f(x^{i-1}) \). And Eq. 3 can be optimized as:

\[
P_\theta(X) = P_\theta(x^1) \prod_{i=2}^l P_\theta(x^i|x^{<i}, y^{<i}) \tag{4}
\]

Fig. 2. The decoding procedure of 4-group context segment. Nodes within same group (marked with same ID) is decoded in parallel using nodes from previous groups as context.

Eq. 4 is a layer-wise factorization method allowing parallel decoding of nodes in each layer. To fully exploit the context of ancestors and siblings, one can also adopt node-wise autoregression inside each layer \( x^i \) to obtain a fully autoregres-
sive probabilistic model as Eq. 5.

\[
P_\theta(x^i|x^{<i}, y^{<i}) = \prod_{k=2}^{N^i} P_\theta(x^i_k|x^{<k}, y^{<k}) \tag{5}
\]

In fact, the factorization described by Eq. 5 corresponds to a more general formulation for OctAttention [3], which achieves great compression bitrate due to its thorough capture of context information. However, the decoding of such model is fully sequential. In other words, the decoding time is proportional to \( \Theta(N^i) \), which can be extremely slow (e.g., \( >900\) s per point cloud).

2.2. Multi-Group Coding Strategy

We propose a multi-group coding strategy to support large- scale context while greatly improve coding efficiency over fully autoregressive model. Specifically, on the basis of Eq. 4, we divide the nodes in each layer into parallel coding context segments. Then we group the nodes in each context segment and autoregressively code these groups. The nodes in each group are coded in parallel, and the occupancy codes of previous groups serve as context for latter groups (Fig. 2.2).

The previous layer context (ancestor) and same layer con-
text (siblings) are modeled by our dual transformer architec-
ture to estimate the likelihood for entropy coding. The decod-
ing process between each context segment and within each group is parallelizable, which greatly reduces decoding complexity. On the other hand, sufficient autoregressive modeling is maintained to promise compression performance.

Formally, denote the \( j^{th} \) segment of nodes in layer \( i \) as \( x^i_{wij} = \{x^i_{wij1}, ..., x^i_{wijg}\} \), where \( g \) is the number of group in \( x^i_{wij1}, x^i_{wijg} \) is the \( g^{th} \) group nodes of \( j^{th} \) segment in \( i^{th} \) layer, and we have the following factorization of layer-wise density:
\[ P_\theta(x_i|x^{<i}, y^{\leq i}) = \prod_{j=1}^{h^i} P_\theta(x_{w_j}^{<i}, y^{<i}) \]  

\[ P_\theta(x_{w_j}^{<i}, y^{<i}) = P_\theta(x_{w_1}^{<i}, \ldots) \]  

where \( h^i \) is the number of segment in \( i \)th layer. Taking Eq. 6 and Eq. 7 back to Eq. 4, we obtain the full likelihood model \( P_\theta(X) \). Our multi-group coding strategy impacts compression performance in terms of decoding time and compression bitrate. As for the former, assuming fully parallelizable hardware, the decoding complexity for a fully autoregressive model (Eq. 5) is \( \Theta(\sum_i N^i) \). In our strategy, with autoregressive dependency between layers and parallelizable decoding for context segments and nodes within each group, the complexity reduces to \( \Theta(l \cdot g) \). Empirically, as shown in Tab. 1, our decoding time is significantly reduced.

For bitrate, we can derive the theoretical bitrate as \( \mathbb{E}_{P(X)}[-\log P(X)] + D_{KL}(P(X)||P_\theta(X)) \). And by applying the conclusion in [14], we have the theoretical lowerbound on the bitrate for fully autoregressive model and our strategy as \( \mathbb{E}[-\log P(X)] \) and \( \mathbb{E}[-\log P(X)] + \sum \mathbb{H}(x_{w_j}^{<i}) - \mathbb{H}(X) \), respectively\(^1\). Although the theoretical bitrate lower-bound of our method is higher, in practice we can achieve better bitrate by smartly design the model and optimization.

Fig. 3. Dual transformer structure or our model. In practice we process one context segment at a time.

2.3. Dual Transformer Structure

We design a dual branch transformer structure to support our coding strategy, as illustrated in Fig. 3. Despite that our approach is fully autoregressive in layer level in nature, in practice we trace back to at most \( m - 1 \) ancestors to save calculation (pad zero if not enough), obtaining \( n \times m \) sibling and ancestor nodes in total. We use occupancy code \( z \), level and octant \( y \) as context to construct a \( n \times m \times 3 \) tensor as input.

The level-parallel branch is designed to fully use all available information from previous layers, corresponding to Eq. 4. The occupancy codes of current encoding/decoding level are masked with 0 to prevent information leakage. The group-parallel branch is designed to exploit context information carried by occupancy code of sibling nodes. We use a multi-group mask matrix to modulate the attention layer, which ensures that only already decoded occupancy code from siblings can be used as context for current node, corresponding to Eq. 7.

For encoding, we just need one forward pass of the two branches to compress all the nodes. For decoding, to decompress the node of each level, we need to run level-parallel branch once, and run group-parallel branch once, and the total sequential-calculation complexity is \( \Theta(l \cdot g) \).

2.4. Random Masking Pretrain

The essential task of our transformer is to use context \( x^{<i}, y^{<i}, x_{w_j}^{<k} \) to predict current group \( x_{w_j}^{i} \). Therefore, it is important to fully exploit the predict ability of our transformer-based model. Inspired by BERT [15], we design a pre-train method to help our model develop the ability of predicting current node with context information. Specifically, we randomly mask the input occupancy code with 50% probability, and use our model (without multi-group mask) to predict the masked positions with unmasked occupancy codes and full information of ancestors. This operation simulates the process of actual coding. As for gradient backward, we only backward the loss between predicted and ground truth occupancy codes of masked positions.

3. EXPERIMENT

3.1. Datasets

We train and test our method on LiDAR and object dataset for comprehensive evaluation. For LiDAR dataset, we adopt SemanticKITTI [16]. For Object dataset, we adopt 8i Voxelized Full Bodies (MPEG 8i) [17] and Microsoft Voxelized Upper Bodies (MVUB) [18]. To ensure fair comparison, we use the same train-test split as previous works [3, 2], and perform lossy compression task on LiDAR dataset, lossless compression task on object datasets.

3.2. Implementation Details

All experiments are performed on a machine with NVIDIA A100-PCIE-40GB GPU. We also test our method and OctAttention on the machine with one NVIDIA GeForce RTX 3090 for fair comparison with results reported in SparsePCGC [6]. To balance compression bitrate and decoding time, we set \( n = 1024, g = 8 \) for LiDAR model and \( n = 2048, g = 8 \) for object model as default settings.
Table 1. Bpp and coding time results on MPEG 8i dataset compared with G-PCC, voxel-based and octree-based methods.

| Point Cloud                  | Traditional G-PCC | SparsePCGC | VoxelIDNN | MSVoxelIDNN | OctAttention | Ours |
|------------------------------|-------------------|------------|-----------|-------------|--------------|------|
| loot_vox10 (bpp)             | 0.95              | 0.63       | 0.58      | 0.73        | 0.62         | 0.55 |
| redandblack_vox10 (bpp)      | 1.09              | 0.72       | 0.66      | 0.87        | 0.73         | 0.66 |
| boxer_viewdep_vox10 (bpp)    | 0.94              | 0.60       | 0.55      | 0.70        | 0.59         | 0.51 |
| Thaidancer_viewdep_vox10 (bpp) | 0.99         | 0.67       | 0.68      | 0.85        | 0.65         | 0.58 |
| Average bpp                  | 0.99              | 0.66       | 0.62      | 0.79        | 0.65         | 0.58 |
| Average Gain over G-PCC      |                   |            | 33.8%     | 37.6%       | 20.5%        | 34.6%|
| Average Encoding Time (s)    | 4.0               | 9.5        | 885       | 54          | 0.80         | 1.92 |
| Average Decoding Time (s)    | 1.0               | 9.1        | 640       | 58          | 948          | 19.5 |

Fig. 4. Results of different methods on SemanticKITTI.

We compare our work with octree-based compression method OctAttention [3], VoxelContext-Net [2] OctSqueeze [1], competitive voxel-based learned method SparsePCGC [6], and traditional hand-crafted G-PCC from MPEG standard in the stable version (TMC13 v14.0) [19]. In object point cloud compression, we compare our method with G-PCC, OctAttention, SparsePCGC [6], VoxelIDNN [4], MSVoxelIDNN [5]. We apply lossless compression on above methods for fair comparison on bpp and coding time.

3.3. Experimental Results

3.3.1. Lossy Compression Performance

The bpp-distortion curves of LiDAR point cloud compression is shown in Fig. 4. Our model outperforms other methods at all bitrates consistently. Specifically, we save 31% bitrate over G-PCC averagely over five distortion levels. Experimental results verify the effectiveness of our multi-group context model over previous methods.

3.3.2. Lossless Compression Performance

The lossless compression bitrate and coding time on object datasets are shown in Table 1, our method saves 42.4% and 33.6% bpp averagely on MPEG 8i and MVUB over traditional method G-PCC while maintaining a very fast decoding time. Fig. 5 shows that our method w/ pre-train outperforms SOTA octree-based method OctAttention by 6.3% and 1.3% on two datasets, and gets considerable gains over models w/o pre-train.

3.3.3. Qualitative Results

Fig. 5 shows the visualized distortion of our method and G-PCC at similar compression rate. Our method outperforms G-PCC by more than 10 dB on D1 PSNR.

4. CONCLUSION

In this work, We propose an efficient large-scale context entropy model for point cloud geometry compression. To be specific, we propose a multi-group coding strategy to encode and decode the octree efficiently, based on which we propose a dual transformer architecture. We also design a random masking pre-train strategy. Results show that our model achieves SOTA compression performance and significantly reduces decoding time, which makes the practical deployment of octree-based DPCC possible.
5. REFERENCES

[1] 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 (CVPR)*, June 2020.

[2] Zizheng Que, Guo Lu, and Dong Xu, “Voxelcontextnet: An octree based framework for point cloud compression,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6042–6051.

[3] Chunyang Fu, Ge Li, Rui Song, Wei Gao, and Shan Liu, “Octattention: Octree-based large-scale contexts model for point cloud compression,” *arXiv preprint arXiv:2202.06028*, 2022.

[4] Dat Thanh Nguyen, Maurice Quach, Giuseppe Valenzise, and Pierre Duhamel, “Learning-based lossless compression of 3d point cloud geometry,” in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 4220–4224.

[5] Dat Thanh Nguyen, Maurice Quach, Giuseppe Valenzise, and Pierre Duhamel, “Multiscale deep context modeling for lossless point cloud geometry compression,” in *2021 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*. IEEE, 2021, pp. 1–6.

[6] Jianqiang Wang, Dandan Ding, Zhu Li, Xiaoxing Feng, Chuntong Cao, and Zhan Ma, “Sparse tensor-based multiscale representation for point cloud geometry compression,” *arXiv preprint arXiv:2111.10633*, 2021.

[7] Xuanyu Zhou, Charles R. Qi, Yin Zhou, and Dragomir Anguelov, “Riddle: Lidar data compression with range image deep delta encoding,” 2022.

[8] Sukai Wang and Ming Liu, “Point cloud compression with range image-based entropy model for autonomous driving,” in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII*. Springer, 2022, pp. 323–340.

[9] 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*, 2022, pp. 2333–2342.

[10] Zujie Liang and Fan Liang, “Transpcc: Towards deep point cloud compression via transformers,” 2022.

[11] Scott Reed, Aaroh van den Oord, Nal Kalchbrenner, Sergio Gómez Colmenarejo, Ziyu Wang, Yutian Chen, Dan Belov, and Nando de Freitas, “Parallel multiscale autoregressive density estimation,” in *Proceedings of the 34th International Conference on Machine Learning*, Doina Precup and Yee Whye Teh, Eds. 06–11 Aug 2017, vol. 70 of *Proceedings of Machine Learning Research*, pp. 2912–2921, PMLR.

[12] 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*, 2021, pp. 14771–14780.

[13] Fangzheng Lin, Heming Sun, Jinming Liu, and Jiro Katto, “Multistage spatial context models for learned image compression,” in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[14] Fei Huang, Tianhua Tao, Hao Zhou, Lei Li, and Minlie Huang, “On the learning of non-autoregressive transformers,” in *Proceedings of the 39th International Conference on Machine Learning*, Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, Eds. 17–23 Jul 2022, vol. 162 of *Proceedings of Machine Learning Research*, pp. 9356–9376, PMLR.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 2018.

[16] Jens Behley, Martin Garbade, Andres Milioito, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Juergen Gall, “Semantickitti: A dataset for semantic scene understanding of lidar sequences,” *IEEE/CVF International Conference on Computer Vision (ICCV)*, p. 9296–9306, 2019.

[17] Ryan Killea, Yun Li, Saeed Bastani, and Paul McLachlan, “8i voxelized full bodies - a voxelized point cloud dataset,” *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006*, 2017.

[18] Loop Charles, Cai Qin, Orts Escolano Sergio, and A. Chou Philip, “Microsoft voxelized upper bodies - a voxelized point cloud dataset,” *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document m38673/M7201*, 2016.

[19] MPEG, “https://github.com/MPEGGroup/mpeg-ppc-voc-v Tellsme123, 2021.”