Supplementary Material of
Efficient Hierarchical Entropy Model for Learned Point Cloud Compression

1. Implementation Details

1.1. Detailed Context Structure

The ancestral feature is defined as occupancy symbols, level indices, octant indices, and parent coordinates collected from node itself and its $K = 3$ ancestors. In practice, only parent coordinates of the currently coded node are introduced to simplify the context:

$$\mathbf{a}_i = \left\{\mathbf{h}_i^{\varnothing}, \mathbf{h}_{\text{anc}(i)}, \ldots, \mathbf{h}_{\text{anc}(\ldots\text{anc}(i))}, \mathbf{b}_i\right\}, \tag{1}$$

where $\mathbf{h}_i$ are 3-dimensional features including occupancy symbol, level index, and octant index. $\mathbf{b}_i$ are bounding box coordinates of the currently coded node. Therefore, $\mathbf{a}_i$ has 14 dimensions ($3 \times (K+1) + 3 - 1$) because $\mathbf{h}_i^{\varnothing}$ excludes the occupancy symbol x_i. We stack ancestral features of N nodes within the same context window to generate the ancestral context $\mathbf{A}_i$ as:

$$\mathbf{A}_i = \{\mathbf{a}_{i-N+1}, \ldots, \mathbf{a}_i\}. \tag{2}$$

1.2. Geometry-aware Feature Extractor

We adopt an octree-based geometry-aware feature extracting scheme to discover geometric features within each local window. Its overall architecture is shown in Fig. 1. It uses a DGCNN [1] to explore geometric features from parent node coordinates of the currently coded node. Specifically, it performs edge convolution on a dynamic graph, where each node connects to its $k = 20$ nearest neighbors in the feature space. Additionally, octree features are embedded via MLPs and combined with geometric features. The dynamic graph is updated using the aggregated features to take both geometric features and node embeddings into consideration.

1.3. Hierarchical Attention

The specific structure of the hierarchical self-attention model is shown in Tab. 1. In our implementation, the context window size N and local window length L are set to 8192 and 512, respectively. Therefore, the model is composed of $\log_2 \frac{N}{L} + 1 = 5$ blocks that include $4, 4, 4, 4, 2$ localized attention layers, respectively. In each block, T tokens are partitioned into $W = \frac{T}{L}$ local windows. Features

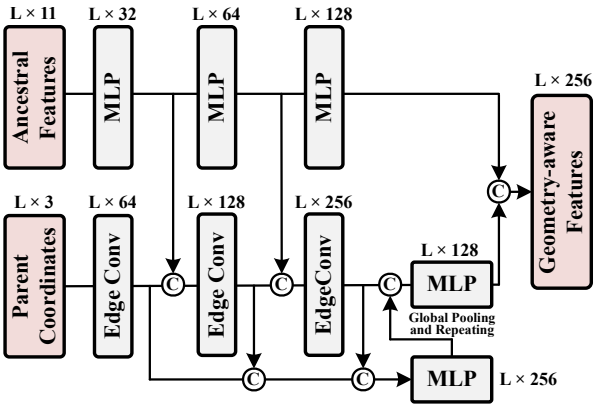

Figure 1. Architecture of the geometry-aware feature extractor.

from $2^{t-1} \times L$ nodes are incorporated into $L = 512$ tokens in block t. In the last block, N nodes are represented with L tokens, which are computed within the same local window. It hence reaches the global receptive field.

The architecture of the hierarchical cross-attention model is detailed in Tab. 2. In practice, we use a doubled local window length $L = 1024$ for cross-attention. Each local window computes dependencies between key and query features with the size of $\frac{L}{2} = 512$. Complexities for self-attention and cross-attention are:

$$\begin{aligned}
\Omega(\text{Self-attetnion}) &= 2 \times l \times l \times \frac{N}{l} \times C = 2lNC, \\
\Omega(\text{Cross-attention}) &= 2 \times \frac{L}{2} \times \frac{L}{2} \times \frac{N}{L} \times C = \frac{L}{2}NC.
\end{aligned} \tag{3}$$

Here, l and L are local window sizes for self-attention and cross-attention, respectively. Therefore, cross-attention with a doubled window length $L = 2l$ still has lower complexity compared with self-attention. The cross-attention model includes 4 blocks with $2, 2, 1, 1$ localized attention layers. We shrink its capacity to improve the efficiency, but it is still capable of preserving satisfactory performance.

Table 1. Detailed architecture of the hierarchical self-attention model. T and W indicate the number of tokens and local windows in the corresponding block. L is the local window size fixed to 512. C and H represent channel dimensions and head numbers, respectively. MSA denotes the multi-head self-attention module.

	Block 1, T8192,W16,L512	**Block 2, T4096,W8,L512**	**Block 3, T2048,W4,L512**	**Block 4, T1024,W2,L512**	**Block 5, T512,W1,L512**
Layer 1	MSA, C256, H4 MLP, C256 Window Shifting	MSA, C256, H4 MLP, C256 Window Shifting	MSA, C256, H4 MLP, C256 Window Shifting	MSA, C256, H4 MLP, C256 Window Shifting	MSA, C256, H4 MLP, C256
Layer 2	MSA, C256, H4 MLP, C256 Reverse Shifting	MSA, C256, H4 MLP, C256 Reverse Shifting	MSA, C256, H4 MLP, C256 Reverse Shifting	MSA, C256, H4 MLP, C256 Reverse Shifting	MSA, C256, H4 MLP, C256
Layer 3	MSA, C256, H4 MLP, C256 Window Shifting	MSA, C256, H4 MLP, C256 Window Shifting	MSA, C256, H4 MLP, C256 Window Shifting	MSA, C256, H4 MLP, C256 Window Shifting	
Layer 4	MSA, C256, H4 MLP, C256 Reverse Shifting	MSA, C256, H4 MLP, C256 Reverse Shifting	MSA, C256, H4 MLP, C256 Reverse Shifting	MSA, C256, H4 MLP, C256 Reverse Shifting	
Node Merging	downsample 2×, T4096, W8, C256	downsample 2×, T2048, W4, C256	downsample 2×, T1024, W2, C256	downsample 2×, T512, W1, C256	

Table 2. Detailed architecture of the hierarchical cross-attention model. MCA is the multi-head cross-attention module.

	Block 1, T8192,W8,L1024	**Block 2, T4096,W4,L1024**	**Block 3, T2048,W2,L1024**	**Block 4, T1024,W1,L1024**
Layer 1	MCA, C256, H4 MLP, C256 Window Shifting	MCA, C256, H4 MLP, C256 Window Shifting	MCA, C256, H4 MLP, C256	MCA, C256, H4 MLP, C256
Layer 2	MCA, C256, H4 MLP, C256 Reverse Shifting	MCA, C256, H4 MLP, C256 Reverse Shifting		
Node Merging	downsample 2×, T4096, W4, C256	downsample 2×, T2048, W2, C256	downsample 2×, T1024, W1, C256	

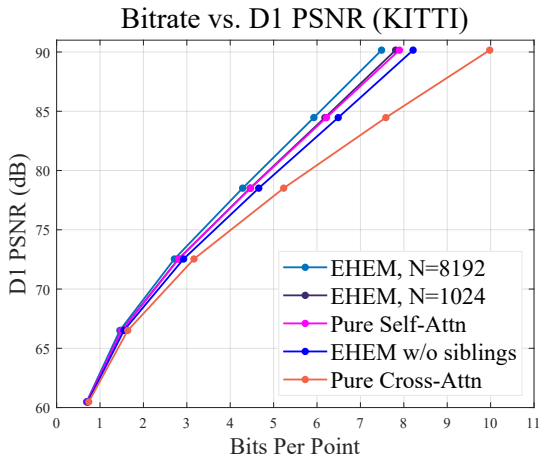

Figure 2. Ablation studies on sibling prior and attention model structure.

Table 3. Bpp for lossless compression on the 8iVFB dataset.

Method	G-PCC	OctAttention	EHEM
Loot10	0.95	0.62	0.58
Redandblack10	1.09	0.73	0.69
Boxer10	0.94	0.59	0.55
Thaidancer10	0.99	0.65	0.62
Average	0.99	0.65	0.61

2. Additional Experiments

2.1. Additional Ablation Studies

Sibling Prior Although exploiting sibling priors causes a two-step coding process, it is effective to boost the compression performance. We build an EHEM variant which predicts $\mathbf{x}_{i_2}$ based on ancestral context $\mathbf{A}_i$ without introducing sibling priors $\mathbf{x}_{i_1}$. As shown in Fig. 2, sibling priors bring considerable performance improvements.

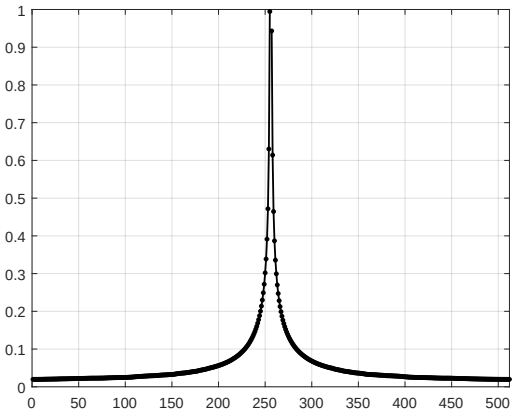

Figure 3. Normalized dependency intensity of neighbors at different positions.

Attention Model Structure We build two EHEM variants to validate the effects of self-attention and cross-attention. In Pure Self-Attn, a hierarchical self-attention model replaces the cross-attention one to compute sibling features. In Pure Cross-Attn, $\tilde{p}(\mathbf{x}_{i_1})$ is directly predicted based on $\mathbf{F}^a_{i_1}$ without performing hierarchical self-attention, then $\mathbf{F}^a_i$ and $\mathbf{x}_{i_1}$ are passed to the hierarchical cross-attention branch to estimate $\tilde{p}(\mathbf{x}_{i_2})$. These two variants are implemented with $N = 1024$. In Fig. 2, the comparison between EHEM (N=1024) and Pure Self-Attn shows that cross-attention works slightly better than self-attention in aggregating sibling features. Meanwhile, the gap between EHEM (N=1024) and Pure Cross-Attn suggests that it is necessary to perform hierarchical self-attention to enhance ancestral features.

2.2. Dense Point Clouds

Results on dense point cloud dataset 8iVFB are shown in Tab. 3. EHEM still outperforms OctAttention and G-PCC at dense point clouds.

2.3. Additional Complexity Analysis

We further test the runtimes of EHEM using a TITAN Xp GPU. Results in Tab. 4 shows that EHEM is still practical on this less powerful GPU.

2.4. Quantitative Dependencies

In the main paper (Fig.4), we provide a visualization of ancestral dependencies among the first 16 nodes in the local window. Here, in Fig. 3, we report quantitative normalized dependency scores across the whole local window with $L = 512$. The attention score matrix has been shifted to make all currently coded nodes located at the index of 256. It proves that closer nodes are more informative.

Table 4. Encoding/decoding times (in seconds) on SemanticKITTI with a TITAN Xp GPU.

Method	D=12	D=14	D=16
EHEM	0.59 / 0.65	1.99 / 2.08	4.30 / 4.43
Light EHEM	0.42 / 0.46	1.26 / 1.43	2.67 / 3.11

References

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