Autoregressive and Residual Index Convolution Model for Point Cloud Geometry Compression

Gerald Baulig and Jiun-In Guo *

Posted Date: 19 January 2026

doi: 10.20944/preprints202601.1367.v1

Keywords: data compression; point cloud; LiDAR; autonomous driving; robotic; virtual reality

 doi:10.20944/preprints202601.1367.v1

Article

Autoregressive and Residual Index Convolution Model for Point Cloud Geometry Compression

Gerald Baulig [1,3] **and Jiun-In Guo** [1,2,3]

[1] College of Electrical & Computer Engineering
[2] Institute of Electronics Engineering
[3] National Yang Ming Chiao Tung University, Hsinchu, Taiwan R.O.C.
* Correspondence: jiguo@nycu.edu.tw

Abstract

This study introduces a hybrid point cloud compression method that transfers from octree-nodes to voxel occupancy estimation to find its lower-bound bitrate by using a Binary Arithmetic Range Coder. In previous attempts, we've shown that our entropy compression model based on index convolution achieves promising performance while maintaining low complexity. However, our previous model lacks an autoregressive approach, which is apparently indispensable to compete with the current state-of-the-art of compression performance. Therefore, we adapt an autoregressive grouping method that iteratively populates, explores, and estimates the occupancy of 1-bit voxel candidates in a more discrete fashion. Furthermore, we refactored our backbone architecture by adding a distiller layer on each convolution, forcing every hidden feature to contribute to the final output. Our proposed model extracts local features using lightweight 1D convolution applied in varied ordering and analyzes causal relationships by optimizing the cross-entropy. This approach efficiently replaces the voxel convolution techniques and attention models used in previous works, providing significant improvements in both time and memory consumption. The effectiveness of our model is demonstrated on three datasets, where it outperforms recent deep learning-based compression models in this field.

Keywords: data compression; point cloud; LiDAR; autonomous driving; robotic; virtual reality

1. Introduction

Point clouds have become a fundamental data representation across a wide range of applications, including autonomous driving, robotics and virtual reality [1–3]. Their ability to accurately capture complex three-dimensional structures makes them indispensable in scenarios where precise geometric detail is required. However, raw point clouds are often extremely large and irregular in structure, leading to substantial storage and transmission costs. As a result, Point Cloud Geometry Compression (PCGC) has emerged as a critical research topic, aiming to reduce data size while preserving spatial fidelity.

Traditional PCGC methods, such as Google's Draco [4] and MPEG's G-PCC [5], rely on handcrafted context models that operate on tree or graph-based spatial structures. These methods offer stable performance and reliable decoding characteristics, yet they are fundamentally limited by heuristic probability estimation. In particular, G-PCC achieves strong Rate-Distortion (R-D) but suffers from high decoding latency, whereas Draco favors real-time decoding at the expense of compression efficiency.

Advances in deep learning have enabled significant improvements in PCGC by learning context-aware occupancy prediction directly from data. Voxel-based models [6–9] leverage 3D convolution or masked autoregression to predict occupancy at fixed resolution grids, while octree-based approaches [10–13] operate on hierarchical spatial partitions. Hybrid techniques [14,15] have demonstrated strong performance by integrating voxel convolution and feature embeddings, switching between voxel and tree representation. Despite these advancements, existing neural approaches often

 doi:10.20944/preprints202601.1367.v1

face a fundamental trade-off between compression performance, runtime efficiency, and memory consumption, particularly during decoding.

In intermediate proceedings [16], we introduced *multi-index convolution* as a novel entropy model for PCGC. However, the resulting performance lagged behind state-of-the-art methods. This shortcoming was primarily due to three factors:

1. the absence of coherent autoregressive feedback from sibling nodes,
2. an inefficient backbone architecture, and
3. the inherently more challenging objective of predicting octree occupancy patterns rather than voxel occupancies.

With MIC-OPCCv2, we present a substantially improved version of our *multi-index convolution* approach, which surpasses current models that rely on self-attention or sparse convolution mechanisms. By reformulating the objective to predict voxel occupancy, our method facilitates an efficient *progressive grouping strategy* [9] that enables iterative neighborhood context exploration during decoding. Moreover, we introduce a novel *Blender & Distiller* network pattern, which intensifies the residual learning pattern inspired by ResNet [17] while adopting a hierarchical fusion mechanism similar in spirit to U-Net [18], but designed as a *voting-based* probability aggregation process. Finally, we address the limitations in diagonal spatial analysis by incorporating the *breadth-first traversal order* into our set of indices, thereby enhancing spatial feature comprehension and robustness. Our method introduces three key contributions:

1. **Extended Multi-Index Convolution:** An extended version of our proposed index convolution, including the *breadth-first traversal order* to approximate diagonal analysis along with sequences of axial index reorderings that expand coherent spatial receptive fields without the computational overhead of 3D sparse convolution or attention layers.
2. **Blender & Distiller Architecture:** A hierarchical feature aggregation and voting mechanism inspired by U-Net-like multi-scale representation learning, improving robustness to large entropy variation across octree levels.
3. **Progressive Grouping Strategy:** An adapted version of a structured decoding schedule that prioritizes contextual refinement for early occupancy decisions while accelerating later decoding stages.

Together, these components enable MIC-OPCCv2 to achieve competitive or state-of-the-art R-D across both sparse LiDAR scans and dense object reconstructions, while maintaining a substantially reduced computational footprint compared to transformer- and sparse-convolution-based models. Experimental results demonstrate that MIC-OPCCv2 generalizes effectively across heterogeneous point cloud distributions and achieves favorable performance in both R-D and decoding efficiency.

The remainder of this article is organized as follows. In Section 2, we review related work in deep learning-based point cloud compression. Section 3 outlines the theoretical background and entropy modeling framework. Section 4 presents our proposed MIC-OPCCv2 model in detail. In Section 5, we describe the experimental setup, including datasets, model configurations, and training strategies. Finally, Section 6 evaluates our method against state-of-the-art models, followed by discussions and conclusions.

2. Related Work

In PCGC, we distinguish between *sparse* and *dense* point clouds primarily reflecting differences in spatial distribution and structural completeness.

Sparse point clouds – typically acquired by mobile LiDAR systems [19–22] – contain relatively few points per unit volume, exhibiting large empty regions, anisotropic sampling patterns, and strong non-uniformity caused by occlusions and long-range sensing. They generally represent expansive, concave outdoor environments in which surfaces are only partially and unevenly observed.

In contrast, dense point clouds – such as voxelized human-body scans or multi-view reconstructions [23,24] – provide near-contiguous sampling of predominantly convex shapes, forming compact, high-density clusters with rich geometric detail and stable local neighborhoods.

 doi:10.20944/preprints202601.1367.v1

Both types pose greater challenges for contextual modeling, occupancy prediction, and entropy coding due to their irregular structure and large proportion of data. This compels most methods to transform them into more uniform representations. These approaches estimate occupancy probabilities through context modeling and achieve compression via entropy coding. Typically, context models rely on spatial feature analysis, such as via 3D voxel convolutions or handcrafted features that incorporate occupancy indicators and quantization levels. More recent studies have shifted toward autoregressive techniques, which enhance compression efficiency but substantially increase inference time. While fully autoregressive models can be parallelized during encoding, decoding must proceed sequentially for each point p_t, since it depends on the prior state $p_{<t}$. This sequential dependency has motivated semi-autoregressive approaches that use grouping strategies to enable partial parallel decoding with minimal trade-offs.

Depending on their representation strategies, methods can be broadly classified into four categories: *tree-based*, *voxel-based*, *projection-based*, and *point-based* approaches. Tree-based models represent geometry via KD-trees [25] or octrees [26] and predict node occupancy hierarchically. Voxel-based methods [6] operate on quantized voxel grids with 3D convolutional networks. Projection-based approaches [27,28] compress 2D depth maps, while point-based models encode learned point-wise features directly.

2.1. Traditional Methods

Traditional methods for PCGC, such as Google's Draco [4] and MPEG's G-PCC [5], are widely adopted due to their interpretability, stability, and general effectiveness. Both convert point clouds into KD-trees and use node occupancy for context modeling. Projection-based methods, such as V-PCC [28], project 3D data into 2D representations for conventional image compression, but they suffer from projection artifacts and partial information loss. All these codecs rely on heuristically designed contexts limited to small neighborhoods, restricting overall performance. This has prompted a shift toward learned entropy models that automatically infer context information using deep neural networks. The timeline in Figure 1 highlights this steady performance improvement of deep learning-based models over traditional G-PCCv14 [5], for both sparse and dense point clouds.

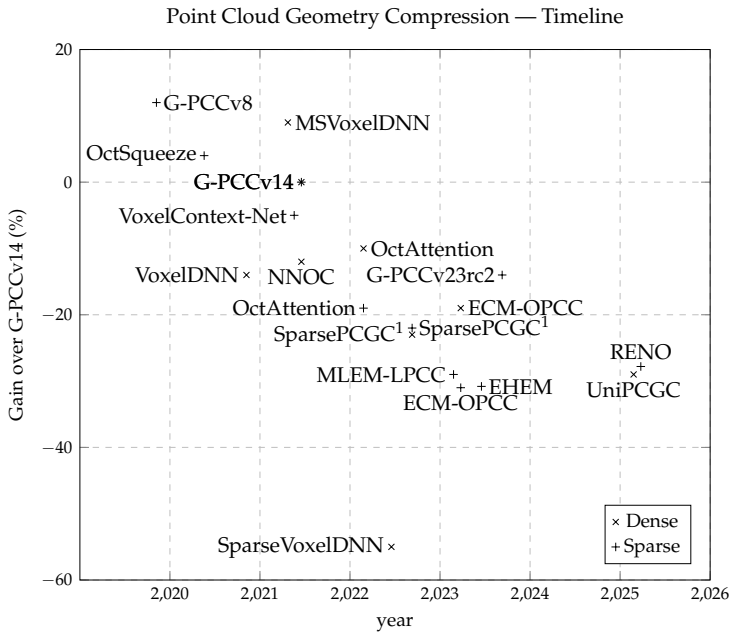


Figure 1. The timeline of key publications in lossless point cloud geometric compression over the past half decade, illustrating the steady performance improvements achieved relative to the G-PCCv14 baseline. [1]SparsePCGC was originally published in November 2021 with comparatively lower performance due to hardware limitations at the time. With the advent of newer GPUs such as the NVIDIA RTX 4090, larger model configurations and reduced inference times have become feasible, resulting in significantly improved compression performance.

 doi:10.20944/preprints202601.1367.v1

2.2. Voxel-Based Entropy Models

These methods employ 3D convolutions over voxelized occupancy grids. However, due to the cubic complexity $O(n^3)$, high-resolution voxelization can quickly exceed hardware capacity. Nguyen *et al.* introduced VoxelDNN [6], inspired by PixelCNN [29], which partitions point clouds into 64^3 voxel blocks and uses masked 3D convolutions in a fully autoregressive manner to predict voxel occupancy probabilities. Kaya *et al.* proposed NNOC [30], which explores the voxel grid layer by layer, but both models remain fully autoregressive and hence computationally expensive. MSVoxelDNN [7] partially relaxes the autoregressive dependency, achieving parallelism at the cost of higher bitrates.

Sparse convolution frameworks such as the Minkowski Engine [31,32] allow high-resolution processing. This enabled Wang *et al.* to develop SparsePCGC [8], capable of 12-bit precision using sparse convolution with an 8-stage autoregressive model. Later, UniPCGC [9] proposed a refined grouping strategy, arguing that early-stage voxels provide underrepresented spatial information. By skipping select voxels in initial groups and expanding later stages, UniPCGC improved both compression ratio and decoding time compared to SparsePCGC. SparseVoxelDNN [33] is namely the transition of VoxelDNN [6] from dense convolution to sparse convolution.

2.3. Octree-Based Entropy Models

Octree-based approaches decompose point clouds hierarchically, enabling localized feature extraction per node. OctSqueeze [10] was the first end-to-end octree-based model, predicting the 8-bit occupancy code of each node using an Multi Layer Perceptron (MLP) that processes ancestor features. However, its lack of neighborhood search and autoregression limited effectiveness, and later voxel-based approaches surpassed it. Recent octree models, including OctAttention [11], ECM-OPCC [12], and EHEM [13], leverage attention mechanisms such as transformers to capture hierarchical dependencies. They convert octrees into deterministic sequences (e.g., breadth-first order), allowing transformers to model autoregressive dependencies efficiently. Yet, the breadth-first ordering only approximates neighborhood relations and does not always reflect true spatial proximity. While these transformer-based methods excel in sparse point clouds, they are computationally heavy. For example, ECM-OPCC [12] achieves strong compression performance but requires 19.5 seconds for decoding and nearly exhausts 40 GB of GPU memory, limiting real-time applicability. EHEM [13] introduces a *hierarchical attention mechanism* designed to accelerate decoding by efficiently expanding the receptive field. It first merges the features of sibling nodes before applying the transformer, and then up-samples them back to their original resolution. This hierarchical fusion strategy effectively broadens the spatial context at low computational cost, demonstrating particular efficiency for sparse point clouds.

2.4. Hybrid Entropy Models

Hybrid models combine multiple representations – such as octrees, voxelization, and spatial coordinates – offering a balance between granularity and efficiency. VoxelContextNet [14] encodes octrees but converts each node into small voxel blocks (11^3) for 3D convolutional feature extraction. The more recent RENO [15] uses sparse convolution with embedded occupancy features (32 channels per voxel) and two autoregressive stages. Our proposed MIC-OPCCv2 consumes point-based features while maintaining an octree structure, but encodes each occupancy symbol bit-by-bit in a voxel-wise manner.

2.5. Summary

In summary, Table 1 provides a structured comparison of leading deep learning-based PCGC methods. We categorize these models according to four key aspects:

1. Type of input features;
2. Main context modeling component;
3. Type of probability output; and
4. Level of autoregression.

 doi:10.20944/preprints202601.1367.v1

The final two columns report their compression performance on sparse versus dense point clouds and their decoding times. All results are cited faithfully from original sources. It is evident that fully autoregressive models achieve strong compression but at impractical decoding times. Semi-autoregressive methods address this limitation through architectural enhancements or model scaling. Finally, voxel-based approaches tend to perform better on dense point clouds, whereas octree-based methods excel on sparse data.

Table 1. A set of deep learning-based entropy models for geometric point cloud compression, grouped and categorized by their input features, primary context modeling components, output probability representations, and autoregressive level. The last two columns on the right summarize the relative compression performance gain^{-1} over G-PCCv14 and the decoding time for both sparse and dense point clouds.

Name	Input	Context Model	Output	Auto-regressive	Sparse Dense	Time
MIC-OPCCv2	location, occupancy (binarized)	Index Convolution	voxels 0..1	semi	32.6% 30.6%	3.5s 5s
MSVoxelDNN	voxels	Masked 3D-Convolution			- -9.7%	- 58s
VoxelDNN				fully	- 13.9%	- 640s
NNOC					- 11.5%	- 1171s
SparseVoxelDNN		Masked & Sparse 3D-Convolution			- 52.8%	- 229s
SparsePCGC		Sparse 3D-Convolution		semi	22.0% 33.8%	1.2s 1.9s
UniPCGC					- 29.2%	- 0.57s
RENO	occupancy (embedded)		octree nodes 1..255	none	17.8% -	0.1s -
Voxel-Context Net	voxels	3D-Convolution			8.1% -	0.09s -
MIC-OPCCv1	location, occupancy (binarized)	Index Convolution			19.6% 30.6%	2.5s 5.0s
OctSqueeze	location, occupancy, level, octant	Multi-Layer Perceptron (MLP)			2.1% -	0.08s -
OctAttention	occupancy, level, octant (embedded)	breadth-first transformer		fully	19.5% 9.7%	530s 1229s
ECM-OPCC				semi	32.6% 19.5%	- 19.8s
EHEM					32.6% -	0.43s -

Projection-based methods are excluded from this survey due to their inherent limitations. For instance, RIDDLE [27] applies only to single range-image frames from individual sensors, whereas many real-world point clouds are composites from multiple scans and sensors [20,21,23,24].

Despite significant progress, existing methods continue to face key trade-offs between compression efficiency, decoding speed, and memory consumption. Fully autoregressive models achieve superior bitrates but suffer from prohibitively long decoding times, while semi-autoregressive or

 doi:10.20944/preprints202601.1367.v1

sparse convolution methods often sacrifice accuracy for faster inference. Transformer-based octree models, though highly expressive, require extensive computational resources and large memory footprints, making them unsuitable for real-time applications or deployment on embedded systems. To address these challenges, we propose **MIC-OPCCv2**, a lightweight and computationally efficient hybrid entropy model based on multi-indexed 1D convolution. Our approach captures spatial context through dynamically ordered convolutions, achieving fast decoding with competitive compression performance – bridging the gap between traditional octree-based and voxel-based neural codecs.

3. Background

Our method builds on the same foundational principles as previous work, leveraging Shannon's theorem [34], which defines the entropy h of a symbol s as

$$h(s) = -\log_2 p(s). \tag{1}$$

This formulation defines the entropy h as the theoretically required number of bits to represent a symbol s, where the probability $p(s)$ lies within the range $0 < p(s) < 1$. During decoding, however, the true probability distribution $p(s)$ is uncertain and can only be approximated by a stochastic, heuristic, or empirical model. We express this approximation as

$$q(\mathbf{x}) = \hat{\mathbf{y}}, \tag{2}$$

where $\mathbf{y}$ is a one-hot encoded vector representing the ground truth symbol s, and $\hat{\mathbf{y}}$ is the estimated probability distribution inferred from the contextual input $\mathbf{x}$. Our goal is to optimize $q_\theta(\mathbf{x})$ – a parameterized empirical model, such as one driven by deep learning – by minimizing the cross-entropy between $\mathbf{y}$ and $Q_\theta(x_i)$:

$$H(y, x) = -\sum_{i=1}^{n} y_i \log Q_\theta(x_i), \tag{3}$$

where $H(y, x)$ measures the divergence between the true and estimated distributions. An Arithmetic Encoder [35] can then transform the symbol s into a compact bitstream, with the efficiency of the encoding directly tied to the accuracy of the probability estimate $Q_\theta(x_i) = \hat{\mathbf{y}}_i$.

3.1. Binary Arithmetic Coding

Arithmetic coding is a form of entropy encoding used in lossless data compression. Unlike Huffman coding [36], which assigns discrete bit patterns to each symbol, arithmetic coding encodes an entire message into a single fractional number within the interval $[0, 1)$. This approach allows for fractional bit efficiency, resulting in compression rates close to the theoretical entropy limit.

In arithmetic coding, a message is represented by a progressively refined numerical interval. Each new symbol subdivides the current interval into smaller partitions according to its estimated probability. The more probable a symbol is, the smaller the range reduction it induces – thus requiring fewer bits to encode. This recursive subdivision continues for all symbols in the message, producing an increasingly precise numerical representation. When the final interval is specified to sufficient precision, its binary representation serves as the compressed output.

Binary Arithmetic Coder (BAC), a simplified form of arithmetic coding, performs recursive interval subdivision based on binary decisions between the most probable symbol and the least probable symbol. The interval width corresponds to the product of the symbol probabilities, and the binary representation of the final interval asymptotically approaches the Shannon theorem's lower bound on entropy. A key advantage of BAC is its clean separation between modeling and coding, allowing a probability model to be optimized independently from the coding engine – a property central to our proposed approach.

 doi:10.20944/preprints202601.1367.v1

3.2. Fast Octree Coder

An octree provides an efficient means of organizing unstructured and unsorted data through recursive spatial partitioning [26]. However, costly spatial computation and floating-point arithmetic can be avoided by Fast Octree Coding (FOC) as shown in Figure 2. FOC simply groups a pre-quantized point cloud Q based on the t most significant bits of each component in a quantized point $\mathbf{q} = \mathbf{p}_{b_{L-t}}$, where t reflects the current quantization level and tree depth. Each group raises a bit on the 8-bit occupancy symbol s according to its octant signature at position t. This algorithm allows fast random access to any depth-level in the tree at a complexity of $\mathcal{O}(\log_2 n)$

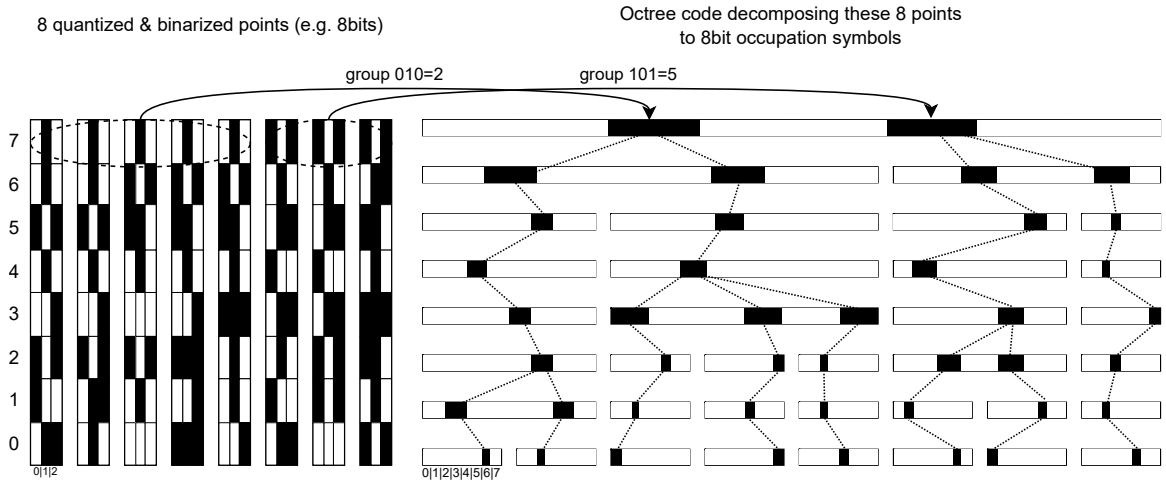


Figure 2. Fast Octree Coding. Any octree level at depth t can be constructed instantly with a complexity of $\mathcal{O}(\log_2 n)$ by grouping points according to their $3t$ most significant bits, enabling efficient hierarchical partitioning of the quantized point cloud.

A breadth-first traversal of the octree, expressed as $S = \{\mathbf{s}_0 = (s_0, \ldots, s_j), \ldots, \mathbf{s}_L\}$, typically yields much lower entropy per symbol s than the raw spatial coordinates $\mathbf{p} = (x, y, z)$ of a point cloud P.

Earlier octree-based models, such as OctSqueeze [10], included the real-valued positions of each octree node as features. However, the absence of real position values in recent works [9,11–13,15] implies that this feature is less effective for spatial context modeling. Instead, embedded occupancy flags, 3-bit octants, and quantization depth provide more discriminative and compact features.

4. Methodology

Building upon the limitations identified in prior research, we propose MIC-OPCCv2 – an enhanced version of our lightweight multi-index convolutional entropy model [16] designed for efficient octree-based point cloud compression. An overview of the framework is provided in Figure 3. The architecture integrates four key components:

1. a *Progressive Grouping* scheme, which introduces autoregressive dependencies between voxel candidates at identical resolution levels;
2. a *Blender & Distiller* network, which refines hierarchical representations through residual feature aggregation;
3. a *multi-index convolution* backbone, enabling spatially coherent context extraction at minimal computational cost; and
4. a *binary location* encoder, serving as a compact and expressive geometric feature representation.

Together, these components allow MIC-OPCCv2 to achieve competitive or state-of-the-art compression performance while operating with significantly lower memory usage and substantially reduced decoding complexity compared to existing neural PCGC methods.

 doi:10.20944/preprints202601.1367.v1

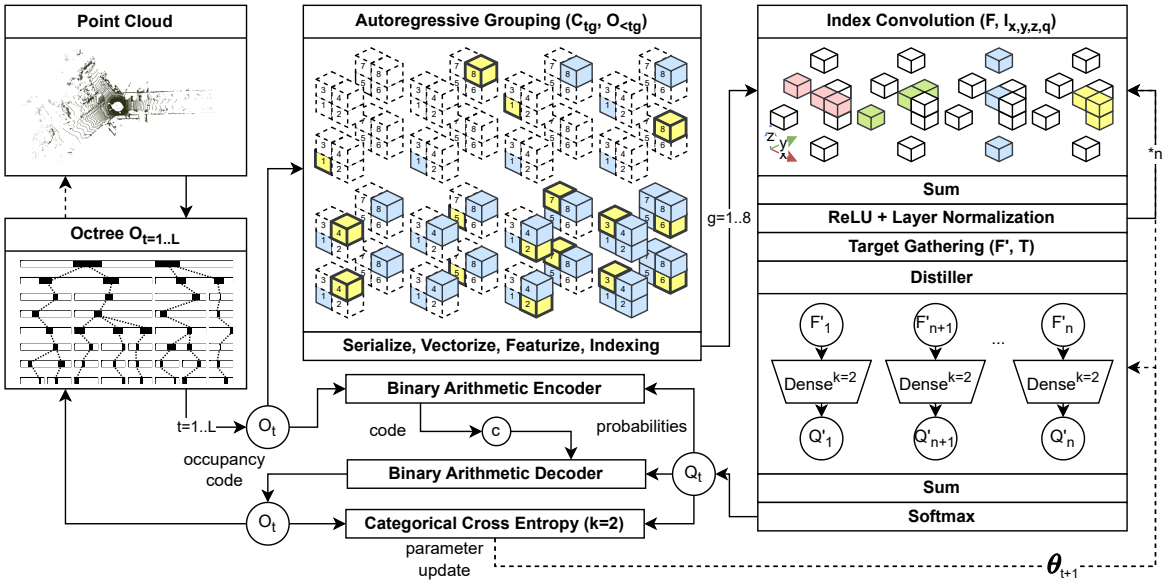


Figure 3. Overview of the MIC-OPCCv2 architecture. The input point cloud is quantized and binarized to form an octree representation. Each octree node expands into eight voxel candidates, which are arranged into eight autoregressive groups following a checkerboard scheduling scheme. Each group is processed by the Blender & Distiller network using multi-index convolution to estimate occupancy probabilities. These probabilities are consumed by a BAC during compression and are supervised through categorical cross-entropy during training. The resulting occupancy symbols can be converted back into an octree and subsequently into a reconstructed quantized point cloud.

The following sections detail the core elements of the proposed codec. Section 4.1 introduces the multi-index convolution mechanism. Section 4.2-4.3 describes the Blender & Distiller network and the Progressive Grouping strategy. Section 4.4 outlines the construction of binarized location features, and Section 4.5 presents the training objective and loss formulation.

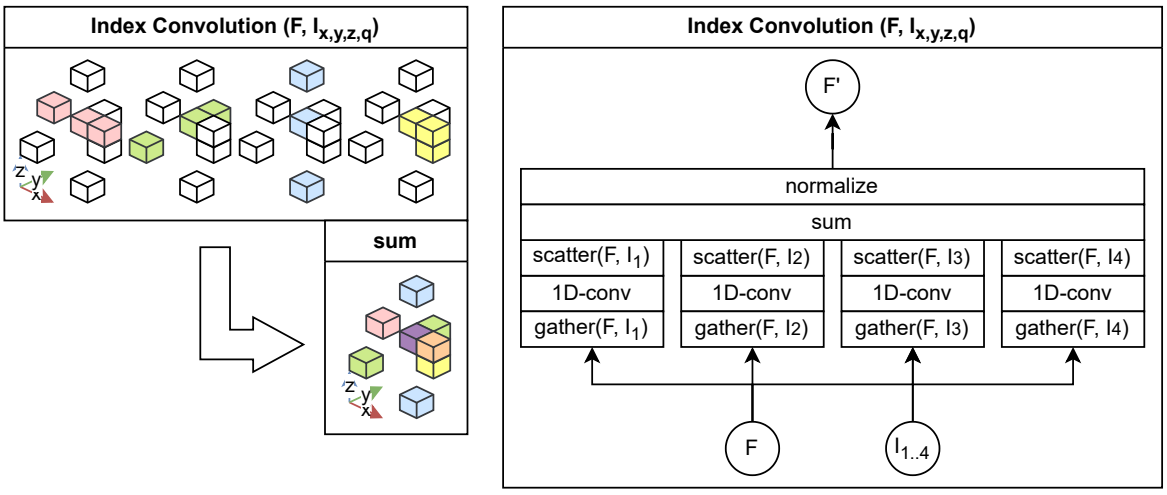


Figure 4. Illustration of the proposed multi-indexed 1D convolution process, which reorders spatial features along multiple dimensions to form directionally variant receptive fields. Each convolution output is reverted and aggregated in the original order to accumulate contextual information in all directions.

4.1. Index Convolution

Transformer-based methods [11–13] process octree nodes sequentially following a breadth-first traversal order. However, this neighborhood ordering is not always spatially consistent, as nearby points or voxels may appear far apart in the traversal sequence as ocrtee nodes. To overcome this limitation, we introduce a multi-indexed 1D convolution operation.

 doi:10.20944/preprints202601.1367.v1

For each spatial dimension, an `arg-sort` operation generates an index that defines the processing order. This index is created once at the beginning. The point features are then reordered according to this index before applying 1D convolution. After convolution, the outputs are reverted to their original order using the inverse index and concatenated. Repeating this process across all dimensions progressively enlarges the model's receptive field, enabling the aggregation of broader spatial dependencies.

Our indexed convolution can be interpreted as an approximated low-complex k-nearest-neighbor lookup along fixed axes, effectively merging adjacent voxels in sorted order. Unlike previous methods that rely on zero-padding for empty voxels within a fixed receptive window, our approach ensures that only existing voxels contribute to the receptive field. This dynamic adaptation allows the model to handle sparse regions more effectively by implicitly adjusting the receptive field size.

4.2. Blender & Distiller

Samples in PCGC exhibit a wide range of entropy characteristics. Low-resolution samples tend to emit low entropy, while high-resolution samples approach near-random distributions with substantially higher entropy. Consequently, prior research has emphasized the value of residual network architectures to accommodate such variability [9,17,33]. The underlying intuition is that low-entropy samples can be effectively modeled with shallow networks, whereas high-entropy samples require deeper networks capable of capturing more complex causal dependencies.

Voxel occupancy estimation, as formulated in PCC, is conceptually analogous to image segmentation. In this domain, *U-Net* architectures [18,37] have demonstrated superior performance compared to conventional *ResNet*-based models, primarily due to their ability to combine multi-scale contextual information through skip connections. Inspired by these findings, we propose a **Blender & Distiller** architecture, illustrated in Figure 5.

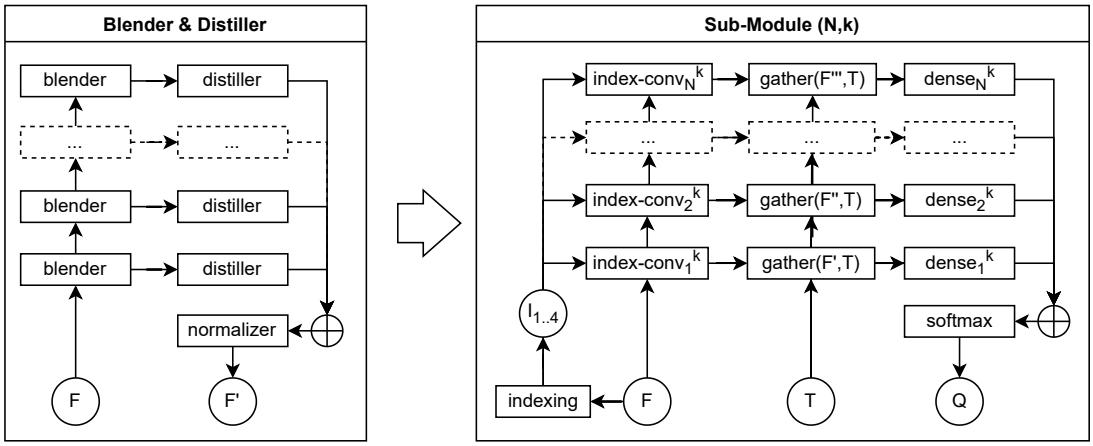


Figure 5. The proposed Blender & Distiller architecture is a strongly residual framework inspired by U-Net [18], but differs by incorporating a multi-stage voting mechanism on the output side, where each distiller contributes to the final probability estimation.

In our design, the *blender layers* act as recursive feature aggregators that combine information across different receptive fields. Aggregation can be achieved through summation, concatenation, convolution, or attention-based fusion. In our case, multi-index convolution serves as the primary aggregation mechanism, progressively refining spatial context representations across hierarchical depth. Each blender layer thus produces increasingly complex hidden features, with deeper layers contributing to broader spatial understanding.

The *distiller layers*, on the other hand, are responsible for extracting probability estimations from each blender stage. Their outputs are subsequently combined through a normalized summation, effectively forming a collective "vote" across all hierarchical levels of the network. This mechanism

 doi:10.20944/preprints202601.1367.v1

enforces contribution from every stage, ensuring that intermediate representations meaningfully participate in the final prediction. Empirically, this voting-based mechanism proves highly effective for PCGC, as it maintains strong contextual awareness across both local and global spatial scales.

Unlike the U-Net [18], our design omits connections between distiller inputs, simplifying the data flow while maintaining hierarchical feature consistency. For binary classification tasks, each distiller outputs two positive unconstrained logits. This formulation allows a distiller to either abstain from contributing (by emitting near-zero magnitudes) or to dominate the final consensus when confident. Through backpropagation, each distiller dynamically adjusts its contribution to the aggregate prediction, facilitating adaptive behavior across samples of varying entropy.

Formally, the aggregated output of all N distillers is expressed as

$$F_{t,g}^{N+1} = \sum_{i=1}^{N} \text{Distiller}_i(F_{t,g}^{i}), \tag{4}$$

$$Q_\theta\left(C_{t,g} \middle| O_{<(t,g)}\right) = \text{SoftMax}(F_{t,g}^{N+1}), \tag{5}$$

where $\text{Distiller}_i(F_{t,g}^{i})$ denotes the i-th distiller output operating on hidden features $F_{t,g}^{i}$ derived from the preceding blender layer. The softmax normalization converts the summed logits into a probability distribution $Q_\theta(C_{t,g}|O_{<(t,g)})$, which represents the estimated occupancy likelihood for voxel candidates $C_{t,g}$ conditioned on the previously decoded occupancies $O_{<(t,g)}$.

These probabilities directly feed into the categorical cross-entropy loss described in Section 4.5, ensuring that learning is guided by the aggregated multi-scale consensus of all distillers. In this way, the Blender & Distiller network forms a mathematically consistent bridge between spatial feature extraction and probabilistic entropy modeling, enabling efficient and robust learning across diverse octree resolutions.

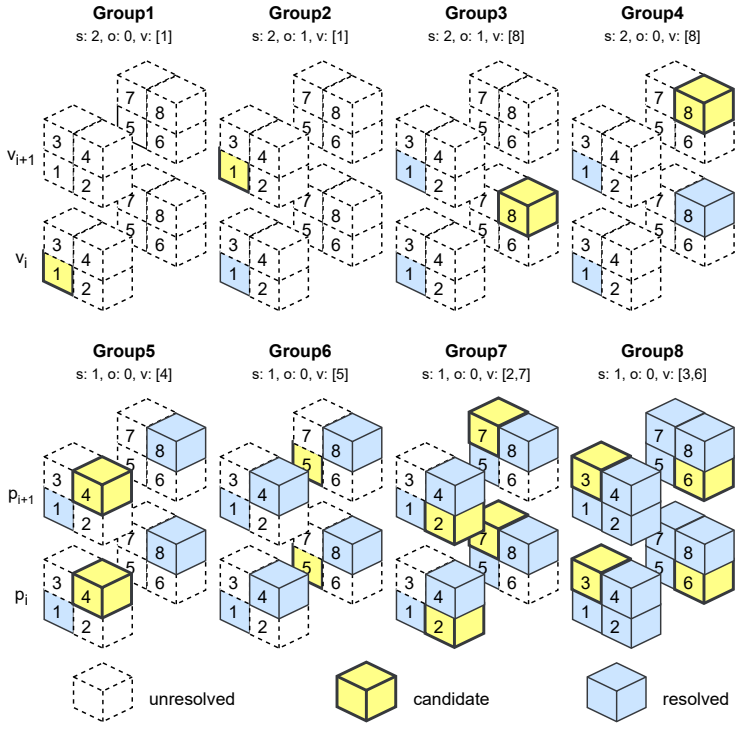


Figure 6. Progressive grouping strategy of MIC-OPCCv2. Each point $\mathbf{p}_i$ expands into a block of eight voxel candidates v. For every autoregressive group, voxel candidates are selected according to a stride s, an offset o, and a target index v. Early groups operate under limited spatial context and therefore decode only a subset of voxels, enabling more reliable probability estimation for later groups. As contextual information accumulates, later groups increase throughput by decoding multiple voxels concurrently.

 doi:10.20944/preprints202601.1367.v1

4.3. Proregressive Grouping

To incorporate autoregressive dependencies efficiently, we adopt a grouping strategy inspired by UniPCGC [9], which improved upon the SparsePCGC [8] framework. The core idea is to process subsets of voxels iteratively, balancing early-stage contextual learning with late-stage decoding efficiency. Our variant of a *Progressive Grouping* scheme alternates this approach by skipping every second voxel in the first 4 groups, pioneering with a lower number of voxel candidates into that unknown distribution, followed by denser processing in later groups when more of the actual distribution reveals. This strategy effectively balances context enrichment and decoding speed, leading to improved overall performance.

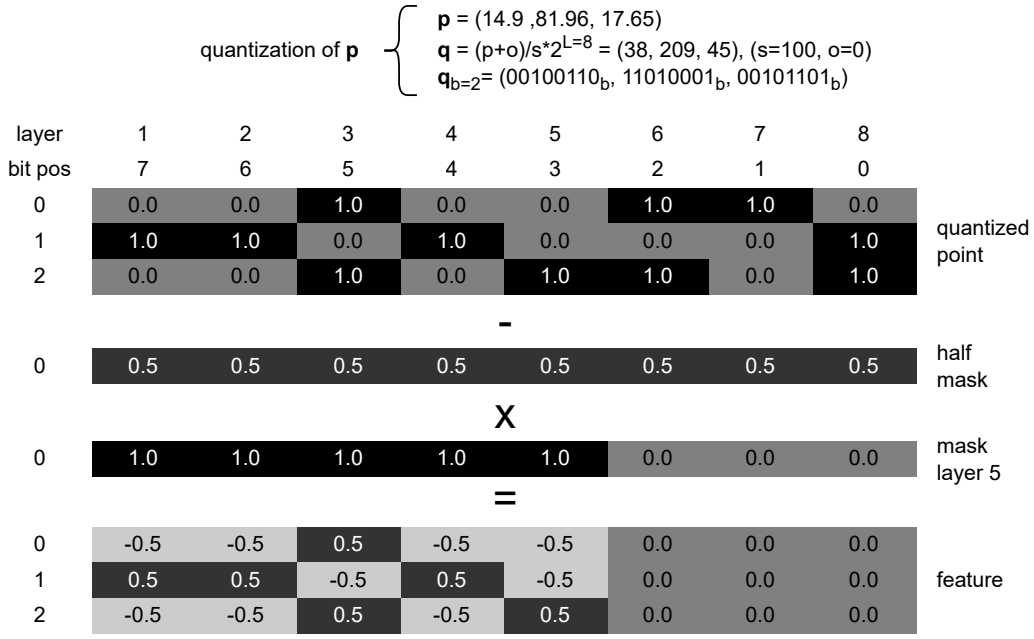


Figure 7. Illustration of the binarization, normalization, and masking process for feature generation at quantization level $L = 8$ and current coding depth $t = 5$. Bits beyond the current depth are masked to prevent access to unavailable information during decoding.

4.4. Binary Location Encoding

Each point $\mathbf{p} = (x, y, z) \in \mathbb{R}$ is first normalized and quantized according to

$$\mathbf{q} = \left\lfloor \frac{2^L}{s}(\mathbf{p} - o) \right\rfloor, \tag{6}$$

ensuring all coordinates are positive and confined to the range $[0, 2^L)$. A binarization function generates bit-level representations:

$$b_i(p) = \left\lfloor \frac{p}{2^i} \right\rfloor \bmod 2, \tag{7}$$

and concatenation across axes produces hierarchical binary descriptors:

$$\mathbf{b}_L(\mathbf{q}) = [b_{L-1}(z), b_{L-1}(y), b_{L-1}(x), \ldots, b_0(z), b_0(y), b_0(x)]. \tag{8}$$

A mask $\mathbf{m}_{td}$ ensures that only information up to the current decoding depth t is visible:

$$m_i = \begin{cases} 1, & \text{if } i < td, \\ 0, & \text{otherwise} \end{cases}. \tag{9}$$

 doi:10.20944/preprints202601.1367.v1

Finally, the masked binary features are centered to $[-0.5, 0.5]$ to distinguish true zeros from masked bits:

$$F^0 = \{(\mathbf{b}_L(\mathbf{p}_j) - 0.5)\mathbf{m}_{td} \mid \forall j\}. \tag{10}$$

Using this binarization scheme, variant indices I can be derived for different voxel orderings, enabling diverse spatial perspectives in the multi-index convolution process. By summing the bits across the quantization levels L and dimensions d, we obtain an index I_q corresponding to the *breadth-first traversal order* of an octree:

$$I_q = \left\{ \sum_{k=0}^{L-1} \sum_{d=1}^{3} b_k(p_d) 2^{3k+d-1} \middle| \forall \mathbf{p} = (x, y, z) \right\}. \tag{11}$$

Alternatively, by summing the binary bits along dimensions first and then over quantization levels, we derive *axial voxel orderings*, such as I_x, I_y, and I_z:

$$I_x = \left\{ \sum_{d=1}^{3} \sum_{k=1}^{L} b_{L-k}(p_d) 2^{Ld-k} \middle| \forall \mathbf{p} = (x, y, z) \right\}, \tag{12}$$

$$I_y = \left\{ \sum_{d=1}^{3} \sum_{k=1}^{L} b_{L-k}(p_d) 2^{Ld-k} \middle| \forall \mathbf{p} = (y, z, x) \right\}, \tag{13}$$

$$I_z = \left\{ \sum_{d=1}^{3} \sum_{k=1}^{L} b_{L-k}(p_d) 2^{Ld-k} \middle| \forall \mathbf{p} = (z, x, y) \right\}. \tag{14}$$

By cyclically swapping the dimensional components in $\mathbf{p}$, we obtain different axial orderings for each dimension d. These variant indices serve as sorting references for our *multi-index convolutions*, allowing the model to perceive spatial relationships from multiple directional perspectives.

4.5. Loss Function

The training objective of MIC-OPCCv2 is to minimize the *categorical cross-entropy loss* between the predicted probability distribution $P_{t,g}$ and the true occupancy label $O_{t,g}$ for each voxel, across all resolution levels t and autoregressive groups g. The loss function is formulated as:

$$\mathcal{L}_{t,g} = \mathbb{E}_{O_{t,g} \sim P_{t,g}} \left[-\log \mathcal{Q}_\theta(C_{t,g} \middle| O_{<(t,g)}) \right], \tag{15}$$

where $\mathcal{Q}_\theta$ represents our probability estimation model parameterized by θ, applied over the current voxel candidates $C_{t,g}$ and the already processed occupancy symbols $O_{<(t,g)}$ from previous decoding steps.

For each target voxel, the model outputs two class probabilities – occupied versus empty – using a softmax activation applied to the final feature representation $F_{t,g}^{N+1}$, as defined in Equation 5.

Each distiller layer Distiller_i implements a fully connected (dense) layer with kernel size $k = 2$, using *SoftPlus* for unbound positive activation:

$$\text{Distiller}_i(F_{t,g}^i) = \text{SoftPlus}\left(\text{Dense}_i^{k=2}(F_{t,g}^i)\right). \tag{16}$$

The input features $F_{t,g}^i$ correspond to the subset of encoded voxels $T_{t,g}$ at the current stage i, defined as:

$$F_{t,g}^i = \{x \mid \forall n \in T_{t,g} : x_n \in F^i\}. \tag{17}$$

 doi:10.20944/preprints202601.1367.v1

The distillers extract two scalar outputs per voxel, corresponding to the probability logits of occupancy and emptiness. These logits are derived from the recursively accumulated hidden features F^i, obtained via the *blender layers*:

$$F^i = \text{Blender}_i(F^{i-1}), \tag{18}$$

where each blender performs feature aggregation across all spatial indices using layer normalization and ReLU activation:

$$\text{Blender}_i(F^{i-1}) = \text{LayerNorm}\left(\sum_{d=1}^{4} \text{ReLU}\left(\text{IndexConv}_{i,d}^{k=64}(F^{i-1}, I_d)\right)\right). \tag{19}$$

This formulation summarizes four index-convolutions with different sorting orders I_d, starting from the initial feature set F^0:

$$F^0 = \left\{\mathbf{b}_L(\mathbf{p}) \mid \forall \mathbf{p} \in \bigcup\{C_{t,g}, O_{<(t,g)}\}\right\}, \tag{20}$$

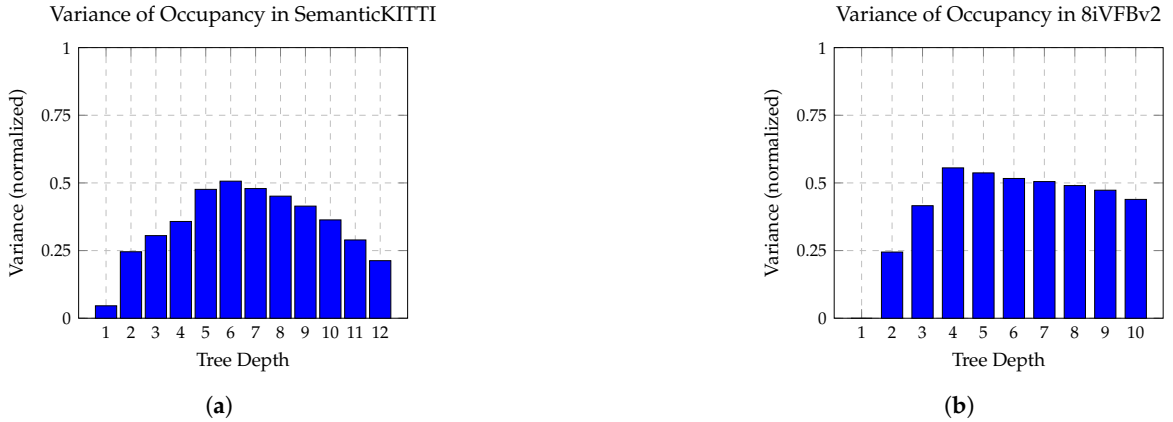
where $\mathbf{b}_L(\mathbf{p})$ represents the binarized and serialized feature vector of spatial locations $\mathbf{p}$. Here, $O_{<(t,g)}$ refers to all decoded voxels from prior steps, and C_{tg} represents candidate voxels generated for the current group. Our overall probability mass function is therefore:

$$\mathcal{P}_\theta(\mathcal{O}) = \prod_{t=1}^{L}\prod_{g=1}^{G=8} \mathcal{Q}_\theta\left(C_{t,g}\middle|O_{<(t,g)}\right). \tag{21}$$

This autoregressive structure allows MIC-OPCCv2 to iteratively refine spatial and contextual representations across layers and groups, improving the accuracy of occupancy probability estimation for each voxel candidate.

4.6. Model Splitting

Each octree level exhibits distinct entropy characteristics due to the hierarchical refinement of spatial resolution, as illustrated in Figure 8. The upper levels near the root generally show low entropy, since most nodes are fully occupied when the point cloud is centered. Intermediate levels contain the widest variety of occupancy patterns and therefore present the highest entropy. At deeper levels, although spatial precision increases, occupancy becomes sparse and the entropy decreases again, as fully occupied patterns are rare.



(**a**) (**b**)

Figure 8. Variance in occupancy distributions across octree levels. (**a**) SemanticKITTI (sparse): entropy is low at shallow and deep levels, but peaks at mid-levels where structural variation is highest. (**b**) 8iVFBv2 (dense): occupancy distributions maintain consistently high variance starting from level 4 due to dense and surface-complete geometry.

 doi:10.20944/preprints202601.1367.v1

In principle, each octree level could be processed by an individually parameterized sub-module with dedicated convolutional and dense layers, enabling optimal adaptation to its entropy profile. However, such a design is computationally expensive, memory-intensive, and reduces parameter sharing, which may impair generalization. To balance expressiveness and efficiency, we group neighboring octree levels into a limited number of sub-modules, each tailored to a characteristic entropy regime, for instance:

- **Module A (Shallow Levels):** Processes the coarsest levels, where occupancy is dense and entropy is low. A compact configuration with fewer parameters is sufficient, as occupancy patterns are highly predictable.
- **Module B (Intermediate Levels):** Covers the layers with the greatest variability in occupancy and the highest entropy. Here, a more expressive representation is required. Since spatial structure is still relatively coarse, increasing the channel dimension k is more effective than increasing the number of layers N.
- **Module C (Deep Levels):** Handles the finest levels, where occupancy is sparse and spatial geometry must be inferred from broader context. In this regime, a large receptive field is essential, favoring a higher depth N of convolutional layers to propagate long-range dependencies.

This level-wise grouping strategy preserves predictive accuracy across the entire octree depth while controlling computational cost, enabling efficient modeling of both coarse global structure and fine-grained geometric detail. Moreover, the modular decomposition allows individual sub-modules to be fine-tuned, replaced, reused, or extended independently, facilitating adaptation of the codec to different application requirements such as memory constraints, target resolution, or operational latency.

5. Experiments

This section outlines the experimental setup used to evaluate the proposed MIC-OPCCv2 framework, including datasets, implementation details, and evaluation metrics. We then compare our method against representative state-of-the-art PCGC approaches on both sparse and dense benchmarks.

5.1. Datasets

To verify the robustness of MIC-OPCCv2 across different geometric structures, we evaluate the model on both *sparse* and *dense* point cloud datasets. For fair comparison, we adopt the same training and test splits as previous studies [12].

5.1.1. SemanticKITTI

The SemanticKITTI dataset [19] consists of large-scale outdoor LiDAR scans captured from a rotating 64-beam sensor at 10 Hz. Each frame contains roughly 10^5 points with coordinates $\mathbf{p} = (x, y, z, l)$, where the intensity l is not considered in our experiments. The point clouds are stored in 32-bit floating-point precision with a spatial resolution of 10^{-4} meters and span up to approximately 270 m in diameter, requiring 18 bits per coordinate for lossless quantization. Following the standard protocol, sequences 00–10 are used for training, while the remaining sequences serve for testing.

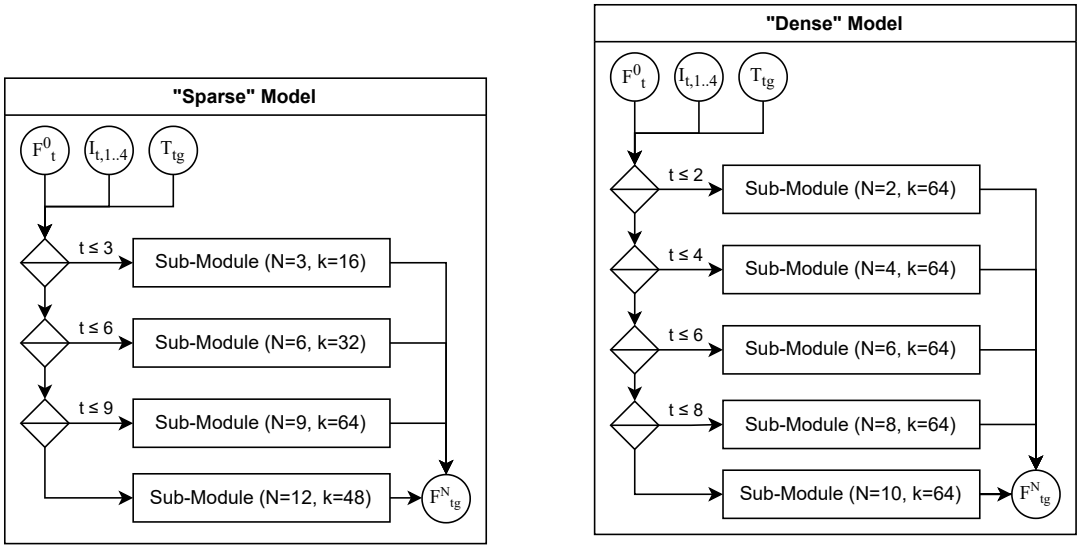
5.1.2. MPEG's 8i Voxelized Full Bodies (8iVFBv2)

The 8iVFBv2 dataset [23] is used to evaluate performance on dense point clouds composed of voxelized human scans. Although the dataset contains both geometry and color, only geometry is considered in this work. The dataset is widely adopted due to its standardized quantization levels (10–12 bits), enabling controlled R-D comparisons. Following common practice [38], we evaluate four benchmark sequences: *Redandblack* and *Loot* (300 frames each), as well as single frames from *Thaidancer* and *Boxer*, while *Longdress* and *Soldier* are used for training.

 doi:10.20944/preprints202601.1367.v1

5.2. Implementation Details

MIC-OPCCv2 is implemented in `TensorFlow 2.9` and employs an integrated arithmetic coder provided by Ballé *et al.* [39]. We configure two model variants (as illustrated in Figure 9): one optimized for sparse point clouds and one for dense point clouds.



(**a**) (**b**)

Figure 9. Model configurations for sparse and dense point clouds. (**a**) The sparse configuration employs four sub-modules, each with a distinct number of blender layers N and kernel size k, reflecting the varying entropy across octree depths in sparse point clouds. (**b**) The dense configuration assumes a more uniform occupancy distribution, using five sub-modules with fixed kernel size and gradually increasing depth to accommodate consistently high spatial detail.

The *sparse model* supports a quantization precision of up to $L = 12$ and is divided into four sub-modules (Section 4.6), with each sub-module responsible for three octree levels. These sub-modules use $N = [3, 6, 9, 12]$ blender layers and kernel sizes $k = [16, 32, 64, 48]$ respectively. The total number of trainable parameters remains below one million, and the memory footprint during decoding does not exceed 2 KB per point.

The *dense model* supports a quantization precision of up to $L = 10$ and is composed of five sub-modules, each covering two octree levels. Here, the sub-modules use $N = [2, 4, 6, 8, 10]$ blender layers with a constant kernel size of $k = 64$. This configuration results in approximately 1.4 M trainable parameters, and the memory footprint during decoding remains below 4 KB per point.

5.3. Training Details

Both model variants are trained on single NVIDIA RTX 3090 GPU with 24 GB VRAM for approximately 30 epochs using the ADAM [40] optimizer, an initial learning rate of 10^{-4} with exponential decay (0.9), and a dropout rate of 0.01 to mitigate overfitting. The loss function follows Equation (15), computed over all octree levels and autoregressive groups. The final training accuracies reach about 88% (sparse) and 94% (dense) for occupancy prediction.

5.4. Baseline

We compare MIC-OPCCv2 against representative traditional and neural point cloud geometry compression methods, including G-PCCv14 [5], VoxelContextNet [14], SparseVoxelDNN [33], SparsePCGCv2 [8], UniPCGC [9], RENO [15], EHEM [13], and ECM-OPCC [12]. Methods that are

 doi:10.20944/preprints202601.1367.v1

either superseded by more recent variants or consistently underperform traditional codecs are omitted. Furthermore, approaches such as MuSCLE [41], RIDDLE [27], and MLEM-LPCC [42], which rely on latent feature reconstruction and therefore operate under a different compression paradigm, are excluded as well to maintain fair comparison.

5.5. Evaluation Metrics

We follow MPEG standard evaluation procedures [38] and report:

- **Bits per point (bpp):** average coding cost per point.
- **D1-PSNR (dB):** point-to-point PSNR for geometric reconstruction fidelity.
- **D2-PSNR (dB):** point-to-plane PSNR for surface reconstruction fidelity.
- **BD-Rate (%):** Bjøntegaard delta bitrate relative to a reference model [43].
- **Runtime (s/frame):** wall-clock encoding and decoding time.

For sparse compression on SemanticKITTI, we evaluate R-D using:

$$\text{D1-PSNR} = 10 \log_{10} \frac{3p^2}{\text{MSE}_{sym}}, \tag{22}$$

$$\text{MSE}_{sym} = \frac{1}{2}\big(\text{MSE}(P, \hat{P}) + \text{MSE}(\hat{P}, P)\big), \tag{23}$$

where

$$\text{MSE}(P, \hat{P}) = \frac{1}{|P|} \sum_i \min_j |\mathbf{p}_i - \hat{\mathbf{p}}_j|^2, \tag{24}$$

and the point-to-plane version uses

$$\text{MSE}_{\mathbf{n}}(P, \hat{P}) = \frac{1}{|P|} \sum_i \min_j |\mathbf{n}_i(\mathbf{p}_i - \hat{\mathbf{p}}_j)|^2 \tag{25}$$

where $\mathbf{n}_i$ is the normal-vector for $\mathbf{p}_i$.

For lossless dense compression, we report BD-Rate relative to G-PCCv14.

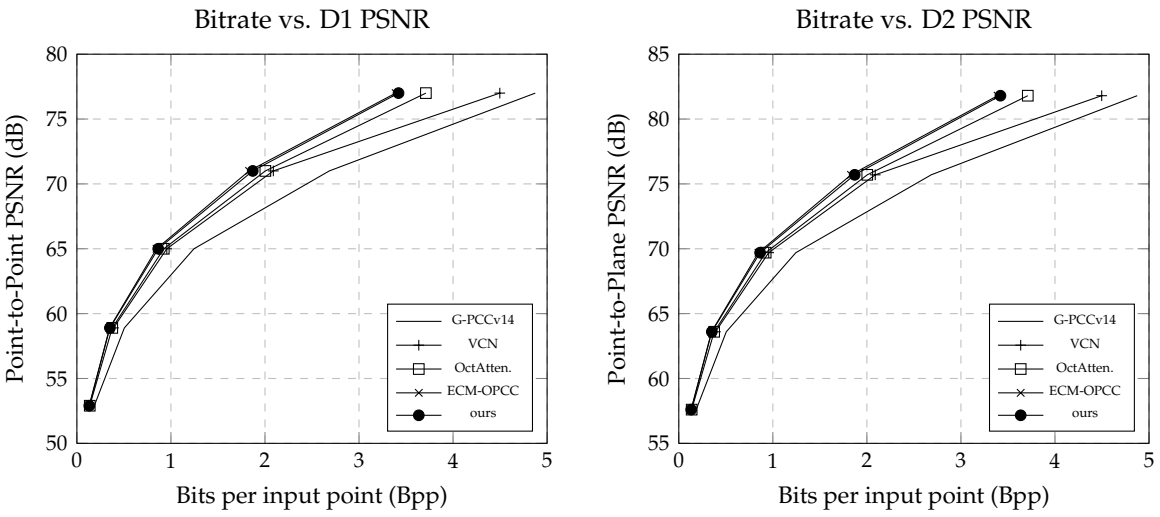
6. Results

Figure 10 illustrates the R-D of MIC-OPCCv2 on the SemanticKITTI dataset. With a quantization precision of $L = 12$, our method achieves a bitrate of 3.42 bpp at a D2-PSNR of 82 dB, matching ECM-OPCC and thereby approaching the current state-of-the-art for sparse point cloud geometry compression.

VoxelContextNet, SparsePCGCv2, and ECM-OPCC all employ identical quantization strategies, allowing direct comparison of their PSNR values. In contrast, RENO and EHEM assume a peak signal of $p = 59.7$ and apply different quantization configurations. To ensure a fair evaluation, we normalize their PSNR values to $p = 1$ but adopt their quantization configurations accordingly, as reflected in Table 2. Despite not being trained at these distortion levels, MIC-OPCCv2 consistently outperforms both RENO and EHEM in R-D. This indicates that our model captures an underlying geometric representation that generalizes effectively across different resolutions, rather than relying on quantization-specific fitting.

For dense geometry, evaluated on the 8iVFBv2 dataset, MIC-OPCCv2 attains an average bitrate of 0.50 bpp, outperforming UniPCGC and ECM-OPCC by a BD-Rate of $\sim$ 1.14% and $\sim$ 11.2%, respectively, while trailing behind SparseVoxelDNN (see Table 3). However, MIC-OPCCv2 decodes in approximately 5 seconds, making it $\sim 40\times$ faster than SparseVoxelDNN and $\sim 4\times$ faster than ECM-OPCC, though still slower than UniPCGC, which reports 0.57 seconds per frame.

 doi:10.20944/preprints202601.1367.v1



(**a**) (**b**)

Figure 10. Results of our proposed MIC-OPCC model against state-of-the-art baselines on the SemanticKITTI dataset. While achieving comparable compression ratios to OctAttention, our model significantly outperforms it in both encoding and decoding speed.

Table 2. Theoretical computational complexity of three leading point cloud compression architectures and their corresponding practical decoding times.

	Quantization $L = 12$	**D1-PSNR (dB)** $p = 1$	bpp	time (s)	**ours** bpp	**ours** time (s)
ECM-OPCC	$\left\lfloor P \frac{2^L-1}{\max(P)-\min(P)} \right\rfloor$	77.2	3.39	-	3.41	4.0
EHEM	$\left\lfloor P \frac{2^L-1}{400} \right\rfloor$	60.0	2.6	0.43	1.99	3.0
RENO	$\left\lfloor P \frac{1000}{2^{18}-L} \right\rfloor$	61.8	2.9	0.05	2.41	3.6

Table 3. Average compression ratio (bpp) of 8iVFBv2, showing the gain over G-PCCv14 and coding time.

Point Clouds	**GPCC**	**MIC-OPCCv2**	**UniPCGC**	**SparseVoxelDNN**	**ECM-OPCC**
Redandblack	0.82	0.57	0.59	0.41	0.66
Loot	0.69	0.48	0.49	0.32	0.55
Thaidancer	0.70	0.50	0.51	0.33	0.58
Boxer	0.65	0.43	0.45	0.30	0.51
Average Bpp	0.72	0.50	0.51	0.34	0.58
Gain	0.0%	30.6%	29.2%	52.8%	19.4%
Enc time(s)	1.99	5.0	0.56	7.2	1.92
Dec time(s)	1.49	5.0	0.57	229	19.5
Test Device	i7	RTX3090	RTX4080	RTX3090	RTX3090

While the model achieves competitive or state-of-the-art R-D performance, the observed decoding latency highlights the need for further implementation-level optimization to fully exploit the theoretical efficiency of the architecture.

Overall, the results demonstrate that incorporating the autoregressive grouping strategy markedly enhances the effectiveness of multi-index convolution. MIC-OPCCv2 surpasses every baseline model in at least one of the two key performance dimensions: bitrate efficiency or decoding speed. This indicates that the model generalizes well across both sparse and dense point cloud distributions,

 doi:10.20944/preprints202601.1367.v1

achieving competitive or state-of-the-art R-D. However, the current decoding latency suggests that additional engineering optimizations are still needed to fully realize the theoretical efficiency of the proposed architecture.

7. Discussion

The experimental results from Section 6 demonstrate that MIC-OPCCv2 effectively bridges the gap between sparse convolutional and transformer-based PCGC models. By eliminating the attention overhead while preserving long-range context through multi-index convolution, the method achieves a favorable trade-off between R-D and computational efficiency. Moreover, its autoregressive grouping scheme enables flexible decoding parallelism, bringing it closer to real world application, but yet struggles with real-time performance due to implementation issues.

We discuss the latency problem by analyzing the theoretical runtime complexity in the following Section 7.1. Furthermore, in Section 7.2 we demonstrate and illustrate how the perceptive field of multi-index convolution expands in sparse and dense point clouds differently, to discuss strength and weakness of our methods, that may leave room for future investigation and improvement. To quantify the contribution of each proposed component, we present an ablation study in Section 7.3, demonstrating how voxel-wise prediction, the *Blender & Distiller* network, and Progressive Grouping each incrementally improve compression efficiency.

7.1. Runtime Complexity

The key idea of our multi-index convolution is to achieve a large receptive field in three-dimensional space while maintaining low computational complexity. Table 4 summarizes the theoretical complexity of transformers, Minkowski convolution, and our proposed multi-index convolution, alongside their measured decoding times.

Table 4. Theoretical computational complexity of three leading point cloud compression architectures and their corresponding practical decoding times.

Method	Complexity	Example	Decoding Time (s)
Transformer	$\mathcal{O}\left(n(3ckw + kw^2)\right)$	ECM-OPCC	19.5
Minkowski Convolution	$\mathcal{O}\left(m(ckw^d)\right)$	UniPCGC	0.57
Multi-Index Convolution	$\mathcal{O}(n(ckwi))$	MIC-OPCCv2	5.0

It is evident that transformer architectures exhibit the highest computational complexity, making them the least efficient in this comparison. For each data point n, a transformer requires the computation of the key, value, and query layers, contributing $3ck$, as well as the attention matrix with complexity kw^2, where w denotes the window size, c the number of input channels, and k the kernel size (i.e., number of output channels).

The Minkowski convolution, as defined by Choy. C. *et al.* [31], generalizes the standard convolution to sparse domains, with a core complexity of $ckw^{d=3}$ that scales with the number of query points m. To maintain equivalence with dense convolutional outputs, the query count m increases across stacked layers due to the propagation of kernel offsets around each point n:

$$m = n\left(1 + \frac{(L-1)^{w^d}}{\epsilon}\right), \tag{26}$$

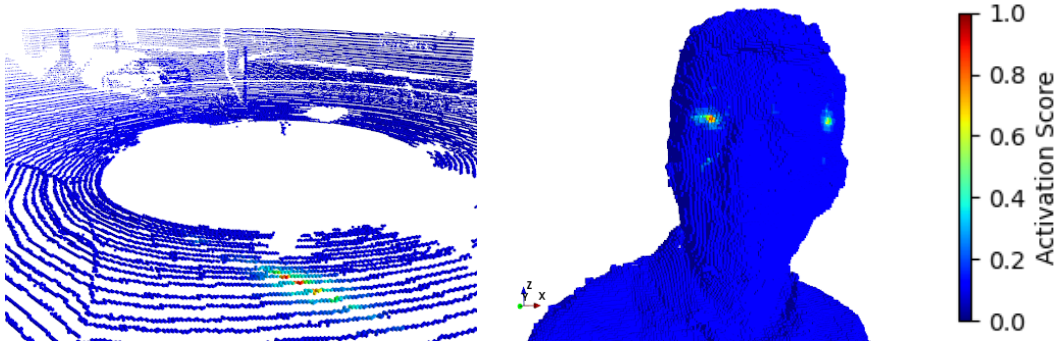
where L is the number of layers and ϵ is a correction factor accounting for overlapping receptive fields. This factor ϵ tends to be smaller for dense point clouds (due to more overlap) and larger for sparse point clouds.

 doi:10.20944/preprints202601.1367.v1

In contrast, the theoretical complexity of our multi-index convolution, given by $n(ckwi)$, is substantially lower than that of Minkowski convolution, since $w(i=4) < w^{d=3}$, where i denotes the number of applied index orders I and n remains constant. However, despite this theoretical advantage, our current implementation exhibits a runtime roughly ten times slower than sparse Minkowski-based models. We attribute this discrepancy to non-optimized index reordering operations, which cause extended GPU idle periods during the rearrangement of feature vectors F across index sets I. Ongoing optimization efforts aim to minimize these bottlenecks and further improve practical efficiency.

7.2. Perceptive Field

Figure 11 illustrates the theoretical attention score of the spatial receptive field generated by our multi-index convolution, using a gaussian filter. This visualization demonstrates how the proposed method effectively bridges large gaps in sparse point clouds, enabling better spatial context understanding in samples as in the SemanticKITTI dataset. On the other hand, this method occasionally extends beyond the object's convex hull, unintentionally capturing points from the backside. Hence, the spatial compactness of this method is not guaranteed and intuitively highlights a limitation when applied to dense or closed-surface geometries as in samples from the MVUB [24] dataset. Nonetheless, we assume that the learned filters of our model should easily detect the un-similarity between these far distant voxels and lower their activation score.



(a) (b) (c)

Figure 11. Theoretical activation score of the spatial receptive field generated by our multi-index convolution. **(a)** illustrates the receptive field on a sample from the SemanticKITTI dataset. **(b)** illustrates the receptive field on a sample from the MVUB dataset.

7.3. Ablation Study

The ablation study in Table 5 disentangles the contributions of the major components introduced in MIC-OPCCv2. Beginning with the baseline configuration of MIC-OPCCv1, which performs *octant-based* occupancy estimation using sub-modules tuned to different octree depths, we progressively activate the proposed design elements from Section 4 and measure their impact relative to the G-PCCv14 anchor.

Table 5. Ablation study of the proposed methods from Section 4. All BD-Rates are relative to G-PCCv14.

Method	**MIC-OPCCv1**					**MIC-OPCCv2**
octet estimation	✓					
voxel estimation		✓	✓	✓	✓	✓
Blender & Distiller			✓	✓	✓	✓
Sequential Grouping				✓		
Progressive Grouping					✓	✓
Sub-Modules	✓					✓
Gain^{-1}	20.2%	21.8%	25.4%	28.1%	28.6%	30.6%
Decoding Time (s)	2.2	2.8	2.9	5.4	5.3	5.1

 doi:10.20944/preprints202601.1367.v1

Switching from *octant estimation* to *voxel-wise occupancy estimation* yields the first substantial improvement, increasing the BD-Rate gain from 20.2% to 21.8%. This confirms that predicting voxel occupancy directly provides a more expressive supervision signal and enables more effective context modeling.

Introducing the *Blender & Distiller* architecture leads to a further gain of 25.4%. This demonstrates the value of multi-scale residual aggregation and the voting-based distillation scheme for stabilizing predictions across octree levels with varying entropy characteristics.

Adding a *Sequential Grouping* strategy – similar to conventional autoregressive decoding – pushes the gain to 28.1%, but also increases decoding time from roughly 3 seconds to over 5 seconds per frame. This suggests that a naïve sequential ordering improves R-D performance but at the expense of latency.

Replacing sequential grouping with the more efficient *Progressive Grouping* strategy produces comparable gain (28.6%), but avoids additional runtime penalties. Progressive grouping leverages partial parallelism while preserving relevant context among voxel candidates, making it the preferred strategy.

Finally, combining progressive grouping with the *Sub-Module* design – allocating depth-specific parameterization across octree levels – yields the full MIC-OPCCv2 model. This configuration achieves the highest gain of 30.6% while maintaining a decoding time of approximately 5.1 seconds, representing the best trade-off between compression efficiency and computational cost among all tested variants.

Overall, the ablation study highlights that each proposed component contributes meaningfully to performance, with the largest gains arising from voxel estimation, the Blender & Distiller architecture, and the progressive grouping strategy. Their combination enables MIC-OPCCv2 to substantially outperform the MIC-OPCCv1 baseline while preserving a manageable decoding complexity.

8. Conclusions and Future Work

In this paper, we introduced MIC-OPCCv2, a fast and lightweight neural entropy model for octree-based point cloud compression. The proposed framework employs a novel *multi-index convolution* mechanism to efficiently extract spatial context features from sparse and dense 3D data, avoiding the computational cost of transformer or full 3D convolution architectures. By combining this convolution strategy with the *Blender & Distiller* network and a semi-autoregressive grouping mechanism, MIC-OPCCv2 achieves strong R-D performance with significantly reduced decoding latency and memory footprint.

Experimental evaluations demonstrate that MIC-OPCCv2 surpasses traditional codecs such as G-PCC and achieves comparable or better compression efficiency than recent neural methods, including UniPCGC and ECM-OPCC, while decoding up to $\times 4$ faster. The model generalizes well across multiple datasets – handling both sparse LiDAR scans and dense object reconstructions – confirming its adaptability and robustness to varying data distributions. Furthermore, the modular nature of MIC-OPCCv2 allows flexible trade-offs between model complexity, memory consumption, and runtime, making it more suitable for deployment in systems such as autonomous vehicles, robotics, and 3D mapping.

Despite these promising results, several avenues for future research remain. First, the current model relies on fixed index permutations for spatial perception; dynamic or learned indexing strategies could further improve contextual sensitivity and adaptivity to varying geometries. Second, extending the entropy model to jointly handle geometry and attribute compression (e.g., color, intensity, or semantic labels) could provide an integrated and unified point cloud codec. Third, incorporating hardware-aware optimization – such as quantization-aware training or sparse tensor acceleration – may further enhance real-time decoding. Finally, exploring hybrid probabilistic modeling approaches, combining multi-index convolution with lightweight attention or diffusion-based priors, could yield further improvements in compression efficiency and reconstruction fidelity.

In summary, MIC-OPCCv2 establishes a new balance between compression efficiency, decoding speed, and computational scalability. It demonstrates that a carefully designed convolutional structure

 doi:10.20944/preprints202601.1367.v1

can outperform more complex architectures, providing an effective and practical solution for next-generation 3D data transmission and storage.

Funding: This work is financially supported in part (project number: 113UC2N006) by the Co-creation Platform of the Industry Academia Innovation School, NYCU, under the framework of the National Key Fields Industry-University Cooperation and Skilled Personnel Training Act, from the Ministry of Education (MOE) and industry partners in Taiwan.

Data Availability Statement: Source code and pre-trained models are published on https://github.com/bugerry87/mic-opcc

Acknowledgments: This work is also supported in part by the National Science and Technology Council (NSTC), Taiwan R.O.C. projects with grants 113-2640-E-A49-013, 113-2634-F-A49-004, 114-2640-E-A49-015, 114-2218-E-002-007, and 114-2218-E-A49-020

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

BAC	Binary Arithmetic Coder
BD-Rate	Bjøntegaard Delta Rate
ECM-OPCC	Efficient Context Model for Octree-based Point Cloud Compression
EHEM	Efficient Hierarchical Entropy Model for Learned Point Cloud Compression
FOC	Fast Octree Coding
G-PCC	Geometry-based Point Cloud Compression
MIC-OPCC	Multi-Index Convolution for Octree-based Point Cloud Compression
MLP	Multi Layer Perceptron
MPEG	Moving Picture Experts Group
MVUB	Microsoft's Voxelized Upper Bodies
NNOC	Neural Network Modeling of Probabilities for Coding the Octree Representation of Point Clouds
PCGC	Point Cloud Geometry Compression
PSNR	Peak Signal to Noise Ratio
R-D	Rate-Distortion
RENO	Real-Time Neural Compression for 3D LiDAR Point Clouds
RIDDLE	Lidar Data Compression with Range Image Deep Delta Encoding
UniPCGC	Towards Practical Point Cloud Geometry Compression via an Efficient Unified Approach
V-PCC	Video-based Point Cloud Compression

References

1. Meng, H.; Lu, H. A Survey of Deep Learning Technology in Visual SLAM. In Proceedings of the 2024 International Wireless Communications and Mobile Computing (IWCMC), 2024, pp. 0037–0042. https://doi.org/10.1109/IWCMC61514.2024.10592584.
2. Bhattacharyya, C.; Kim, S. Survey and Performance Analysis on Point Cloud Classification Models. In Proceedings of the 2023 23rd International Conference on Control, Automation and Systems (ICCAS), 2023, pp. 254–258. https://doi.org/10.23919/ICCAS59377.2023.10316922.
3. Roriz, R.; Silva, H.; Dias, F.; Gomes, T. A Survey on Data Compression Techniques for Automotive LiDAR Point Clouds. *Sensors* **2024**, *24*. https://doi.org/10.3390/s24103185.
4. Google. Draco 3D Graphics Compression. https://github.com/google/draco, 2017. accessed: 2024.
5. Li, G.; Gao, W.; Gao, W., MPEG Geometry-Based Point Cloud Compression (G-PCC) Standard. In *Point Cloud Compression: Technologies and Standardization*; Springer Nature Singapore: Singapore, 2024; pp. 135–165. https://doi.org/10.1007/978-981-97-1957-0_7.
6. Nguyen, D.T.; Quach, M.; Valenzise, G.; Duhamel, P. Learning-Based Lossless Compression of 3D Point Cloud Geometry. In Proceedings of the ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 4220–4224. https://doi.org/10.1109/ICASSP39728.2021.9414763.
7. Nguyen, D.T.; Quach, M.; Valenzise, G.; Duhamel, P. Multiscale deep context modeling for lossless point cloud geometry compression. In Proceedings of the 2021 IEEE International Conference on Multimedia Expo Workshops (ICMEW), 2021, pp. 1–6. https://doi.org/10.1109/ICMEW53276.2021.9455990.

 doi:10.20944/preprints202601.1367.v1

8. Wang, J.; Ding, D.; Li, Z.; Feng, X.; Cao, C.; Ma, Z. Sparse Tensor-based Multiscale Representation for Point Cloud Geometry Compression, 2021, [arXiv:cs.CV/2111.10633]. to be published.
9. Wang, K.; Gao, W. UniPCGC: Towards Practical Point Cloud Geometry Compression via an Efficient Unified Approach. *Proceedings of the AAAI Conference on Artificial Intelligence* **2025**, *39*, 12721–12729. https://doi.org/10.1609/aaai.v39i12.33387.
10. Huang, L.; Wang, S.; Wong, K.; Liu, J.; Urtasun, R. OctSqueeze: Octree-Structured Entropy Model for LiDAR Compression. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 1310–1320. https://doi.org/10.1109/CVPR42600.2020.00139.
11. Fu, C.; Li, G.; Song, R.; Gao, W.; Liu, S. OctAttention: Octree-Based Large-Scale Contexts Model for Point Cloud Compression. *Proceedings of the AAAI Conference on Artificial Intelligence* **2022**, *36*, 625–633. https://doi.org/10.1609/aaai.v36i1.19942.
12. Jin, Y.; Zhu, Z.; Xu, T.; Lin, Y.; Wang, Y. ECM-OPCC: Efficient Context Model for Octree-Based Point Cloud Compression. In Proceedings of the ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2024, pp. 7985–7989. https://doi.org/10.1109/ICASSP48485.2024.10446374.
13. Song, R.; Fu, C.; Liu, S.; Li, G. Efficient Hierarchical Entropy Model for Learned Point Cloud Compression. In Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023, pp. 14368–14377. https://doi.org/10.1109/CVPR52729.2023.01381.
14. Que, Z.; Lu, G.; Xu, D. VoxelContext-Net: An Octree Based Framework for Point Cloud Compression. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, pp. 6042–6051.
15. You, K.; Chen, T.; Ding, D.; Asif, M.S.; Ma, Z. Reno: Real-time neural compression for 3d lidar point clouds. In Proceedings of the Proceedings of the Computer Vision and Pattern Recognition Conference, 2025, pp. 22172–22181.
16. Baulig, G.; Guo, J.I. MIC-OPCC: Multi-Indexed Convolution Model for Octree Point Cloud Compression. In Proceedings of the 2025 Data Compression Conference (DCC), 2025, pp. 360–360. https://doi.org/10.1109/DCC62719.2025.00048.
17. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778. https://doi.org/10.1109/CVPR.2016.90.
18. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Proceedings of the Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015; Navab, N.; Hornegger, J.; Wells, W.M.; Frangi, A.F., Eds., Cham, 2015; pp. 234–241.
19. Behley, J.; Garbade, M.; Milioto, A.; Quenzel, J.; Behnke, S.; Stachniss, C.; Gall, J. SemanticKitti: A dataset for semantic scene understanding of lidar sequences. In Proceedings of the Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 9297–9307.
20. Fong, W.K.; Mohan, R.; Hurtado, J.V.; Zhou, L.; Caesar, H.; Beijbom, O.; Valada, A. Panoptic nuScenes: A Large-Scale Benchmark for LiDAR Panoptic Segmentation and Tracking. *arXiv arXiv:2109.03805* **2021**.
21. Sun, P.; Kretzschmar, H.; Dotiwalla, X.; Chouard, A.; Patnaik, V.; Tsui, P.; Guo, J.; Zhou, Y.; Chai, Y.; Caine, B.; et al. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.
22. Pandey, G.; McBride, J.R.; Eustice, R.M. Ford campus vision and lidar data set. *International Journal of Robotics Research* **2011**, *30*, 1543–1552.
23. Eugene, d.; Bob, H.; Taos, M.; Philip, A.C. 8i Voxelized Full Bodies - A Voxelized Point Cloud Dataset, 2017.
24. Charles, L.; Qin, C.; Sergio, O.E.; Philip, A.C. Microsoft voxelized upper bodies - a voxelized point cloud dataset, 2016.
25. Bentley, J.L. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* **1975**, *18*, 509–517. https://doi.org/10.1145/361002.361007.
26. Meagher, D. Geometric modeling using octree encoding. *Computer Graphics and Image Processing* **1982**, *19*, 129–147. https://doi.org/https://doi.org/10.1016/0146-664X(82)90104-6.
27. Zhou, X.; Qi, C.R.; Zhou, Y.; Anguelov, D. RIDDLE: Lidar Data Compression with Range Image Deep Delta Encoding. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 17191–17200. https://doi.org/10.1109/CVPR52688.2022.01670.
28. Li, G.; Gao, W.; Gao, W., MPEG Video-Based Point Cloud Compression (V-PCC) Standard. In *Point Cloud Compression: Technologies and Standardization*; Springer Nature Singapore: Singapore, 2024; pp. 199–218. https://doi.org/10.1007/978-981-97-1957-0_9.

 doi:10.20944/preprints202601.1367.v1

29. van den Oord, A.; Kalchbrenner, N.; Vinyals, O.; Espeholt, L.; Graves, A.; Kavukcuoglu, K. Conditional Image Generation with PixelCNN Decoders. *CoRR* **2016**, *abs/1606.05328*, [1606.05328].
30. Kaya, E.C.; Tabus, I. Neural Network Modeling of Probabilities for Coding the Octree Representation of Point Clouds, 2021, [arXiv:cs.CV/2106.06482].
31. Choy, C.; Gwak, J.; Savarese, S. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 3070–3079. https://doi.org/10.1109/CVPR.2019.00319.
32. Tang, H.; Liu, Z.; Zhao, S.; Lin, Y.; Lin, J.; Wang, H.; Han, S. Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution. *CoRR* **2020**, *abs/2007.16100*, [2007.16100].
33. Nguyen, D.T.; Kaup, A. Learning-Based Lossless Point Cloud Geometry Coding Using Sparse Tensors. In Proceedings of the 2022 IEEE International Conference on Image Processing (ICIP), 2022, pp. 2341–2345. https://doi.org/10.1109/ICIP46576.2022.9897827.
34. Shannon, C.E. A mathematical theory of communication. *The Bell System Technical Journal* **1948**, *27*, 379–423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x.
35. Witten, I.H.; Neal, R.M.; Cleary, J.G. Arithmetic Coding for Data Compression. *Commun. ACM* **1987**, *30*, 520–540. https://doi.org/10.1145/214762.214771.
36. Huffman, D.A. A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE* **1952**, *40*, 1098–1101. https://doi.org/10.1109/JRPROC.1952.273898.
37. Ibtehaz, N.; Rahman, M.S. MultiResUNet : Rethinking the U-Net architecture for multimodal biomedical image segmentation. *Neural Networks* **2020**, *121*, 74–87. https://doi.org/https://doi.org/10.1016/j.neunet.2019.08.025.
38. Schwarz, S.; Preda, M.; Baroncini, V.; Budagavi, M.; Cesar, P.; Chou, P.A.; Cohen, R.A.; Krivokuća, M.; Lasserre, S.; Li, Z.; et al. Emerging MPEG Standards for Point Cloud Compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* **2019**, *9*, 133–148. https://doi.org/10.1109/JETCAS.2018.2885981.
39. Ballé, J.; Laparra, V.; Simoncelli, E.P. End-to-end Optimized Image Compression. *CoRR* **2016**, *abs/1611.01704*, [1611.01704].
40. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings; Bengio, Y.; LeCun, Y., Eds., 2015.
41. Biswas, S.; Liu, J.; Wong, K.; Wang, S.; Urtasun, R. MuSCLE: Multi Sweep Compression of LiDAR using Deep Entropy Models. *Advances in Neural Information Processing Systems (NeurIPS)* **2020**, *33*, 1–12.
42. Fan, T.; Gao, L.; Xu, Y.; Wang, D.; Li, Z. Multiscale Latent-Guided Entropy Model for LiDAR Point Cloud Compression. *IEEE Transactions on Circuits and Systems for Video Technology* **2023**, *33*, 7857–7869. https://doi.org/10.1109/TCSVT.2023.3276788.
43. Bjontegaard, G. Calculation of average PSNR differences between RD-curves. Technical report, ITU-T SG16/Q6 VCEG 13th meeting, Austin, TX, USA, 2001. Document VCEG-M33.