

# OctSqueeze: Octree-Structured Entropy Model for LiDAR Compression

Lila Huang<sup>1,2</sup> Shenlong Wang<sup>1,3</sup> Kelvin Wong<sup>1,3</sup> Jerry Liu<sup>1</sup> Raquel Urtasun<sup>1,3</sup>

<sup>1</sup>Uber Advanced Technologies Group <sup>2</sup>University of Waterloo <sup>3</sup> University of Toronto

{lila.huang, slwang, kelvin.wong, jerryl, urtasun}@uber.com

## Abstract

We present a novel deep compression algorithm to reduce the memory footprint of LiDAR point clouds. Our method exploits the sparsity and structural redundancy between points to reduce the bitrate. Towards this goal, we first encode the LiDAR points into an octree, a data-efficient structure suitable for sparse point clouds. We then design a tree-structured conditional entropy model that models the probabilities of the octree symbols to encode the octree into a compact bitstream. We validate the effectiveness of our method over two large-scale datasets. The results demonstrate that our approach reduces the bitrate by 10-20% at the same reconstruction quality, compared to the previous state-of-the-art. Importantly, we also show that for the same bitrate, our approach outperforms other compression algorithms when performing downstream 3D segmentation and detection tasks using compressed representations. Our algorithm can be used to reduce the onboard and offboard storage of LiDAR points for applications such as self-driving cars, where a single vehicle captures 84 billion points per day.

## 1. Introduction

In the past few decades, we have witnessed artificial intelligence revolutionizing robotic perception. Robots powered by these AI algorithms often utilize a plethora of different sensors to perceive and interact with the world. In particular, 3D sensors such as LiDAR and structured light cameras have proven to be crucial for many types of robots, such as self-driving cars, indoor rovers, robot arms, and drones, thanks to their ability to accurately capture the 3D geometry of a scene. These sensors produce a significant amount of data: a single Velodyne HDL-64 LiDAR sensor generates over 100,000 points per sweep, resulting in over 84 billion points per day. This enormous quantity of raw sensor data brings challenges to onboard and offboard storage as well as real-time communication. Hence, it is necessary to develop an efficient compression method for 3D point clouds.

Raw 3D point clouds are represented as unstructured  $n \times 3$  matrices at float precision. This uncompressed data

representation does not exploit the fact that the geometry of the scene is usually well structured. Prior works have approached point cloud compression by using data structures such as KD-trees [4] and octrees [19] to encode a point cloud's structure. Quantization is exploited to further reduce storage. However, there remains a massive quantity of redundant information hidden in these representations, such as repetitive local structures, planar surfaces, or object categories with a strong shape prior, such as cars and humans. In theory, this redundant information can be exploited during compression to reduce the bitrate even further. However, this has not yet been exploited to its full potential in point cloud compression.

The recent success of deep neural networks in image and video compression brings a new paradigm towards structured data compression for point clouds. These approaches typically contain three steps: 1) encode the data into a hidden representation through a convolutional neural network; 2) quantize the hidden features; and 3) learn an entropy model to reduce the bitstream further through entropy coding. The key to the learned entropy model is encoding context information to improve the predictability of a symbol's occurrence, which directly increases its compressibility. However, it is non-trivial to apply these deep compression algorithms directly on a LiDAR point cloud, as it is sparse and non-grid structured. Hence, there are two major challenges that we need to address: 1) What is a memory-efficient data structure to represent LiDAR while exploiting its sparsity? 2) How can we train a deep entropy model to encode the representation to bitstreams efficiently?

In this work, we propose a novel deep learning model for LiDAR point cloud compression. Our approach first exploits the efficient and self-adaptive octree structure to get an initial encoding of the raw point cloud. We then learn a tree-structured deep conditional entropy model over each intermediate node of the tree, incorporating both the prior and the context of the scene simultaneously to help predict the node symbols. The predicted probabilities from our learned entropy model are then passed to an entropy coder to encode the serialized symbols into the final bitstream.

We evaluate the performance of our approach over two

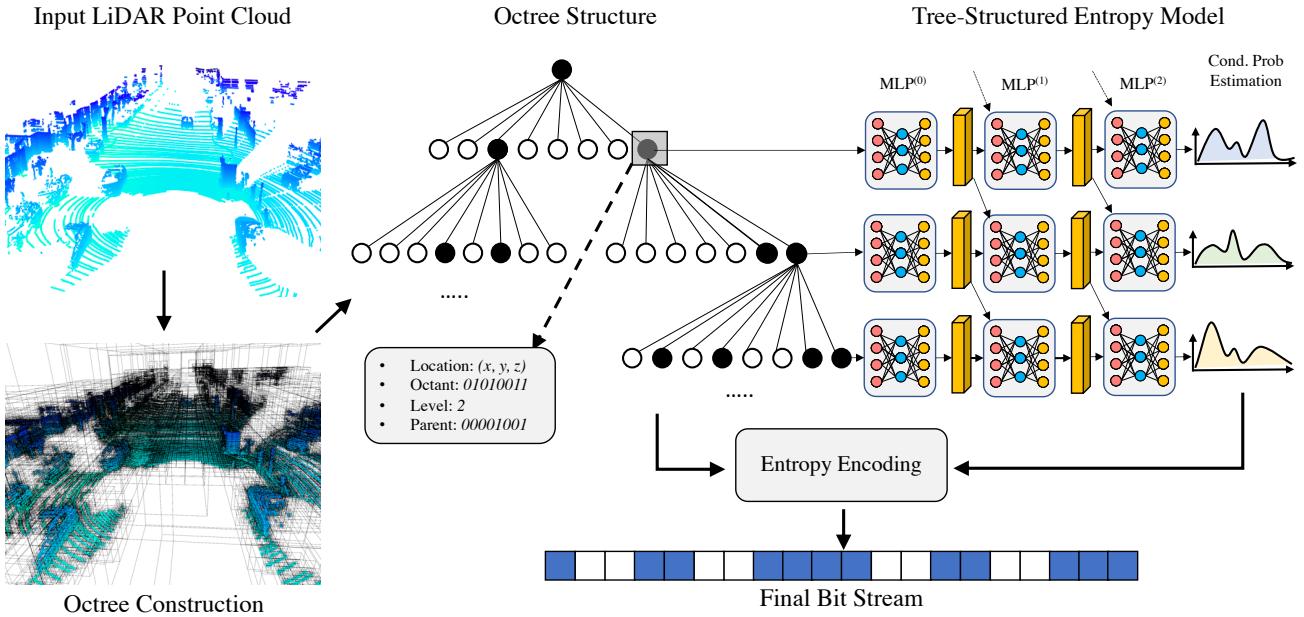


Figure 1: The overview of the proposed octree-structured entropy model for LiDAR point cloud compression. The input point cloud, received as a  $n \times 3$  float array, is quantized to  $k$  bits by scaling to  $[0, 2^k)$  and rounding down. An octree is constructed from the quantized point cloud. Each node is represented by an 8-bit occupancy symbol. We apply a tree-structured conditional entropy model on top of the octree to estimate the probability of each symbol conditioned on prior context. Finally, we use the estimated probability to encode the serialized symbols into the final compressed bitstream.

challenging LiDAR point cloud datasets comprising of complicated urban traffic scenes, namely the KITTI [3] and NorthAmerica datasets. Our results show that the proposed model outperforms all state-of-the-art methods in terms of both reconstruction quality and downstream task performance. At the same reconstruction quality, our bitrate is 10-20% lower than the previous state-of-the-art.

## 2. Related Work

### 2.1. Point Cloud Compression

Tree structures are the primary methods used in prior point cloud compression algorithms. Numerous approaches store data in an octree and perform entropy coding with hand-crafted entropy models such as adaptive histograms, parent context [6], and estimations based on planar approximations [31] or neighbour proximity [13]. To exploit temporal redundancies in point cloud streams, Kammerl *et al.* [14] encode the xor differences between successive octree representations and Mekuria *et al.* [20] use ICP to encode blocks with rigid transformations. Both methods use range coding with empirical histograms for entropy coding. The advantage of the octree structure is that it can model arbitrary point clouds in a hierarchical structure, which provide a natural progressive coding—if the octree is traversed in breadth-first order, then decoding can stop at any time; the longer the decoding, the finer the precision of the point cloud reconstruction. A related structure is utilized in Google’s open-source compression software *Draco* [8] which uses a KD-tree compression method [5]. All of the

above approaches do not leverage deep learning.

Besides tree structures, point clouds can be represented as regular voxel grids [27, 12]. These methods use voxel-based convolutional autoencoders which can learn surface representations of point clouds but struggles with large-scale sparse data. Moreover, since the geometry of a LiDAR scan can be represented by a panorama image with one channel for distance, point clouds can also be represented as range images, and compressed via image compression techniques. For example, Houshiair *et al.* [11] use conventional image compressors such as JPEG, PNG, and TIFF to compress LiDAR range images.

### 2.2. Deep Learning on Point Clouds

Inspired by recent successes in the image domain, researchers have developed a flurry of new deep learning methods for point cloud data. One class of methods uses deep convolutional neural networks to process voxel representations of the 3D point cloud [47, 18, 25, 52, 50, 49]. These approaches, however, require large memory footprints and thus induce a trade-off between input resolution and model capacity. To address this shortcoming, [28, 9] propose to use sparse operators on the point cloud’s voxel representation and [33, 23] propose to process 2D projections of the point cloud instead.

Another line of work tackles this problem by directly operating on the point cloud, thus leveraging its sparsity to sidestep this trade-off. PointNet [24] uses multi-layer perceptrons to extract features from individual points and then pools them into a global feature. As PointNet can-

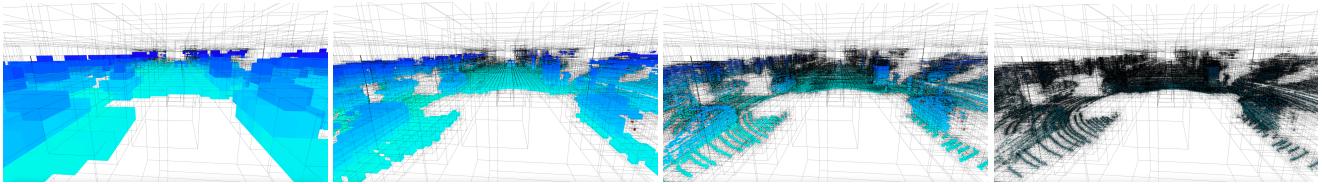


Figure 2: Construction of octree structures to represent a point cloud. Max depth of the octree (from left to right): 8, 10, 12, 14.

not capture local structures in the point cloud, a number of follow-up works have proposed to hierarchically aggregate local information [26, 42, 46, 48, 35]. These methods can be viewed as graph neural networks that operate on graphs defined by each point’s local neighbourhood; *e.g.*,  $k$ -nearest neighbors graph. Other possible graphs include KD-trees [15] and octrees [29, 41]. Inspired by the success of these graph-structured networks, we designed an entropy model that operates on an octree’s serialized byte streams but exploits its structure to encode contextual information.

### 2.3. Deep Image and Video Compression

The field of image and video compression is extensive and has been well-explored over the past few decades, ranging from lossless image formats (PNG, TIFF), to lossy image codecs (JPEG, BPG), to video codecs (AVC/H.264, HEVC/H.265). In recent years, there has been a rapid increase in learned image and video compression methods [37, 1, 2, 21, 34, 45, 30, 17, 10], which exploit concepts from traditional codecs and the power of deep neural networks. These approaches typically use deep convolutional autoencoders to apply nonlinear transforms to traditional components of the compression pipeline, from transform coding used in JPEG to motion compensation used in video codecs. Moreover, many approaches use separate neural nets to model the entropy of the image/video latent codes as a tight lower bound of the bitrate; this model is then used during entropy coding to losslessly compress the symbols into bits. Such approaches have included fully factorized models [1, 34], encoding “side information” as latent variables for entropy prediction [2, 22, 17] as well as using autoregressive models (*e.g.* PixelCNN [40]) to model pixel-level conditional distributions [21, 37, 10, 45]. Inspired by these approaches towards entropy modeling, we aim to apply these insights towards the compression of point clouds.

## 3. Octree-Structured Entropy Model

In this work we tackle the problem of *lossy compression* on 3D LiDAR point clouds. Our goal is to reduce the storage footprint of our encodings as much as possible while preserving reconstruction quality. Towards this goal, we propose a novel, octree-structured compression method using a *deep entropy model*.

Specifically, we firstly quantize and encode a LiDAR point cloud into an octree. Each node of the tree uses an

8-bit symbol to encode the occupancy of its children. We then serialize the octree into an intermediate, uncompressed bytestream of symbols. For each node, we select a set of context features that are available during decoding time. We then feed these context features into our tree-structured deep entropy model, which is trained to predict the probability of each symbol’s presence given the context input. These probabilities are then directly fed into arithmetic encoding with the symbol bytestream to produce the final bitstream, where the bitrate is approximately measured by the cross-entropy of these probabilities with the actual symbol distribution. Our overall approach is shown in Fig. 1.

### 3.1. Octree Structure

Two difficulties in LiDAR point cloud compression are the sparsity of the data and the lack of structure in a raw point cloud. Space-partitioning data structures, such as octrees and KD-trees, effectively provide a representation for 3D spatial data while keeping sparsity in mind, as their memory usage scales with the number of points in the cloud compared to voxel representations which scale with the cloud’s bounding volume. In addition, tree structures give implicit levels of detail which can be used for progressive decoding. We choose to use an octree as the base data structure for quantization due to its memory efficiency and ease of construction and serialization.

**Bit Representation:** An octree [19] stores point clouds by recursively partitioning the input space into equal octants and storing occupancy in a tree structure. Each intermediate node of the octree contains a 8-bit symbol to store the occupancy of its eight child nodes, with each bit corresponding to a specific child. Each leaf contains a single point and stores additional information to represent the position of the point relative to the cell corner. The size of leaf information is adaptive and depends on the level. An octree with  $k$  levels can store  $k$  bits of precision by keeping the last  $k - i$  bits of each of the  $(x, y, z)$  coordinates for a child on the  $i$ -th level of the octree. The resolution increases as the number of levels in the octree increases. The advantage of such a representation is twofold: firstly, only non-empty cells are further subdivided and encoded, which makes the data structure adapt to different levels of sparsity; secondly, the occupancy symbol per node is a tight bit representation. Fig. 2 shows the partial construction of the octree structure at different levels from a KITTI point cloud [7].

**Serialization:** Using a breadth-first or depth-first traversal, an octree can be serialized into two intermediate uncompressed bytestreams of occupancy codes and leaf-node offsets. The original tree can be completely reconstructed from these streams. We note that serialization is a lossless scheme in the sense that offsets and occupancy information are all exactly preserved. Thus the only lossy procedure is due to quantization during construction of the octree. Consequently, octree-based compression schemes are lossy up to this quantization error, which gives an upper bound on the distortion ratio.

We use the occupancy serialization format during our entropy coding stage, detailed in Sec. 3.2 and Sec. 3.3. During range decoding of a given occupancy code, we note that information such as node depth, parent occupancy, and spatial locations of the current octant are already known given prior knowledge of the traversal format. Hence we incorporate this information as a context  $\mathbf{c}_i$  for each node that we can use during entropy coding.

### 3.2. A Deep Entropy Model for Entropy Coding

The serialized occupancy bytestream of the octree can be further losslessly encoded into a shorter bit-stream through entropy coding. Entropy encoding is theoretically grounded in information theory. Specifically, an entropy model estimates the probability of occurrence of a given symbol; the probabilities can be adaptive given available context information. A key intuition behind entropy coding is that symbols that are predicted with higher probability can be encoded with fewer bits, achieving higher compression rates.

Existing entropy models on octree structures tend to either lack the ability to accurately represent the data in the case of adaptive histograms [31, 14], or require very long decoding times in the case of geometric predictions [13]. Moreover, these entropy models do not fully utilize the hierarchical octree structure to encode geometric priors of the scene to facilitate entropy prediction. Inspired by the success of using deep entropy models in image and video compression, we propose a deep network which models the entropy of the serialized octree data during entropy coding. Our approach extends prior methods in the sense that we better utilize the contextual information over the octree structure for prediction through an end-to-end learnable density estimation network.

**Formulation:** Given the sequence of occupancy 8-bit symbols  $\mathbf{x} = [x_1, x_2 \dots x_n]$ , the goal of an entropy model is to learn an estimated distribution  $q(\mathbf{x})$  such that it minimizes the cross-entropy with the actual distribution of the symbols  $p(\mathbf{x})$ :

$$H(p, q) = \mathbb{E}_{\mathbf{x} \sim p}[-\log_2 q(\mathbf{x})] \quad (1)$$

According to Shannon’s source coding theorem [32], the cross-entropy between  $q(\mathbf{x})$  and  $p(\mathbf{x})$  provides a tight lower bound on the bitrate achievable by arithmetic or range coding algorithms [43]; the better  $q(\mathbf{x})$  approximates  $p(\mathbf{x})$ , the lower the true bitrate. We thus train to minimize the cross-entropy loss between the model’s predicted distribution  $q$  and the distribution of training data.

**Entropy Model:** We now describe the formulation of our entropy model over the octree structure  $\mathbf{x}$ . We factorize  $q(\mathbf{x})$  into a product of conditional probabilities of each individual occupancy symbol  $x_i$  as follows:

$$q(\mathbf{x}) = \prod_i q_i(x_i | \mathbf{x}_{\text{an}(i)}, \mathbf{c}_i; \mathbf{w}). \quad (2)$$

where  $\mathbf{x}_{\text{an}(i)} = \{x_{\text{pa}(i)}, x_{\text{pa}(\text{pa}(i))}, \dots, x_{\text{pa}(\dots(\text{pa}(i)))}\}$  with  $|\mathbf{x}_{\text{an}(i)}| \leq K$  is the set of ancestor nodes of a given node  $i$ , up to a given order  $K$ , and  $\mathbf{w}$  is the weights parametrizing our entropy model. Here,  $\mathbf{c}_i$  is the context information that is available as prior knowledge during encoding/decoding of  $x_i$ , such as octant index, spatial location of the octant, level in the octree, parent occupancy, etc. These models take advantage of the tree structure to gather both the information from nodes at coarser levels and the context information available at the current node. Intuitively, conditioning on ancestor nodes can help to reduce the entropy for the current node prediction, since it is easier to predict the finer geometry structure at the current node when the coarse structure represented by ancestor nodes is already known. Context information such as location information help to reduce entropy even further by capturing the prior structure of the scene. For instance, in the setting of using LiDAR in the self-driving scenario, an occupancy node 0.5 meters above the LiDAR sensor is unlikely to be occupied.

**Architecture:** Our proposed entropy architecture models  $q_i(x_i | \mathbf{x}_{\text{an}(i)}, \mathbf{c}_i; \mathbf{w})$  by first extracting an independent contextual embedding for each  $x_i$ , and then performing progressive aggregation of contextual embeddings to incorporate ancestral information  $\mathbf{x}_{\text{an}(i)}$  for a given node.

For a given intermediate octree node  $x_i$ , the input context feature  $\mathbf{c}_i$  includes the node’s location, octant, level, and parent (see Fig. 1). Specifically, ‘location’ is the node’s 3D location encoded as a vector in  $\mathbb{R}^3$ , ‘octant’ is its octant index encoded as an integer in  $\{0, \dots, 7\}$ , ‘level’ is its depth encoded as an integer in  $\{0, \dots, \text{tree-depth}\}$ , and ‘parent’ is its parent’s 8-bit occupancy encoded as an integer in  $\{0, \dots, 255\}$ . We extract an independent deep feature for each node through a multi-layer perceptron (MLP) with the context feature  $\mathbf{c}_i$  as input:

$$\mathbf{h}_i^{(0)} = \text{MLP}^{(0)}(\mathbf{c}_i) \quad (3)$$

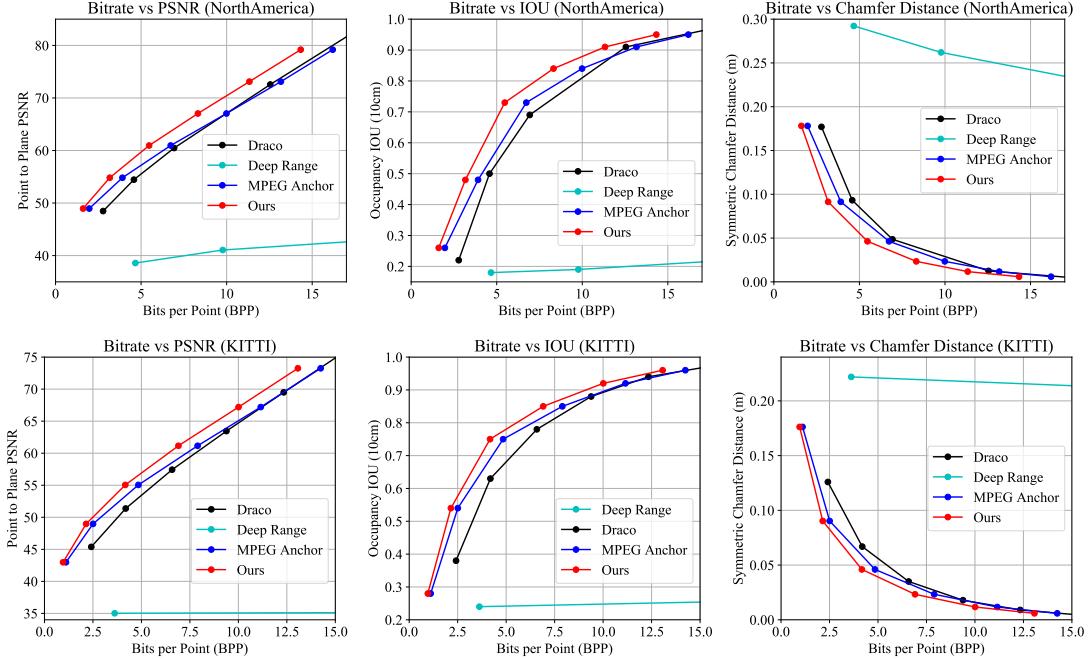


Figure 3: Quantitative results on NorthAmerica and KITTI. From left to right: point-to-plane PSNR, IOU, and Chamfer distance.

Then, starting with the feature  $\mathbf{h}_i^{(0)}$  for each node, we perform  $K$  aggregations between the current node feature and the feature of its parent. At iteration  $k$ , the aggregation can also be modeled as an MLP:

$$\mathbf{h}_i^{(k)} = \text{MLP}^{(k)}([\mathbf{h}_i^{(k-1)}, \mathbf{h}_{\text{pa}(i)}^{(k-1)}]) \quad (4)$$

where  $\mathbf{h}_{\text{pa}(i)}^{(k-1)}$  is the hidden feature of node  $i$ 's parent. For the root node, we consider its parent feature as all zero features for model consistency. The final output of our model is a linear layer on top of the  $K$ -th aggregated feature  $\mathbf{h}_i^{(k)}$ , producing a 256-dimensional softmax of probabilities for the 8-bit occupancy symbol of the given node:

$$q_i(\cdot | \mathbf{x}_{\text{an}(i)}, \mathbf{c}_i; \mathbf{w}) = g(\mathbf{h}_i^{(k)}) \quad (5)$$

Note that these aggregations only aggregate the node feature with that of its parent, never its child; the child input context is not available during sequential decoding. Moreover, each additional aggregation increases the receptive field of ancestral features by 1, and so the  $k$ -th aggregation has a receptive field of  $k$  ancestors. Fig. 1 depicts our proposed stacked entropy model with  $K = 3$ . In this figure, a model with  $K$  levels of aggregations predicts the probability of the current node  $x_i$  by considering the node feature itself as well as  $K - 1$  generations of the ancestor's feature.

In this sense, we can view our aggregation as conceptually similar to other autoregressive models, such as the “masked convolution” used PixelCNN [39] and “causal convolution” proposed in Wavenet [38]. Unlike previous work either on 2D grids or 1D sequences, our autoregressive model is applied along the octree traversal path from the root to each node.

**Detailed Architecture:** Here we discuss the detailed architecture of the each submodule of our stacked entropy model. The first MLP is a 5-layer MLP with 128 dimensional hidden features. All subsequent MLPs are 3-layer MLPs (with residual layers) with 128 dimensional hidden features. A final linear layer followed by a softmax is used to make the 256-way prediction. Every MLP is Linear + ReLU without normalization layers.

**Learning:** At training time, the full entropy model is trained end-to-end with the cross-entropy loss on each node:

$$\ell = - \sum_i \sum_j y_{i,j} \log q_{i,j} \quad (6)$$

where  $\mathbf{y}_i$  is the one-hot encoding of the ground-truth symbol at node  $i$ , and  $q_{i,j}$  is the predicted probability of symbol  $j$ 's occurrence at node  $i$ .

### 3.3. Entropy Coder

**Encoding:** At the encoding stage, we apply our model sequentially across different levels, from the root to leaves. Our proposed entropy model does not propagate information between nodes at the same level. Therefore, within each level, we are able to parallelize the computation for probability estimation. Afterwards, we losslessly compress the octree raw bit-stream using an entropy coding algorithm such as arithmetic coding. Our network determines the arithmetic coder's entropy model by predicting the categorical distribution (0 to 255) for each byte  $x_i$  in the sequence.

L	P	O	LL	Bitrate		
				Depth = 12	Depth = 14	Depth = 16
✓				3.91	9.99	16.21
✓	✓			3.86	9.79	15.91
✓	✓	✓		3.62	9.33	15.41
✓	✓	✓	✓	3.59	9.27	15.35
✓	✓	✓	✓	<b>3.48</b>	<b>8.91</b>	<b>14.97</b>

Table 1: Ablation study on input context features. L, P, O, and LL stand for the node’s octree level, its parent occupancy symbol, its octant index, and its spatial location respectively.

# Aggregations	Bitrate		
	Depth = 12	Depth = 14	Depth = 16
0	3.48	8.91	14.97
1	3.39	8.78	14.84
2	3.31	8.59	14.64
3	3.25	8.47	14.51
4	<b>3.17</b>	<b>8.32</b>	<b>14.33</b>

Table 2: Ablation study on the number of aggregations.

**Decoding:** To decode, the same entropy model is used in the arithmetic coder’s decoding algorithm. An octree is then built from the decompressed bitstream and used to reconstruct the point cloud. Due to the auto-regressive fashion of the entropy model, each node probability estimation is only dependent on itself and decoded node features at higher level of the octree. In addition, the octree is serialized in a breadth-first search fashion. As a result, given a node  $x_i$ , its ancestors in the octree  $\mathbf{x}_{\text{an}(i)}$  are decoded before  $x_i$ , making it feasible for the decoder to also decode  $x_i$ .

## 4. Experiments

In this section we validate the effectiveness of our proposed approach on two challenging real-world LiDAR datasets with drastically varying scenes. We compare our method against several state-of-the-art point cloud compression algorithms in terms of both reconstruction quality and their effects on downstream perception tasks.

### 4.1. Datasets

**NorthAmerica:** We collected a new internal dataset comprising of driving scenes from a wide variety of urban and highway environments in multiple cities/states across North America. From this dataset, we sampled 500K raw LiDAR scans collected by a Velodyne HDL-64 sensor to train our entropy model. No additional filtering or processing is applied to these LiDAR point clouds. For evaluation of reconstruction quality, we collected 472 snippets each containing 250 LiDAR scans. In addition, we also annotate these frames with 2D bird’s eye view bounding boxes for the vehicle, pedestrian, and motorbike classes, as well as per-point semantic labels for the vehicle, pedestrian, motorbike, road, and background classes. We use these labels for

evaluation on downstream perception tasks.

**KITTI:** To evaluate our method’s domain transfer capability, we show results on SemanticKITTI [3]—a public self-driving dataset containing 21351 scans with 4.5 billion points collected from a Velodyne HDL-64 sensor. As SemanticKITTI also contains dense point-wise labels from 25 classes, we also evaluate downstream task performance on this dataset. Note that there is a significant domain shift from our internal data to KITTI in terms of the scene layout as well as sensor configuration, such as sensor height, ego-occlusion, ray angles, etc.

### 4.2. Experimental Details

**Baselines:** Our baselines include two of the best off-the-shelf point cloud compression approaches, namely Google’s Draco encoder (‘Draco’) [8] and Mekuria *et al.*’s octree-based algorithm [20] which serves as the MPEG anchor (‘MPEG anchor’). In addition, we compare our method against a deep baseline model using a range image representation for the point cloud (‘Deep Range’). For the range image representation, we utilize the rolling shutter characteristics to convert each LiDAR scan from Euclidean coordinates to polar coordinates, and store it as a 2.5D range image. We then train the Ballé hyperprior model [2], a state-of-the-art image compression model, on these images. During decoding we reconstruct the 2.5D range image and convert it back to Euclidean point cloud.

**Implementation Details:** We train our entropy models on full 16-level octrees. Training a single model on the full 16-level octree allows for variable rate compression within the same model, since during test time, we can truncate the same octree over different levels to evaluate our models over different levels of quantization. Specifically, we evaluate our octree models with depths ranging from 11 to 16 to measure the bitrate-quality tradeoff. The quantization error ranges from 0.3cm to 9.75cm, and every decrement in tree height doubles this value.

Our entropy model is implemented in PyTorch and trained over 16 GPUs with the Adam optimizer. We use a learning rate of  $1e-4$  for 500K iterations.

### 4.3. Compression Metrics

**Reconstruction Quality:** To evaluate reconstruction quality, we use two families of metrics: distance and occupancy. A commonly used distance-based metric to evaluate point cloud similarity is the symmetric point-to-point Chamfer distance  $CD_{\text{sym}}$ . For a given GT point cloud  $\mathcal{P} = \{\mathbf{p}_i\}_{i=1,\dots,N}$  and reconstructed point cloud  $\hat{\mathcal{P}}$ :

$$CD(\mathcal{P}, \hat{\mathcal{P}}) = \frac{1}{|\mathcal{P}|} \sum_i \min_j \|\mathbf{p}_i - \hat{\mathbf{p}}_j\|_2 \quad (7)$$

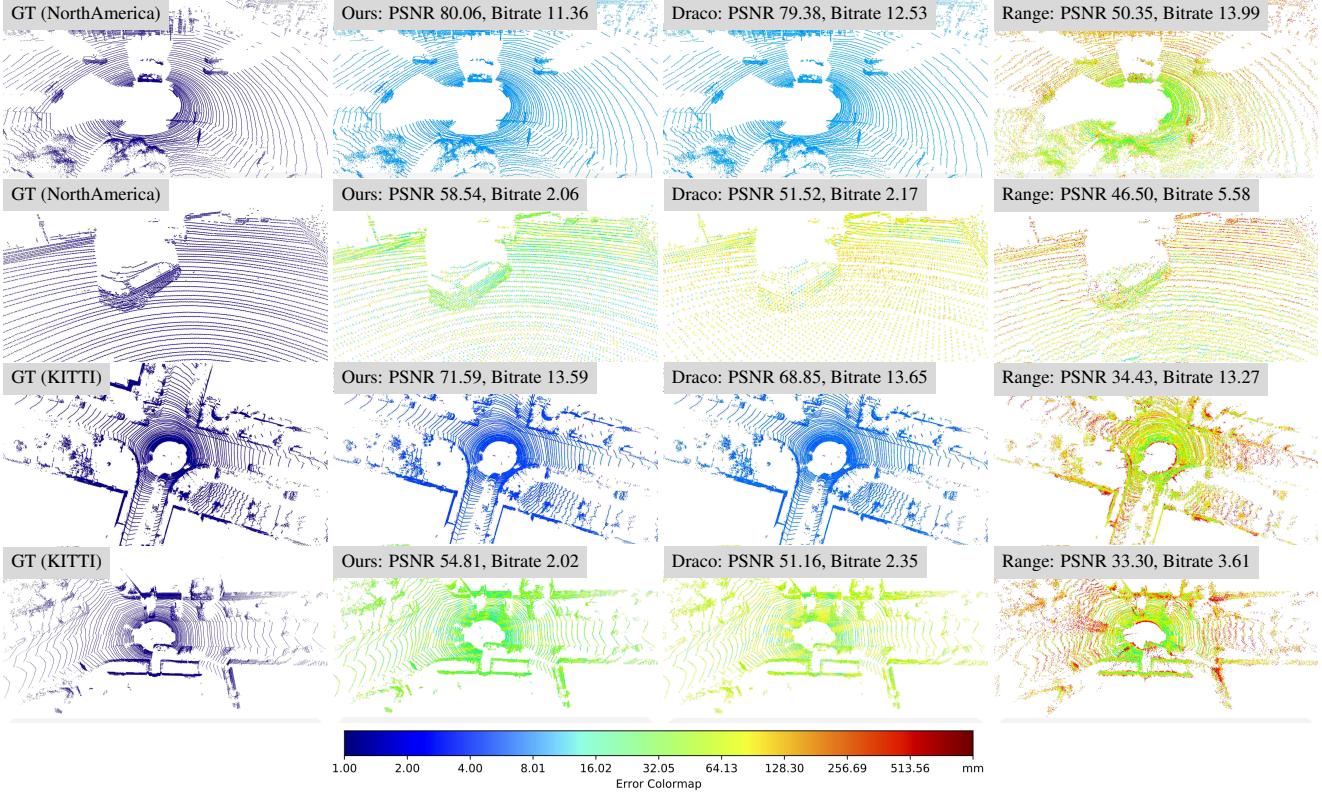


Figure 4: Qualitative results on NorthAmerica and KITTI. From left to right: Ground Truth, Ours, Draco, and Deep Range.

$$CD_{sym}(\mathcal{P}, \hat{\mathcal{P}}) = CD(\mathcal{P}, \hat{\mathcal{P}}) + CD(\hat{\mathcal{P}}, \mathcal{P}) \quad (8)$$

A second distance-based metric, symmetric point-to-plane PSNR<sub>sym</sub>, [36] accounts for point cloud resolution:

$$PSNR(\mathcal{P}, \hat{\mathcal{P}}) = 10 \log_{10} \frac{\max_i \|\mathbf{p}_i - \hat{\mathbf{p}}_i\|_2^2}{MSE(\mathcal{P}, \hat{\mathcal{P}})} \quad (9)$$

$$PSNR_{sym}(\mathcal{P}, \hat{\mathcal{P}}) = \min\{PSNR(\mathcal{P}, \hat{\mathcal{P}}), PSNR(\hat{\mathcal{P}}, \mathcal{P})\} \quad (10)$$

where  $MSE(\mathcal{P}, \hat{\mathcal{P}}) = \frac{1}{|\mathcal{P}|} \sum_i ((\hat{\mathbf{p}}_i - \mathbf{p}_i) \cdot \mathbf{n}_i)^2$  is the point-to-plane distance,  $\hat{\mathbf{p}}_i = \arg \min_{\mathbf{p} \in \hat{\mathcal{P}}} \|\mathbf{p}_i - \mathbf{p}\|_2^2$  is the closest point in  $\hat{\mathcal{P}}$  for each point  $\mathbf{p}_i$ , and  $\mathbf{n}_i$  is the normal at each  $\mathbf{p}_i$ . We estimate the normal  $\mathbf{n}_i$  at each point  $\mathbf{p}_i \in \mathcal{P}$  using the Open3D function `estimate_normals` with  $k = 12$  nearest neighbors [51].

**Occupancy Quality:** It is common practice to use LiDAR point clouds in voxelized form for perception tasks [16, 52, 49]. To reflect this, we computed occupancy-based metrics. In particular, we report the intersection-over-union (IOU) using  $0.2 \times 0.2 \times 0.1$  meter voxels:

$$IOU = \frac{TP}{TP + FP + FN} \quad (11)$$

where TP, FP, FN are the numbers of true positives, false positives, and false negatives in terms of voxel occupancy.

#### 4.4. Compression Results

**Quantitative Results on NorthAmerica:** We report the bitrate versus reconstruction quality metrics (PSNR, IOU, Chamfer) over all competing algorithms on the NorthAmerica dataset. As shown in Fig. 3, our method outperforms all previous state-of-the-art algorithms, with a 10-20% bitrate reduction over Draco and MPEG Anchor at the same reconstruction quality. All three methods significantly outperform the deep range image compression method. Note that since we use the same octree data structure, our approach has the same reconstruction quality as MPEG Anchor. However, our bitrate is much lower thanks to the deep entropy model. These results validate our proposed deep entropy model and our choice of an octree data structure to compress sparse LiDAR point clouds.

**Quantitative Results on KITTI:** In Fig. 3, we show the bitrate versus reconstruction quality metrics on KITTI. Although our model was trained using only data from NorthAmerica, it can still significantly outperform all competing algorithms, especially at lower bitrates.

**Qualitative Results:** Fig. 4 shows point cloud reconstructions on KITTI and NorthAmerica colored by reconstruction error. For fair comparison, we choose results from the competing algorithms that have been compressed at a similar bitrate rate. All cases show that our method and Draco

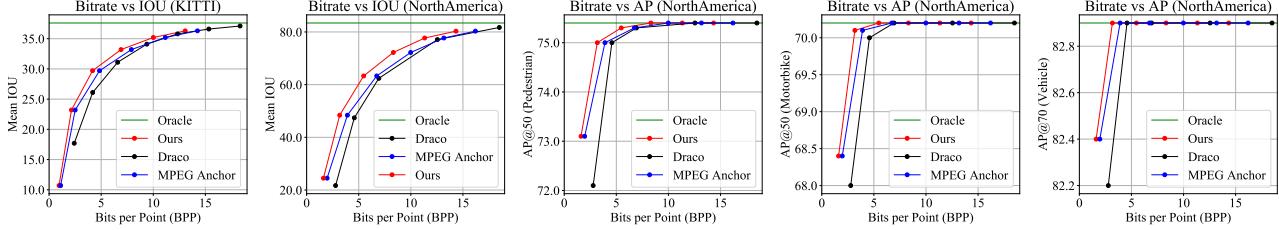


Figure 5: Quantitative results of downstream perception tasks. The leftmost two figures show IOU performance on semantic segmentation for KITTI and NorthAmerica respectively. The rightmost three figures show AP performance on object detection for NorthAmerica.

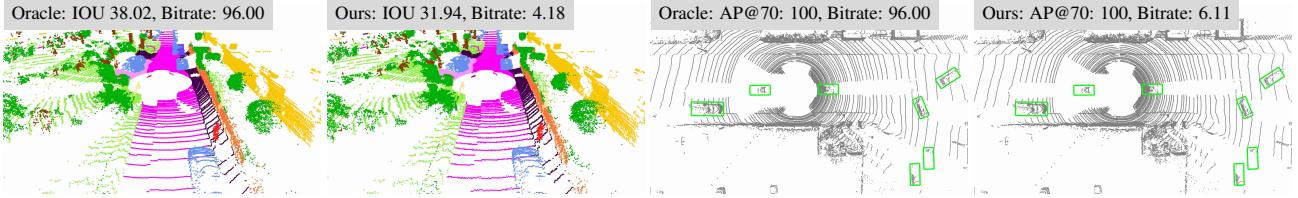


Figure 6: Qualitative results of semantic segmentation (right) and object detection (left).

give more faithful reconstructions than range image compression at comparable bitrates, as the range image reconstruction suffers from noise and errors at object boundaries as well as lower/upper LiDAR beam. At the same bitrate, our reconstruction quality is also better than Draco.

**Ablation Studies:** We perform ablation studies on the entropy model, both over the context features  $c_i$  as well as over the number of aggregations  $K$ . In Tab. 1, we ablate over context features by progressively incorporating the four features that we use: the node’s octree level, its parent occupancy symbol, its octant index, and its spatial location. Note that these ablations are performed without any aggregations ( $K = 0$ ), demonstrating the predictive power of context features alone. As shown in the table, we can see that gradually adding more context information consistently lowers the entropy of our encoding.

Next, we evaluate how the high-order ancestor information helps to predict the probability. We evaluate the proposed entropy model with different levels of aggregation,  $K = 0, \dots, 4$ , incorporating  $K$  levels of “ancestor” contexts. Tab. 2 show that in general, conducting more aggregations consistently improves the entropy of our model.

#### 4.5. Effects on Downstream Perception Tasks

Another important metric for compression is its effects on the performance of relevant downstream tasks. We quantify these effects for two fundamental perception tasks: semantic segmentation and object detection.

In our experiments, we evaluate the semantic segmentation and object detection models described in [44] over point clouds reconstructed from various compression schemes. Note that we train these perception models on uncompressed point clouds with detection and segmentation labels—for NorthAmerica, we use the training dataset

described in [44], and for KITTI, we use the official training dataset [3]. For semantic segmentation, we report mean intersection-over-union (IOU) computed using voxelized ground truth labels. For object detection, we report average precision (AP) at 50% IOU threshold for pedestrians and motorbikes, and 70% for vehicles.

As shown in Fig. 5 and Fig. 6, our method outperforms all competing baselines on both NorthAmerica and KITTI. Our method’s strength is particularly highlighted in semantic segmentation where preserving the fine-grained details of the point cloud is especially important. For example, at 5 bits-per-point, our method achieves a 5-10% improvement over Draco and MPEG for NorthAmerica. In object detection, our method consistently outperforms the baselines, albeit more slightly than in segmentation; this is due to the fact that the object detection model is already robust to a range of bitrates. Overall, these results attest to the performance of our method and help illustrate its effects on tasks relevant to many robotics applications.

## 5. Conclusion

We presented a novel LiDAR point cloud compression algorithm. Our method uses a deep tree-structured entropy model on an octree representation of the points that leverages available context information to reduce the entropy of each intermediate node. This entropy model exploits both the sparsity and structural redundancy between points to reduce the overall bitrate. We validate the effectiveness of our method over two large-scale datasets. The results suggest that our approach significantly reduces the bitrate compared against other competing algorithms at the same reconstruction quality. In addition, we demonstrate that our compressed representations achieve a lower error on downstream tasks than prior state-of-the-art work.

## References

- [1] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. End-to-end optimized image compression. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. 3
- [2] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. 3, 6
- [3] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jürgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 9296–9306. IEEE, 2019. 2, 6, 8
- [4] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, 1975. 1
- [5] Olivier Devillers and Pierre-Marie Gandon. Geometric compression for interactive transmission. In *IEEE Visualization 2000, October 8-13, 2000, Hilton Hotel, Salt Lake City, Utah, USA, Proceedings*, pages 319–326. IEEE Computer Society and ACM, 2000. 2
- [6] Diogo C. Garcia and Ricardo L. de Queiroz. Intra-frame context-based octree coding for point-cloud geometry. In *2018 IEEE International Conference on Image Processing, ICIP 2018, Athens, Greece, October 7-10, 2018*, pages 1807–1811. IEEE, 2018. 2
- [7] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pages 3354–3361. IEEE Computer Society, 2012. 3
- [8] Google. Draco 3d data compression. <https://github.com/google/draco>, 2017. 2, 6
- [9] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 9224–9232. IEEE Computer Society, 2018. 2
- [10] AmirHossein Habibian, Ties van Rozendaal, Jakub M. Tomczak, and Taco Cohen. Video compression with rate-distortion autoencoders. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 7032–7041. IEEE, 2019. 3
- [11] Hamidreza Houshiar and Andreas Nüchter. 3d point cloud compression using conventional image compression for efficient data transmission. In *XXV International Conference on Information, Communication and Automation Technologies, ICAT 2015, Sarajevo, Bosnia and Herzegovina, October 29-31, 2015*, pages 1–8. IEEE Computer Society, 2015. 2
- [12] Tianxin Huang and Yong Liu. 3d point cloud geometry compression on deep learning. In Laurent Amsaleg, Benoit Huet, Martha A. Larson, Guillaume Gravier, Hayley Hung, Chong-Wah Ngo, and Wei Tsang Ooi, editors, *Proceedings of the 27th ACM International Conference on Multimedia, MM 2019, Nice, France, October 21-25, 2019*, pages 890–898. ACM, 2019. 2
- [13] Yan Huang, Jingliang Peng, C.-C. Jay Kuo, and M. Gopi. A generic scheme for progressive point cloud coding. *IEEE Trans. Vis. Comput. Graph.*, 14(2):440–453, 2008. 2, 4
- [14] Julius Kammerl, Nico Blodow, Radu Bogdan Rusu, Suat Gedikli, Michael Beetz, and Eckehard G. Steinbach. Real-time compression of point cloud streams. In *IEEE International Conference on Robotics and Automation, ICRA 2012, 14-18 May, 2012, St. Paul, Minnesota, USA*, pages 778–785. IEEE, 2012. 2, 4
- [15] Roman Klokov and Victor S. Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 863–872. IEEE Computer Society, 2017. 3
- [16] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 12697–12705. Computer Vision Foundation / IEEE, 2019. 7
- [17] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Chunlei Cai, and Zhiyong Gao. DVC: an end-to-end deep video compression framework. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 11006–11015. Computer Vision Foundation / IEEE, 2019. 3
- [18] Daniel Maturana and Sebastian A. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015, Hamburg, Germany, September 28 - October 2, 2015*, pages 922–928. IEEE, 2015. 2
- [19] Donald Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(1):85, 1982. 1, 3
- [20] Rufael Mekuria, Kees Blom, and Pablo César. Design, implementation, and evaluation of a point cloud codec for tele-immersive video. *IEEE Trans. Circuits Syst. Video Techn.*, 27(4):828–842, 2017. 2, 6
- [21] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Conditional probability models for deep image compression. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 4394–4402. IEEE Computer Society, 2018. 3
- [22] Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Practical full resolution learned lossless image compression. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 10629–10638. Computer Vision Foundation / IEEE, 2019. 3

- [23] Gregory P. Meyer, Ankit Laddha, Eric Kee, Carlos Valdespi-Gonzalez, and Carl K. Wellington. Lasernet: An efficient probabilistic 3d object detector for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 12677–12686. Computer Vision Foundation / IEEE, 2019. [2](#)
- [24] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 77–85. IEEE Computer Society, 2017. [2](#)
- [25] Charles Ruizhongtai Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 5648–5656. IEEE Computer Society, 2016. [2](#)
- [26] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5099–5108, 2017. [3](#)
- [27] Maurice Quach, Giuseppe Valenzise, and Frédéric Dufaux. Learning convolutional transforms for lossy point cloud geometry compression. In *2019 IEEE International Conference on Image Processing, ICIP 2019, Taipei, Taiwan, September 22-25, 2019*, pages 4320–4324. IEEE, 2019. [2](#)
- [28] Mengye Ren, Andrei Pokrovsky, Bin Yang, and Raquel Urtasun. Sbnet: Sparse blocks network for fast inference. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8711–8720. IEEE Computer Society, 2018. [2](#)
- [29] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 6620–6629. IEEE Computer Society, 2017. [3](#)
- [30] Oren Rippel, Sanjay Nair, Carissa Lew, Steve Branson, Alexander G. Anderson, and Lubomir D. Bourdev. Learned video compression. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 3453–3462. IEEE, 2019. [3](#)
- [31] Ruwen Schnabel and Reinhard Klein. Octree-based point-cloud compression. In Mario Botsch, Baoquan Chen, Mark Pauly, and Matthias Zwicker, editors, *Symposium on Point Based Graphics, Boston, Massachusetts, USA, 2006. Proceedings*, pages 111–120. Eurographics Association, 2006. [2, 4](#)
- [32] Claude E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27(3):379–423, 1948. [4](#)
- [33] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 945–953. IEEE Computer Society, 2015. [2](#)
- [34] Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. Lossy image compression with compressive autoencoders. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. [3](#)
- [35] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 6410–6419. IEEE, 2019. [3](#)
- [36] Dong Tian, Hideaki Ochiaimizu, Chen Feng, Robert A. Cohen, and Anthony Vetro. Geometric distortion metrics for point cloud compression. In *2017 IEEE International Conference on Image Processing, ICIP 2017, Beijing, China, September 17-20, 2017*, pages 3460–3464. IEEE, 2017. [7](#)
- [37] George Toderici, Damien Vincent, Nick Johnston, Sung Jin Hwang, David Minnen, Joel Shor, and Michele Covell. Full resolution image compression with recurrent neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5435–5443. IEEE Computer Society, 2017. [3](#)
- [38] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016*, page 125. ISCA, 2016. [5](#)
- [39] Aäron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 4790–4798, 2016. [5](#)
- [40] Aäron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 1747–1756. JMLR.org, 2016. [3](#)
- [41] Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-CNN: octree-based convolutional neural networks for 3d shape analysis. *ACM Trans. Graph.*, 36(4):72:1–72:11, 2017. [3](#)
- [42] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *2018 IEEE Conference on*

*Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 2589–2597. IEEE Computer Society, 2018. 3

- [43] Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic coding for data compression. *Commun. ACM*, 30(6):520–540, 1987. 4
- [44] Kelvin Wong, Shenlong Wang, Mengye Ren, Ming Liang, and Raquel Urtasun. Identifying unknown instances for autonomous driving. *CoRR*, abs/1910.11296, 2019. 8
- [45] Chao-Yuan Wu, Nayan Singhal, and Philipp Krähenbühl. Video compression through image interpolation. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VIII*, volume 11212 of *Lecture Notes in Computer Science*, pages 425–440. Springer, 2018. 3
- [46] Wenxuan Wu, Zhongang Qi, and Fuxin Li. Pointconv: Deep convolutional networks on 3d point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 9621–9630. Computer Vision Foundation / IEEE, 2019. 3
- [47] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Lin-guang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1912–1920. IEEE Computer Society, 2015. 2
- [48] Yuwen Xiong, Mengye Ren, Renjie Liao, Kelvin Wong, and Raquel Urtasun. Deformable filter convolution for point cloud reasoning. *CoRR*, abs/1907.13079, 2019. 3
- [49] Bin Yang, Wenjie Luo, and Raquel Urtasun. PIXOR: real-time 3d object detection from point clouds. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 7652–7660. IEEE Computer Society, 2018. 2, 7
- [50] Chris Zhang, Wenjie Luo, and Raquel Urtasun. Efficient convolutions for real-time semantic segmentation of 3d point clouds. In *2018 International Conference on 3D Vision, 3DV 2018, Verona, Italy, September 5-8, 2018*, pages 399–408. IEEE Computer Society, 2018. 2
- [51] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 7
- [52] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 4490–4499. IEEE Computer Society, 2018. 2, 7

# Supplementary Material – OctSqueeze: Octree-Structured Entropy Model for LiDAR Compression

Lila Huang<sup>1,2</sup> Shenlong Wang<sup>2,3</sup> Kelvin Wong<sup>2,3</sup> Jerry Liu<sup>2</sup> Raquel Urtasun<sup>2,3</sup>

<sup>1</sup>University of Waterloo <sup>2</sup>Uber Advanced Technologies Group <sup>3</sup>University of Toronto

{lila.huang, slwang, kelvin.wong, jerryl, urtasun}@uber.com

## Abstract

In this supplementary material, we describe additional experimental results that further validate the efficacy of our proposed method. We also benchmark the runtime of our proposed method and demonstrate its ability to encode LiDAR point clouds in real-time. Moreover, we exhibit an extensive array of qualitative results on NorthAmerica and KITTI that compares our method against Draco in terms of reconstruction quality and downstream task performance.

## 1. Additional Ablation Studies

We conduct a more thorough analysis on our entropy model to validate our choice of model architecture and the feature set we use. In Sec. 1.1, we show that our model performs best when using  $K = 4$  levels of aggregation. Then, in Sec. 1.2, we demonstrate that our model’s performance improvements arise as a result of our hierarchical feature aggregation scheme, and not because of the increase in model capacity. Finally, in Sec. 1.3, we present an expanded ablation study on our input feature set at the best level of aggregations; *i.e.*,  $K = 4$ . All experiments are conducted on the NorthAmerica evaluation set.

### 1.1. Number of Aggregations

# Aggregations	Bitrate		
	Depth = 12	Depth = 14	Depth = 16
0	3.48	8.91	14.97
1	3.39	8.78	14.84
2	3.31	8.59	14.64
3	3.25	8.47	14.51
4	<b>3.17</b>	<b>8.32</b>	<b>14.33</b>
5	3.27	8.51	14.55

Table 1: Ablation study on the number of aggregations.

Tab. 1 extends Tab. 2 in the main paper with an additional row entry for  $K = 5$  aggregations. We found that  $K = 5$  aggregations performs worse than  $K = 4$  in terms of bitrate reduction, suggesting that our choice of  $K = 4$  aggregations in the main paper is best for our architecture.

### 1.2. Aggregation of Parental Context Features

We also investigate whether a model that does not aggregate parental context features can achieve similar bitrate reductions as our model with  $K = 4$  aggregations, holding all else equal. To perform this study, we trained an entropy model with the same architecture as our model with  $K = 4$  aggregations, except that each node takes in a copy of its own context feature in the aggregation stage, rather than that of its parent. Tab. 2 shows our results. Surprisingly, we found that not only did the model without parental aggregation perform worse than the one with aggregation, it also performed only as well as our

Parental Aggregations	K	Bitrate		
		Depth = 12	Depth = 14	Depth = 16
	0	3.48	8.91	14.97
✓	4 4	3.47 <b>3.17</b>	8.92 <b>8.32</b>	14.98 <b>14.33</b>

Table 2: We compare the performance of our entropy model with and without aggregating parental context features (bottom two rows). Both models have the same model capacity as one with  $K = 4$  aggregations. For completeness, we also show the performance of our model with  $K = 0$  aggregations.

original, smaller capacity model with  $K = 0$  aggregations! This result suggests that adding more layers to the network alone does not translate to performance gains. Moreover, it validates our design of a tree-structured entropy model that progressively incorporates parental information through aggregations.

### 1.3. Input Context Features

L	P	O	LL	Bitrate		
				Depth = 12	Depth = 14	Depth = 16
✓				3.86	9.79	15.91
✓	✓			3.44	8.89	14.94
✓	✓	✓		3.34	8.72	14.76
✓	✓	✓	✓	<b>3.17</b>	<b>8.32</b>	<b>14.33</b>

Table 3: Ablation study on input context features for our model with  $K = 4$  aggregations. L, P, O, and LL stand for the node’s octree level, its parent occupancy symbol, its octant index, and its spatial location respectively.

We conduct an ablation study on the input context features used by our entropy model at  $K = 4$  aggregations: the node’s octree level, its parent occupancy symbol, its octant index, and its spatial location. Note that in Tab. 1 of the main paper, we presented an analogous ablation study on a model with  $K = 0$  aggregations. In Tab. 3, we observe a similar decrease in bitrate as we increase the number of input context features. This result further corroborates our hypothesis that all four input context features contribute to the predictive power of our entropy model.

## 2. Additional Baselines

We conduct experiments comparing our compression method with two additional baselines. In Sec. 2.1, we experiment with a range view-based compression method that leverages the popular JPEG2000 image codec. Then in Sec. 2.2, we present results from our experiments with the voxel-based point cloud compression algorithm by Quach *et al.* [2].

### 2.1. JPEG Range Encoder

We compare against two baselines that use a range image representation of the input point cloud: Deep Range and JPEG Range. Deep Range is the range view-based method discussed in Sec. 4.2 of the main paper. In particular, given a LiDAR point cloud, we first construct a range image by converting it from Euclidean coordinates to polar coordinates, and then storing it as a 2.5D range image. Deep Range then uses a Ballé hyperprior model [1] to compress the 2.5D range image. In contrast, JPEG Range uses the popular JPEG2000 image codec to compress the 2.5D range image.

As shown in Fig. 1, Deep Range outperforms JPEG Range across all reconstruction quality metrics on both NorthAmerica and KITTI. This is a testament to the performance of deep learning-based methods for image compression. Moreover, as we alluded to in Sec. 4.4 of the main paper, our approach significantly outperforms both Deep Range and JPEG Range owing to its use of an octree data structure to represent the LiDAR point cloud and an octree-structured entropy model to compress it.

### 2.2. Deep Voxel Encoder

We additionally implemented the voxel-based point cloud compression algorithm by Quach *et al.* [2], consisting of a deep 3D convolutional autoencoder architecture with a fully-factorized entropy model inspired from [1]. We trained and evaluated these models on the NorthAmerica LiDAR point cloud dataset, voxelizing points at (0.25m, 0.25m, 1.0m) for

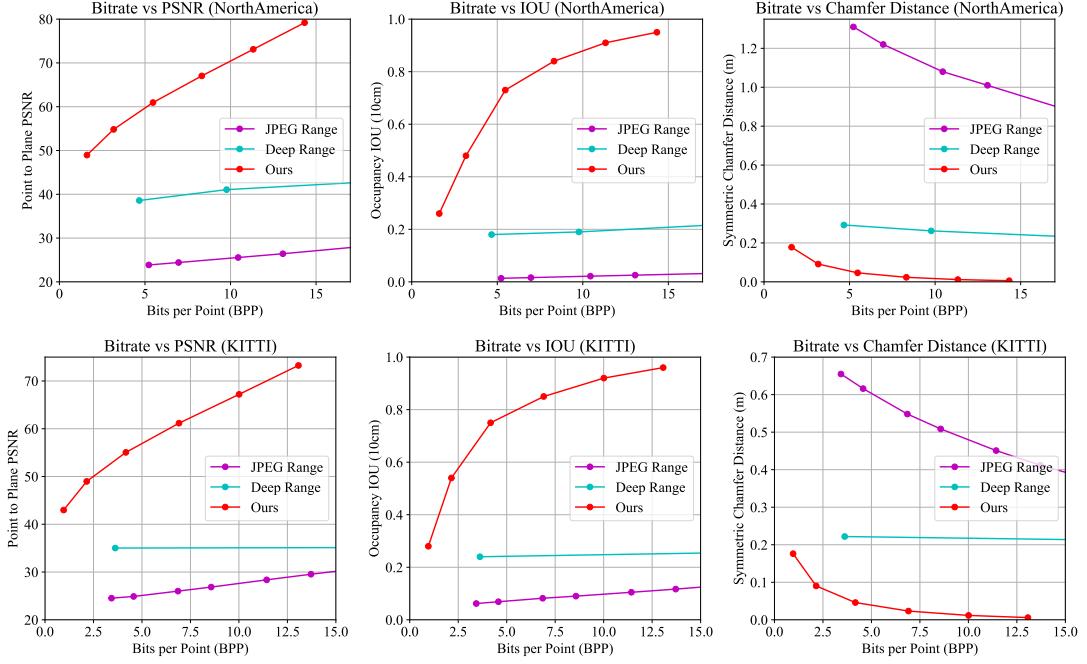


Figure 1: Quantitative results on NorthAmerica and KITTI. From left to right: point-to-plane PSNR, IOU, and Chamfer distance.

length, width, and depth dimensions respectively. Our best-performing reference model achieves a point-to-plane PSNR of 33.76 at a bitrate of 26.81—this performs much worse than the tree-based methods of Draco (PSNR: 48.47, bpp: 2.778) and our approach (PSNR: 48.95, bpp 1.61). The underperformance of the voxel-based compression method indicates that a dense voxel representation may not be the best fit for compressing LiDAR point clouds due to the inherent sparsity and high frequency information in this data.

### 3. Runtime

Depth	Encoding (ms)				Decoding (ms)
	Octree	Network	Range Coding	Total	
11	21.10	13.85	0.80	35.75	92.96
12	23.73	24.67	1.18	49.58	165.70
13	25.51	39.82	2.19	67.52	299.41
14	32.34	56.04	3.15	91.53	486.36
15	34.85	65.17	3.55	103.57	698.98
16	35.52	66.83	3.61	105.96	902.27

Table 4: Runtime of our model with  $K = 4$  aggregations (in milliseconds). ‘Depth’ is the maximum depth of the octree. ‘Octree’ is the time to build the octree; ‘Network’ the time to run our entropy model; and ‘Range Coding’ the time of range coding.

We benchmarked our approach on a workstation with an Intel Xeon E5-2687W CPU and a Nvidia GeForce GTX 1080 GPU. See Tab. 4 for the results. In our experiments, octree building and range coding were implemented in C++, and our entropy model was implemented in Python with PyTorch. Our approach achieves end-to-end encoding in real-time, meaning that our algorithm can be deployed in an online setting. Moreover, we believe there are many opportunities to speed up our research code significantly, especially in CPU/GPU I/O.

### 4. Additional Qualitative Results

We exhibit an extensive array of qualitative results that compare our method against Draco across a spectrum of bitrates. In Fig. 2 and 3, we show the reconstruction quality of our method versus Draco. Then, in Fig. 4 and 5, we show their respective downstream semantic segmentation performance. Finally, in Fig. 6, we show their respective downstream object

detection performance. As indicated in these figures, our model can attain better results than Draco at comparable—or even lower—bitrates.

## 5. Change Log

**ArXiv v2:** We corrected our definitions of the reconstruction metrics: the symmetric point-to-point Chamfer distance and the symmetric point-to-plane PSNR. We also corrected our estimates of OctSqueeze’s decoding runtime.

## References

- [1] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. [2](#)
- [2] Maurice Quach, Giuseppe Valenzise, and Frédéric Dufaux. Learning convolutional transforms for lossy point cloud geometry compression. In *2019 IEEE International Conference on Image Processing, ICIP 2019, Taipei, Taiwan, September 22-25, 2019*, pages 4320–4324. IEEE, 2019. [2](#)

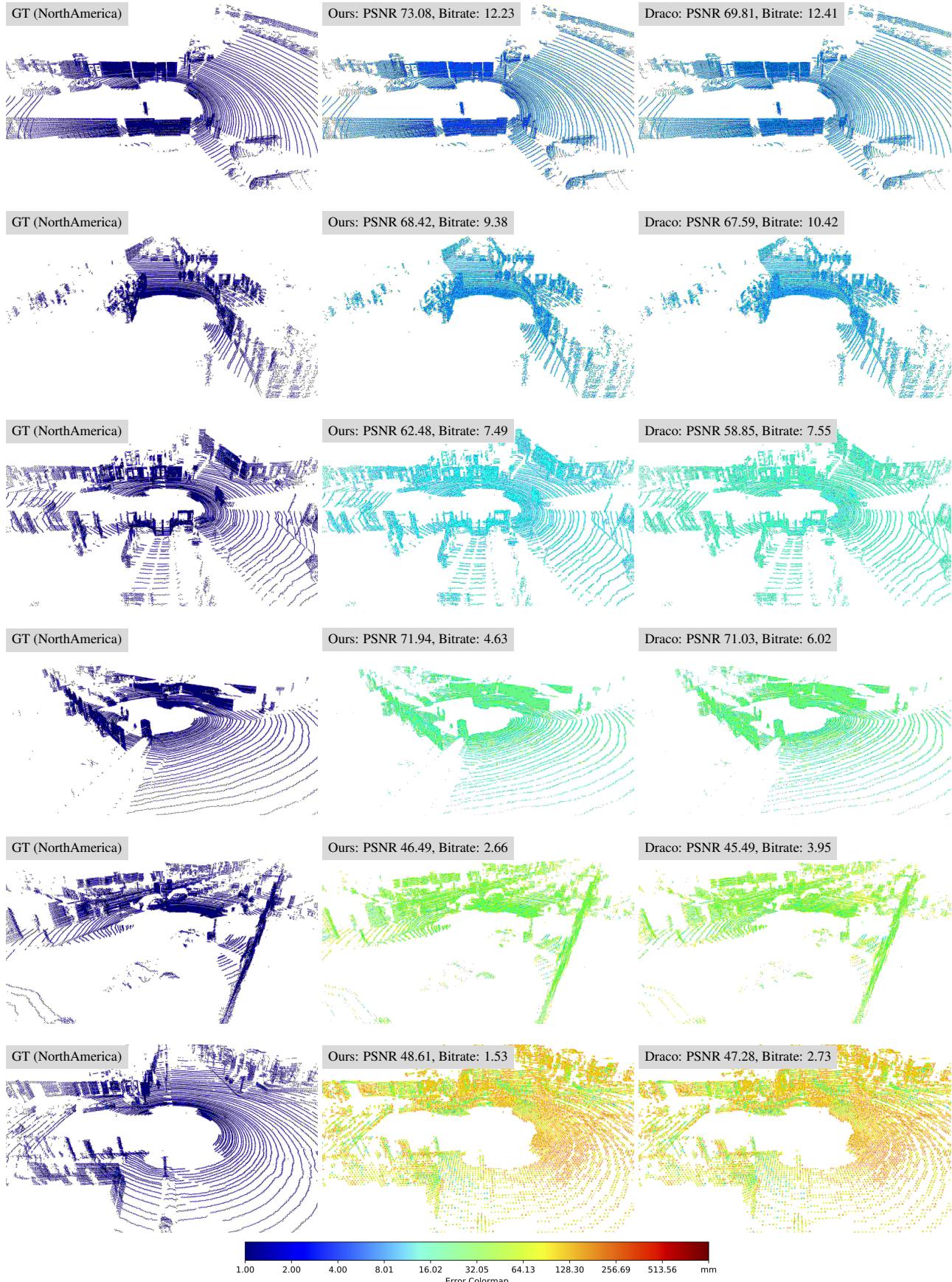


Figure 2: Qualitative results of reconstruction quality for NorthAmerica.

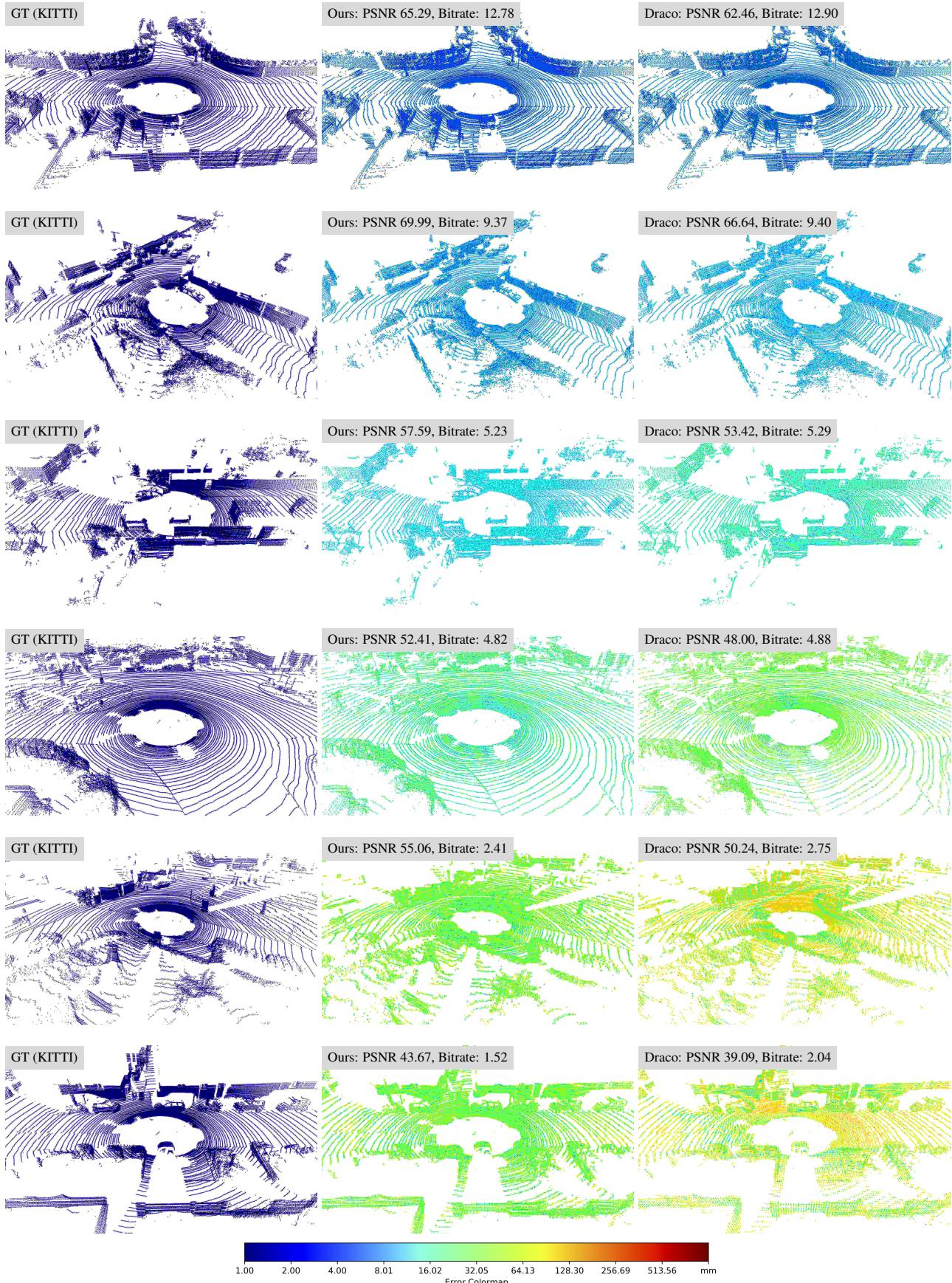


Figure 3: Qualitative results of reconstruction quality for KITTI.

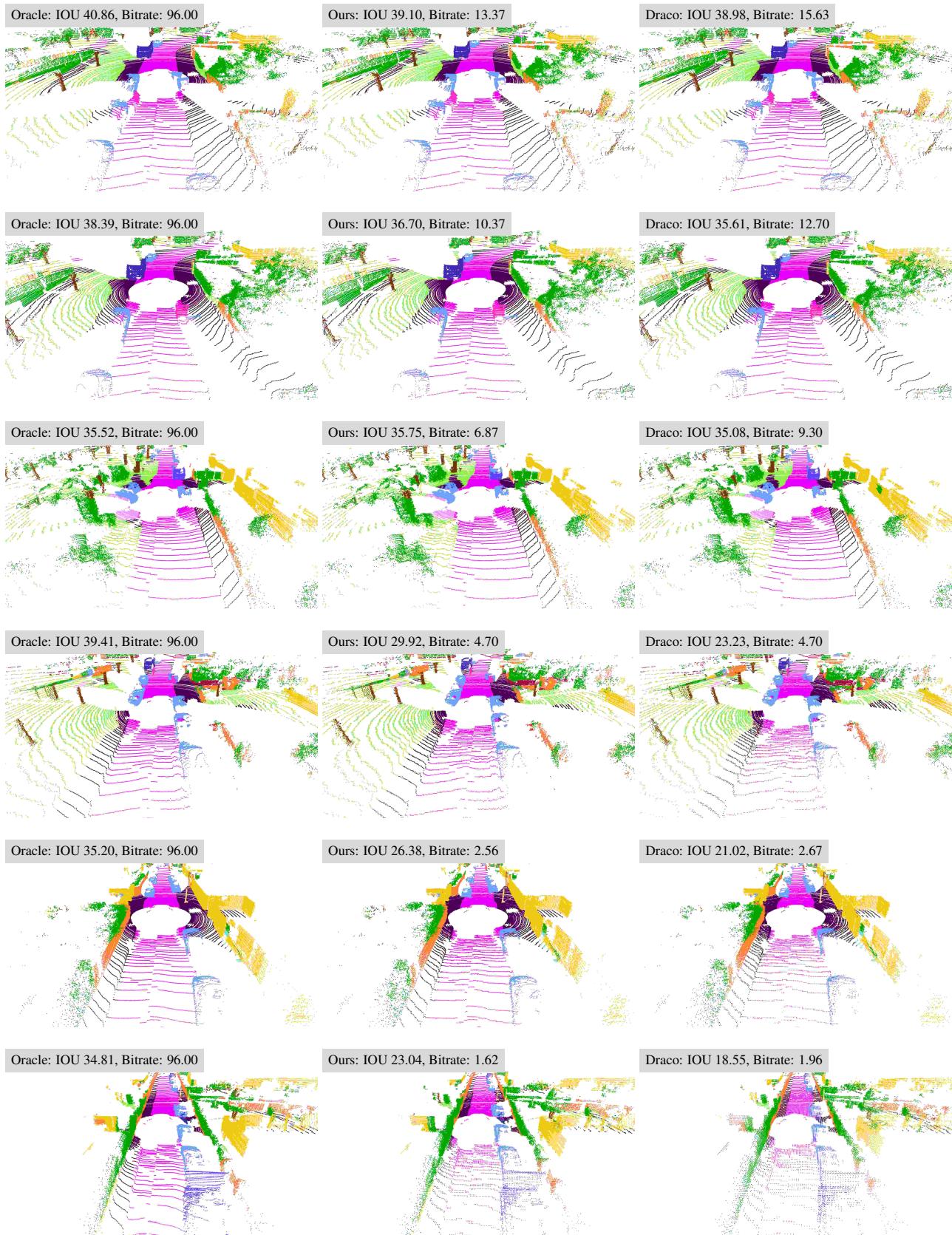


Figure 4: Qualitative results of semantic segmentation for KITTI. IOU is averaged over all classes.

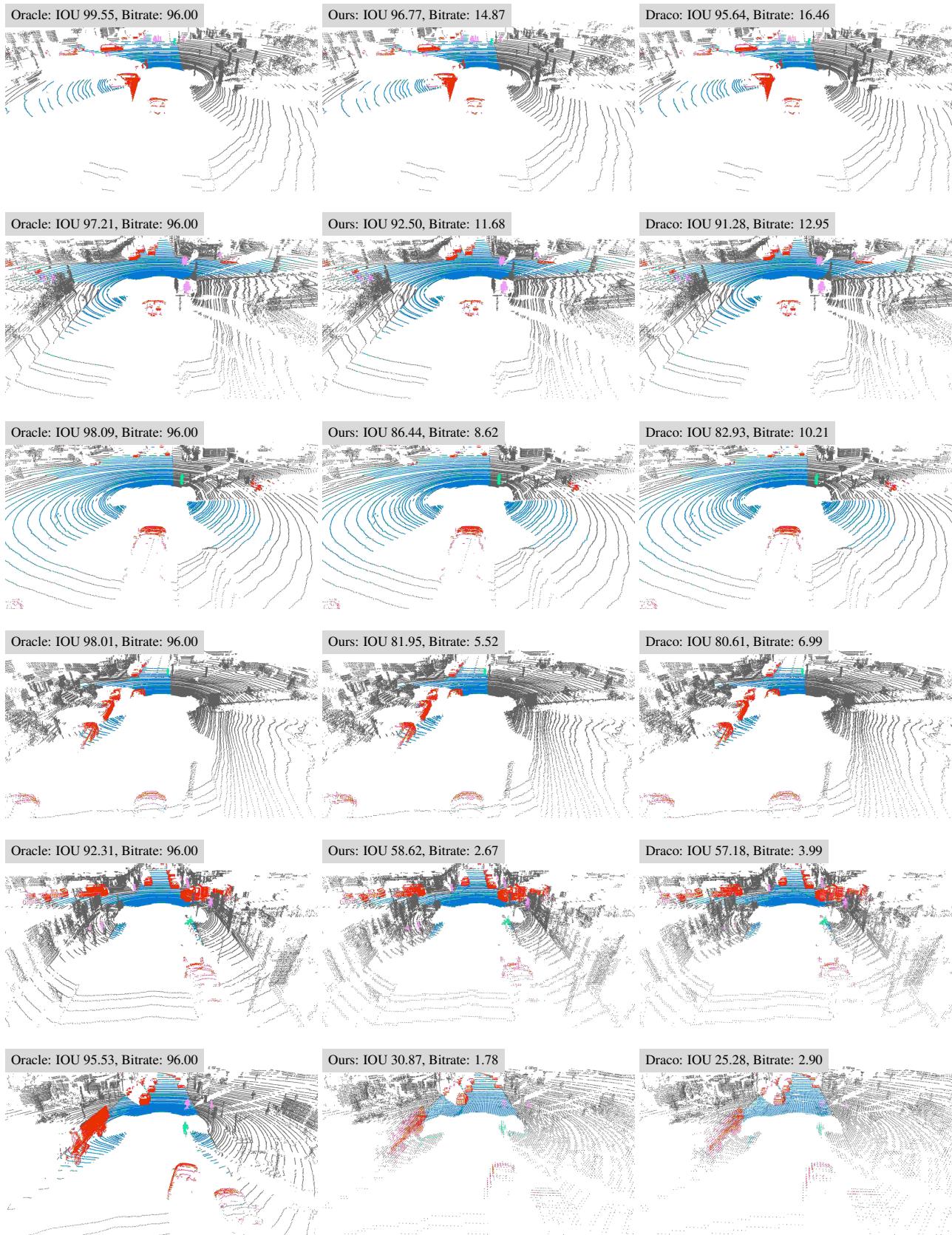


Figure 5: Qualitative results of semantic segmentation for NorthAmerica. IOU is averaged over all classes.

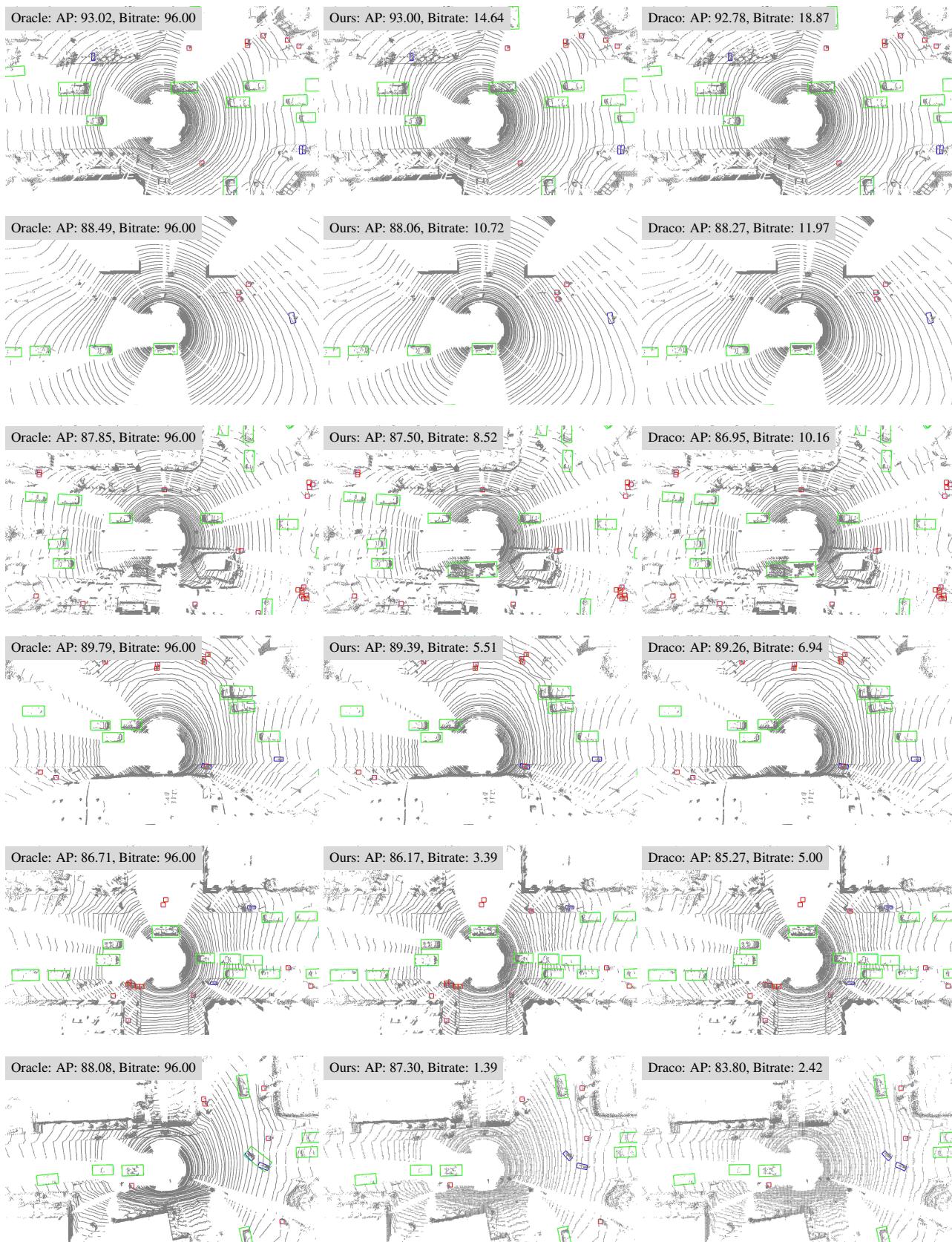


Figure 6: Qualitative results of object detection for NorthAmerica. AP is averaged over vehicle, motorbike, and pedestrian classes.