LIDAR RANGE IMAGE COMPRESSION WITH DEEP DELTA ENCODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Lidars are widely used in applications such as autonomous driving and augmented reality. However, the large volume of data produced by lidars can lead to high cost in data storage and transmission. Besides rising standards in point cloud compression from the MPEG (G-PCC and V-PCC), recent works also explore using deep networks to improve the compression rates. However, most prior work focus on the generic point cloud representation, neglecting the spatial patterns of the points from lidar range images. In this work, we leverage the range image representation and propose a novel deep delta encoding model to compress lidar data. Our deep model takes in local range image patches and predicts the next pixel value in a raster-scanning manner. The residuals between the prediction and the original value can be entropy encoded to achieve lossless compression under certain quantization rates. Evaluated on the Waymo Open Dataset and KITTI, our method demonstrates significant improvement compared to widely algorithms as well as recent deep methods based on the point cloud representation, in both the point cloud reconstruction quality and the downstream perception model performance.

1 INTRODUCTION

Applications that require 3D scene understanding such as autonomous driving and augmented reality lead to fast growing adoption of 3D depth sensors. Among the depth sensors, the active lidar (or LiDAR, short for light detection and ranging) sensors can more accurately capture geometry information at longer ranges, compared to passive depth sensors. This has contributed to lidars becoming standard equipment on most autonomous vehicles, and their adoption on some drones and, more recently, even on smart phones. However, with the growing capability (higher resolution, longer ranges) of lidars and the increasing volume of the captured data, enabling efficient storage and transmission of this new type of data becomes an important research problem. At its core is a strong need for effective lidar data compression algorithms.

While the measurements of a lidar sensor are usually visualized as 3D points (or point clouds), the lidar data can be conveniently represented as a structured range image, where each pixel represents the response of one ray shooting at a predefined elevation and azimuth angle. To leverage the spatial patterns in this structure, we propose to compress the range images directly, instead of compressing the unprojected point cloud from the range image as many previous works (Huang et al., 2020; Biswas et al., 2020; Que et al., 2021). Representing the lidar data in range images naturally takes less bits compared to using the point clouds. A point in a point cloud needs to use three values for its 3D coordinate while it only needs one value for its range in a range image. With sensor calibration information, a range image can be conveniently converted to the corresponding point cloud.

Specifically, we propose a *deep delta encoding* approach to compress the range images with predictive neural networks. Our method is inspired by the traditional delta encoding method for data compression, which is used for lossless image compression with the PNG format. However, instead of computing a simple diff between close-by pixels, we adopt a deep model to predict the pixel value from already decoded context pixels. The design of the deep model is similar to the sequential image decoder as proposed in PixelCNN (Van Oord et al., 2016). It takes a local patch of the quantized range image and predicts the attributes of the next pixel in a raster-scanning order. We can then entropy encode the residuals between the predicted values and the original values to achieve *lossless* compression under a certain quantization size. Although we operate on the range images, we found that a point-cloud-based deep network adapted from the PointNet (Qi et al., 2017), taking local point cloud converted from the image patch, can be more effective than CNNs in handling non-uniform distribution of elevation angles in the range image rows. Furthermore, instead of directly regressing the attributes, we treat each pixel in the input patch as an "anchor" and predict confidence scores as well as residual values per anchor, for all the participating pixels.

The residuals between the prediction and the original pixel values are sparse and concentrated in distribution. We can then entropy encode the residuals to achieve smaller bitrate compared to directly compressing the original image. The more accurate the prediction model, the smaller the entropy of the residuals. Therefore, improving the compression rates is equivalent to developing a more accurate predictive model.

Evaluated on the large-scale Waymo Open Dataset (WOD) (Sun et al., 2020), we show that our method reduces the bitrate by more than 60% for the same distortion (measured using the point-to-point Chamfer distance) or reducing more than 80% distortion for the same bitrate, compared to the MPEG standard compression method G-PCC (Graziosi et al., 2020). On the KITTI dataset (Geiger et al., 2013), we compare with prior art deep compression methods with octrees and show our deep delta encoding method significantly outperforms them. We also evaluate the impact of compression on downstream perception application such as 3D object detection.

To summarize, the contributions of this work are as follows:

- We propose a novel lidar range compression method, named *deep delta encoding*, achieving state-of-the-art performance;
- We design an effective intra-frame predictive deep neural network for accurate prediction of next-pixel attributes in range images; and
- We provide thorough evaluations of the proposed method on both the WOD and KITTI dataset with extensive ablation studies and qualitative analysis.

2 Problem Formulation

A lidar range image $x \in \mathbb{R}^{H \times W \times C}$ is represented as a three-dimensional tensor with H rows, W columns and C channels, with $x_{i,j}$ denoting the pixel in i-th row and j-th column and x_t denoting the t-th pixel in the raster-scanning order. The channels contain the range information and other attributes such as intensity (reflection strength) and the elongation, etc. Each pixel in the range image represents one laser shot, with a specific elevation and azimuth angle (of the laser ray). With the lidar sensor pose and the laser shot angles, one can unproject the range image to a point cloud. Some of the pixels in the range image are invalid (i.e., no response received), so the total number of valid points in a range image is usually smaller than HW.

For our lidar range image compression, the goal is to compress the *quantized* range image x' to a bistream $b \in [0, 1]^n$ (with an n as small as possible), which can later be decompressed into the exact quantized range image x'. It is lossy compression with respective to the raw range image but lossless regarding the quantized range image.

3 Deep Delta Encoding for Range Image Compression

We first describe our overall compression pipeline in Sec. 3.1, then dive deep into the design of our prediction model in Sec. 3.2, and finally describe how we entropy encode the residuals in Sec. 3.3.

3.1 The Compression Pipeline

We follow the delta encoding (or delta compression) idea for lidar range image compression, which is generally applicable to different attributes of a range image, such as range, intensity and elongation. For simplicity, we focus the discussion of the method on compressing the range attribute.

As shown in Fig. 1, the input to our compression pipeline is a raw range image. First, we quantize the range image with a certain quantization interval for each channel (e.g., 0.1m for the range channel

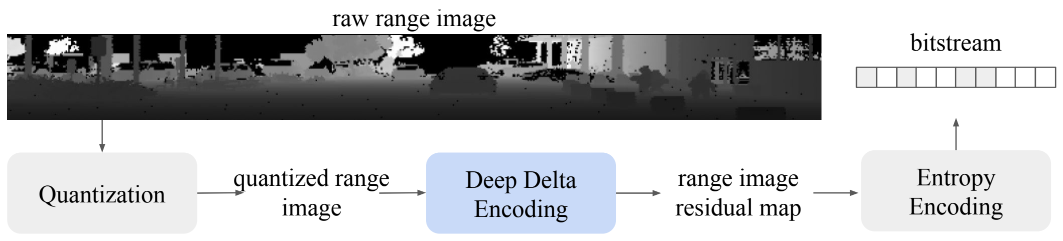


Figure 1: **The deep delta encoding pipeline for lidar range image compression.** Given a lidar range image, we first quantize the attribute values and then run inference of the predictive model on the quantized range image to derive residuals. Finally we use entropy encoders to compress the residuals to a bitstream.

as shown in the figure). This is to enable efficient compression of the residual values (as multipliers of the quantization interval) in the later stage of the pipeline.

Next, the core part of the pipeline is the delta encoding step. In its most naive form, we can store the delta between the adjacent pixel values $x'_{i,j} - x'_{i,j-1}$. As adjacent pixels tend to have similar values, the deltas would be small and with a highly concentrated range therefore can be more easily compressed compared to the original range image. This baseline approach is equivalent to using the left pixel value as the "prediction" for the current pixel. A more generalized solution is to utilize a "model" to predict the value $x_{i,j}$ based on pixels that have already been predicted, i.e. $\{x'_{t-1}, ..., x'_1\}$ with $x_{i,j}$ as the t-th pixel in the raster scanning order. This model can either be a linear model with predefined filters or a learned model such as a deep neural network. The more accurate the model is in predicting the next pixel value, the smaller and more concentrated (towards zero) the residuals are, leading to lower entropy and thus higher compression rate with an entropy encoder. Therefore, under this formulation, an improved model directly results in a higher compression rate.

The last step is the entropy encoder for compressing the residuals, which takes advantage of the concentrated distribution (low entropy) of residual values to achieve high compression rate. We experiment with existing entropy encoders such as Huffman encoding, arithmetic encoding as well as run-length encoders and empirically select the the encoder with the highest compression rate.

3.2 Deep Predictive Model for Range Images

In this section, we describe our design of using a deep neural network to predict range image attributes in a raster-scanning order. Formally, the network models the conditional probability of the t-th pixel value conditioned on the quantized pixel values before t: $p(x_t; \Theta) = p(x_t | \{x'_{t-1}, ..., x'_1\}; \Theta)$, where Θ are the network weights. Empirically, as shown in Fig. 2, instead of using all history (e.g. with a RNN model), we can use local image patch of shape $h \times w$ with the bottom right pixel as the to-be-predicted pixel, masked out as zero. [1] For pixel locations at the boundary of the range images, we enforce the same patch size via zero padding. The output of the model is the predicted attributes $\hat{x}_t$ of the bottom right pixel. Then the residual is computed as the difference between the ground truth quantized pixel value and the prediction: $r_t = x'_t - \hat{x}_t$. The r_t are multiples of the quantization interval and can be represented as discrete symbols to be later entropy encoded.

Inference. At inference time (for compression), we start from the top left patch of the range image to predict pixel x_1 or $x_{1,1}$ and store the residual. This process continue in a raster scanning order to predict pixels $x_{1,2}, ..., x_{1,W}, x_{2,1}, ..., x_{H,W}$. The residual map of size $H \times W$ will be compressed by the entropy encoder later. At the decompression time, we run the prediction model in the same raster-scanning order, which takes input as already reconstructed pixels $\{x'_1, .., x'_{t-1}\}$, predicts the next pixel value $\hat{x}_t$ and then reconstruct the pixel from saved residual as $x'_t = \hat{x}_t + r_t$. Both the compression and decompression process can be parallelized by splitting the input range image into blocks and run the inference in parallel for each block.

[1]For simplicity, we consider $C = 1$ with the range channel only while our method can easily extend to multi-channel joint prediction.

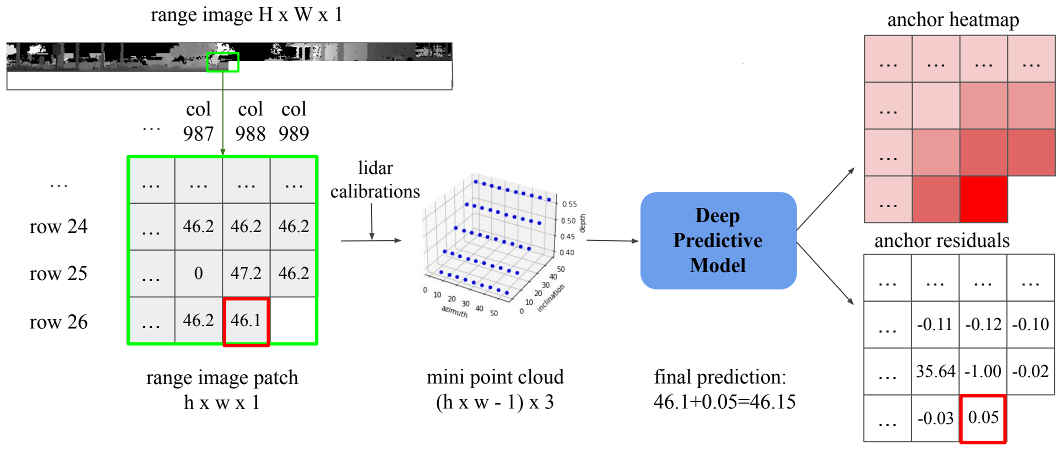


Figure 2: **The deep intra-frame prediction model.** Given a range image patch with quantized attribute values, we associate pixel values with their relative laser shot angles (elevation and azimuth) to the to-be-predicted pixel (on the bottom right). The pixels with relative angles and depth values can be considered a 3D point cloud in the (azimuth, elevation, depth) space. Our predictor then takes this mini point cloud and predict the attribute (depth in this example) of the bottom right pixel.

Deep predictive model. As our pipeline is not tied to any specific deep neural network architecture for the range image prediction, there are several choices of design including fully connected networks (the input is flattened range image patch), convolutional networks (the input is the range image patch as an image) and point-cloud (the input is a mini point cloud where each point is derived from a valid pixel in the range image patch) or graph based neural nets. Empirically, we found using point-cloud-based neural network (adapted from the PointNet proposed by Qi et al. (2017)) achieves the best prediction results. Compared to ConvNet, we found the point-cloud network is more effective in using the laser shot angle signals, which are non-uniform across rows.

As shown in Fig. 2, for the point-cloud-based network, the input is a mini point cloud from the range image patch, where each point corresponds to one pixel with the relative azimuth angle, elevation (inclination) angle to the bottom right pixel as well as the range attribute as the point coordinate. Instead of directly regressing the pixel range value, which suffers from the uncertainty caused by the multi-modal distribution of attributes (esp. on the object boundaries), we formulate the attribute regression problem as an anchor based classification and residual regression problem. We consider all valid pixels in the range image patch as anchors. The deep network predicts which pixel is the closest in value to the masked pixel on the bottom right of the patch and regresses a residual attribute with respect to each anchor pixel.

Let s_i be the predicted scores for the anchor pixels/points and g_i as the residuals predicted for each pixel/point, with $i = 1, ..., hw - 1$. At inference time, we select the anchor point with the highest prediction score $j = \text{argmax} s_i$ and return the predicted attribute as $p_j + g_j$ where p_j is the attribute from the j-th input point.

Loss functions. At training time, our intra-frame prediction model is trained end-to-end with the anchor classification and the anchor residual regression loss. We weight the classification loss by a weight.

$$\mathcal{L} = \gamma \mathcal{L}_{\text{classification}} + \mathcal{L}_{\text{regression}} \tag{1}$$

The classification loss is a cross-entropy loss across $hw - 1$ classes. The ground truth class is selected as the index of the pixel with the closest distance to the to-be-predicted pixel. As the input are quantized values, there could be ties. To avoid ties we add a bias term to the distances to favor the pixels that are closer in Manhattan distance to the to-be-predicted pixel and are closer to the bottom row of the input. The regression loss is a L1 loss between the predicted residual corresponding to the pixel of the ground truth class and the ground truth residual of that pixel. Please see the Appendix for the details on model training.

3.3 Entropy Encoding of the Residuals

After the predictive delta encoding, we get a residual map of the range image. An entropy encoder is used to leverage the sparsity pattern in the residual map to compress it. Given an accurate intra-frame prediction model, most of the residuals will be zero. In addition, as shown in Fig. 6 in the Appendix, larger quantization steps will round more residuals to zero, thus the residuals will become more sparse. We adapted two methods to entropy encode the residuals. In practice, we can select the entropy encoder with the highest compression rates depending on the quantization rates and the predictor.

The first method is to represent the residuals using sparse representation. Given an array of residuals, we represent the array with the values of the nonzero residuals and their indices in the array. For a long run of sparse residuals, the sparse representation will be quite memory efficient. After obtaining the sparse representation of residuals, we use arithmetic encoding to further reduce its size.

The second method is to represent the residuals using run-length encoding. We first flatten the residual map to a vector and then represent it with the values and the run-length of values. This representation achieves better compression rates when the residuals are not very sparse, i.e. when quantization step size is small. After obtaining the run-length representation, we use LZMA compressor to further reduce its size.

4 Experiments

In this section, we first introduce the dataset we experiment with in Sec. 4.1 and the metrics we used in Sec. 4.2. Then we report compression results compared with strong prior methods in Sec. 4.3 both quantitatively and qualitatively. We further evaluate the impact of compressed data to downstream perception tasks (3D detection of vehicles and pedestrians) in Sec. 4.4. Finally, we provide extensive analysis experiments to validate our design choices in Sec. 4.5.

4.1 Dataset

We evaluate our approach using both the Waymo Open Dataset (WOD) (Sun et al., 2020) and the KITTI dataset (Geiger et al., 2013)).

The WOD includes a total number of 1,150 sequences with 798 for training and 202 for validation. Each lidar sequence lasts around 20 seconds with a sampling frequency of 10Hz. A 64-beam lidar is used, providing range images of 64 rows and 2,650 columns, with provided lidar calibration metadata (beam inclination angles). The range channel is cropped to 75m, and each raw range value in the range image uses 32 bits to store in default. We use the training set of the dataset to train our deep predictive model and evaluate the compression rate on the validation set.

For the KITTI dataset, as it only released the unprojected point cloud data but not the the raw range images, we refer to the manual of the lidar used by KITTI (Velodyne HDL) to find out the angular resolutions of the lidar and convert the point clouds to a spherical coordinate with 64 rows and 2,083 columns, as a pseudo range image. For our method, we compress the pseudo range images and compute the distortion by comparing the point clouds between the one unprojected from the quantized and uncompressed range images.

4.2 Compression Metrics

Following previous works (Graziosi et al., 2020; Huang et al., 2020), we use two geometric metrics to evaluate the reconstruction quality of the compressed point cloud data: point-to-point Chamfer distance and point-to-plane peak signal-to-noise ratio (PSNR). We report these metrics as a function of bitrates i.e.,the average number of bits to store one lidar point.

The (symmetric) point-to-point Chamfer distance CD_{sym} measures the average point distances between two point clouds (smaller the better). For a given point cloud $\mathcal{P} = \{p_i\}_{i=1,...N}$ and the reconstructed point cloud $\hat{\mathcal{P}} = \{\hat{p}_j\}_{j=1,...M}$:

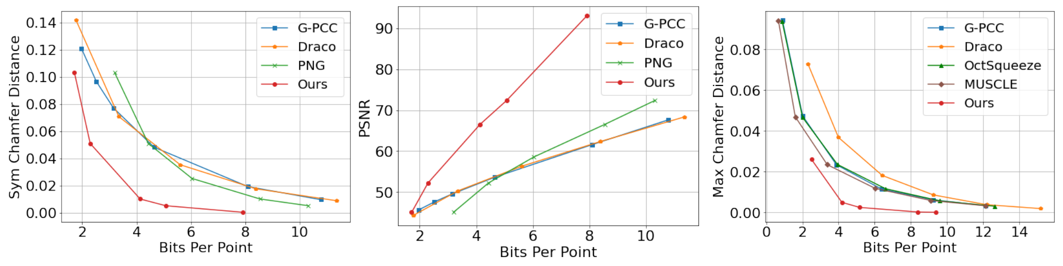


Figure 3: **Evaluation of the compression methods with geometric metrics.** *Left*: symmetirc Chamfer distance v.s. bit per point (bbp) on the WOD data; *Middle*: PSNR v.s. bpp on the WOD data; *Right*: max Chamfer distnace v.s. bpp on the KITTI dataset. We compare our deep delta encoding compressor with three strong non-learning baselines: G-PCC, Draco and PNG (on WOD) and two prior art deep models: OctSqueeze and MUSCLE (on KITTI). At a certain bitrate, a lower the Chamfer distance or the higher the PSNR means better reconstruction quality.

$$\mathrm{CD}(\mathcal{P}, \hat{\mathcal{P}}) = \frac{1}{|\mathcal{P}|} \sum_i \min_j \|p_i - \hat{p}_j\|_2 \tag{2}$$

$$\mathrm{CD}_{sym}(\mathcal{P}, \hat{\mathcal{P}}) = \mathrm{CD}(\mathcal{P}, \hat{\mathcal{P}}) + \mathrm{CD}(\hat{\mathcal{P}}, \mathcal{P}) \tag{3}$$

The second metric, the peak signal-to-noise ratio (PSNR) (Tian et al., 2017) (the larger the better), measures the ratio between the "resolution" of the point cloud r and the average point-to-plane error between the original point cloud $\mathcal{P}$ and the reconstructed point cloud $\hat{\mathcal{P}}$:

$$\mathrm{PSNR}(\mathcal{P}, \hat{\mathcal{P}}) = 10 \log_{10} \frac{r^2}{\max\{\mathrm{MSE}(\mathcal{P}, \hat{\mathcal{P}}), \mathrm{MSE}(\hat{\mathcal{P}}, \mathcal{P})\}} \tag{4}$$

where $\mathrm{MSE}(\mathcal{P}, \hat{\mathcal{P}})\} = \frac{1}{|\mathcal{P}|} \sum_i ((p_i - \hat{p}_i) \cdot n_i)^2$ is the point-to-plane distance, $r = \max_{p_i \in \mathcal{P}} \min_{j \neq i} \|p_i - p_j\|_2$ is the intrinsic resolution of the original point cloud. We estimate the normal n_i using Open3D (Zhou et al., 2018) with $k = 12$ for k nearest neighbor.

4.3 Compression Results

Waymo Open Dataset results We report the bitrate versus reconstruction quality metrics (PSNR, Chamfer distance) of competing methods on all frames from the sequences in the validation set of the Waymo Open Dataset. We compare with three strong baseline methods: the G-PCC (Graziosi et al., 2020) from the MPEG standard that uses geometry-based method (Octree) to compress point clouds; the Draco (dra) proposed by Google; and using the image compression method PNG to compress the range image (the range is coded with 16 bits with a scaling factor of 100). Note that delta encoding is also used in PNG although the prediction is based on heuristic rules rather than learned from data. For all comparisons, we mesure the compression of the range channel.

As shown in Fig. 3 (Left and Middle), our method significantly outperforms prior methods. For the point-to-point error (Chamfer distance), our method reduces bitrates by 20%-150% while achieving the same Chamfer distance. For the point-to-plane error (PSNR) metric, our method reduces the bitrate by 66%-150% while achieving the same PSNR.

Our method also has larger bitrate improvement over previous methods when the reconstruction quality is higher. This indicate our method has a stronger advantage over G-PCC and Draco when the data quality requirement is higher.

KITTI results We compare our model with prior state-of-the-art lidar data compression methods based on Octrees with the generic point cloud representation – specifically the OctSqueeze (Huang et al., 2020) and the MUSCLE (Biswas et al., 2020). We apply our model trained on the

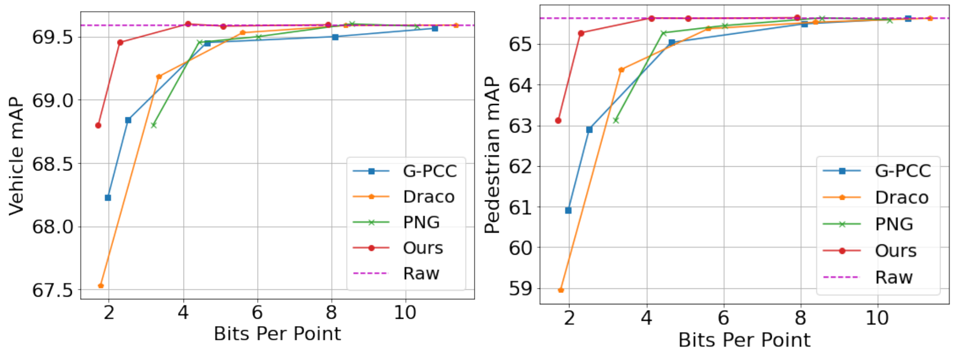


Figure 4: **Impact of lidar data compression to 3D object detection quality.** Using a pre-trained PointPillars detector on the Waymo Open Dataset train set on the raw point cloud (with no compression), we evaluate the detector with compressed point clouds on the validation set.

Waymo Open Dataset and directly apply to the KITTI dataset, using the frames from the SemanticKITTI (Behley et al., 2019) following the previous works.

Fig. 3 (Right) shows the curve of point Chamfer distance v.s. bitrate on the KITTI dataset [2]. Since MUSCLE uses a different version of Chamfer distance, we adapt our metric in order to directly compare with it. The max Chamfer distance is used here: $\text{CD}_{max}\{\mathcal{P}, \hat{\mathcal{P}}) = \max(\text{CD}(\mathcal{P}, \hat{\mathcal{P}}) + \text{CD}(\hat{\mathcal{P}}, \mathcal{P})\}$. Our method is more than 50% lower in bitrate with the same max Chamfer distance at around 0.005 compared to all prior art methods, showing significant advantages. This strong lead attributes to our choice of directly compressing the range images as well as the effective deep model.

Qualitative results. In Fig. 5, we show the reconstructed lidar point clouds from our method, Draco and G-PCC. We can see that the point cloud reconstructed from our method remarkably resembles the original point cloud in geometry even when the bitrate is ambitiously set very low, thanks to compressing directly on the range images to keep the point distribution pattern.

4.4 Impact to Downstream Perception Tasks

For applications like autonomous driving, besides the basic geometry metrics, we also want to understand the impact of laser data compression to downstream perception tasks such as 3D object detection. To understand such impact, we trained the widely used PointPillars detector (Lang et al., 2019) on uncompressed point clouds using the Waymo Open Dataset train set, for the vehicle class and pedestrian class respectively. Detection quality is measured by mean average precision (mAP).

As shown in Fig. 4, our method outperforms other competing baselines in maintaining the best mAP with the same bitrate. At the bitrate $= 2$, our method leads the second best method (G-PCC) by more than 1 point on vehicle detection and 3 point on pedestrian detection.

4.5 Analysis Experiments

For the albation studies, we define the prediction accuracy metric as the percentage of zero residuals (i.e. perfect prediction) in range image residuals, under a specific quantization step size $\delta = 0.1$m (note that 0.1m is actually not that coarse as average point displacement after the quantization is only 2.5cm). A prediction q for the range value p' is counted as correct if $|q - p'| < \delta/2$.

Analysis on predictor choices. To validate our choice of PointNet as the backbone model for the predictor, we ablate architectures used for predictors. From Table 1, we can see that hand-crafted convolution kernels like 1D or 2D linear interpolation have much lower prediction accuracy

[2]Since OctSqueeze and MUSCLE have not released the code and compression model, to compare with them, we refer to the original curves from Biswas et al. (2020). We have sanity checked the consistency of the comparison by replicating the Draco method which matches with their curve.

Table 1: Effects of prediction models.

Model	Acc. @0.1m
Previous valid value	54.35
Linear interpolation	54.64
12-layer CNN	64.62
PointNet	**65.75**

Table 2: Effects of loss functions.

Loss function	Acc. @0.1m
Mean squared error	59.83
Mean absolute error	61.64
Multi-bin loss	59.66
Anchor cls. + residual reg.	**65.75**

Table 3: Effects of context size and input format.

Context size	Input format	Acc. @0.1m
10 x 1	(Δazimuth, Δinclination, depth)	37.53
1 x 10	(Δazimuth, Δinclination, depth)	60.00
5 x 10	(Δrow index, Δcol index, depth)	65.02
5 x 10	(Δazimuth, Δinclination, depth)	65.21
10 x 10	(Δazimuth, Δinclination, depth)	**65.75**

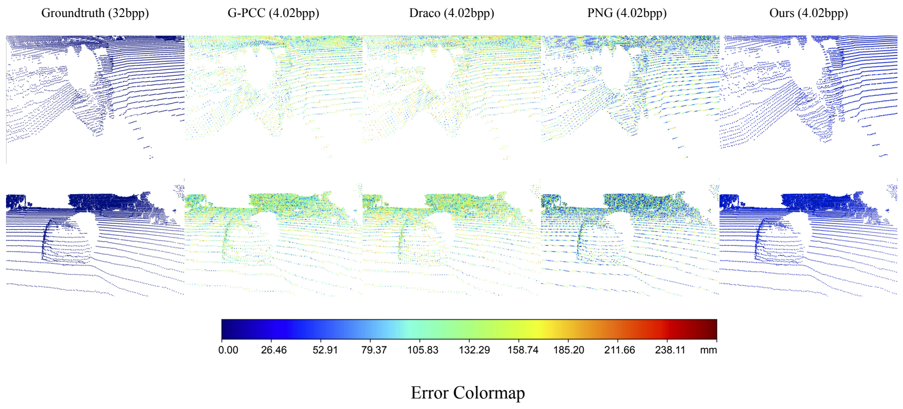


Figure 5: **Visualization of reconstructed point clouds, colored by per point Chamfer distance.** From left to right: raw, G-PCC, Draco, PNG and Ours. It is clear that our compression method, under the same bit per point, has mush less distortion. This figure is best viewed in color.

compared to learned neural networks, and they are only slightly better than the most naive baseline of predicting the current pixel value by using the previous valid pixel value in the same row. The advantages of the neural network methods show that through supervised training, deep models can effectively learn the joint distributions of neighboring attributes. The best quantization accuracy is achieved by using the point-cloud-based deep net adapted from the PointNet.

Analysis on predictor loss functions. Table 2 compares our design with alternative attribute supervision schemes. With direct attribute prediction as a regression problem, we can see using the mean absolute error (L1 loss) is superior to using the mean squared error (L2 loss) as it is affected less by the large errors on the object boundaries. Turning the depth regression problem naively to a multi-bin classification and regression problem (with classification and intra-bin regression for each bin) does not help much either as shown in the third row. Our proposed formulation uses an anchor-based classification and residual regression for each anchor and leads to 4.11 point increase in prediction accuracy compared to the second best option of using mean absolute error.

Analysis on input choices. Table 3 shows how the input choices affect the prediction accuracy. We can see that enlarging the receptive field of input with context from both upper left and upper right can improve the prediction accuracy (row 3 v.s. row 1 and 2). Moreover, including azimuth and inclination as additional input attributes can also improve the predictor (row 4 v.s. row 3).

5 Related Work

Point cloud compression As 3D applications rise, recent years have seen an increasing number of algorithms proposed for point cloud compression. One family of the methods use octree to represent and compress quantized point clouds (Botsch et al., 2002; Devillers & Gandoin, 2000; Schnabel & Klein, 2006). The Motion Picture Experts Group (MPEG) has released a related point cloud compression (PCC) standard, called geometry-based PCC (G-PCC) (Graziosi et al., 2020), using the octree structure and various ways to predict the next level content. More recently, Huang et al. (2020) proposed to use a neural network as a conditional entropy model to estimate the octree symbols and Biswas et al. (2020) extended it by including temporal prior from previous frames. Que et al. (2021) further leverages the voxel context for the octree structure prediction. These neural network based methods consistently show improvements over the G-PCC methods that use hand-crafted entropy models. While the octree based methods are flexible to model arbitrary point clouds (from either a lidar sensor or multi-view reconstruction), they do not make use of the point distribution patterns in lidar range images.

As lidar point cloud is usually represented as a range image with multiple channels typically including range and intensity, image-based compression methods can then be adapted to compress the range images. For example, Houshiar & Nüchter (2015); Beek (2019) applied traditional image compression methods such as JPEG, PNG and TIFF to compress the range images. MPEG proposed a video-based PCC (V-PCC) standard that compresses dynamic point cloud via HEVC video compression codex (Graziosi et al., 2020). Our work extends them to leverage deep models and delta encoding to compress range images.

Auto encoders have also been used to achieve lossy compression of point clouds. Yan et al. (2019); Wiesmann et al. (2021) proposed to train an encoder-decoder point cloud reconstruction network and entropy encode the bottleneck layer as the compressed data. Similarly, Tu et al. (2019) trained an auto encoder to reconstruct the range images and compress the bottleneck vectors. While these methods may achieve high compression rate, the reconstructed point clouds could have strong artifacts especially at the object boundaries.

Learned image and video compression Image and video compression are well studied fields with many standards (for example: PNG, JPEG, TIFF for images, H.264 and HEVC for videos). Among them the PNG is highly related to our work as it supports lossless image compression using delta encoding. With the popularity of deep convolutional neural networks for image understanding, deep model based image and video compression have also been widely explored (Ballé et al., 2016; Toderici et al., 2017; Ballé et al., 2018; Townsend et al., 2019; Mentzer et al., 2019; Ma et al., 2019). Many of them leverages an encoder-decoder neural network (for example, a variational autoencoder as used by Ballé et al. (2016)) for the compressing (encoding the image to a latent vector) and decompressing (decode/generate the image from the vector). As optical images have RGB channels in relatively low precision (for example 8-bit per color), the generation model can formulate pixel prediction as a classification problem (into the 256 bins) (Mentzer et al., 2019)). For the decoding architectures, sequential models such as the PixelCNN (Oord et al., 2016) and the PixelRNN (Van Oord et al., 2016) inspired our predictive model design.

6 Conclusion

With the improved lidar sensor quality and growing data volume, how to efficiently store and transmit lidar data becomes a challenging problem in many 3D applications, such as autonomous driving and augmented reality. To address this challenge, we propose a novel lidar data compression algorithm based on predictive delta encoding of range images, which combines the succinctness of traditional delta encoding and the expressiveness of deep neural networks. Experiments over the Waymo Open Dataset show that compared to previous methods, the proposed approach yields significant improvement in the point cloud reconstruction quality and the downstream perception model performance, under the same compression rates. As future work, we would like to explore using temporal signals to further increase the compression rate for sequential range image data.

References

Velodyne hdl-64e. `https://gpsolution.oss-cn-beijing.aliyuncs.com/manual/LiDAR/MANUAL%2CUSERS%2CHDL-64E_S3.pdf`. Accessed: 2021-10-04.

Draco. `https://github.com/google/draco`. Accessed: 2021-09-28.

Johannes Ballé, Valero Laparra, and Eero P Simoncelli. End-to-end optimized image compression. *arXiv preprint arXiv:1611.01704*, 2016.

Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston. Variational image compression with a scale hyperprior. *arXiv preprint arXiv:1802.01436*, 2018.

Peter van Beek. Image-based compression of lidar sensor data. *Electronic Imaging*, 2019(15):43–1, 2019.

Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9297–9307, 2019.

Sourav Biswas, Jerry Liu, Kelvin Wong, Shenlong Wang, and Raquel Urtasun. Muscle: Multi sweep compression of lidar using deep entropy models. *arXiv preprint arXiv:2011.07590*, 2020.

Mario Botsch, Andreas Wiratanaya, and Leif Kobbelt. Efficient high quality rendering of point sampled geometry. *Rendering Techniques*, 2002:13th, 2002.

Olivier Devillers and P-M Gandoin. Geometric compression for interactive transmission. In *Proceedings Visualization 2000. VIS 2000 (Cat. No. 00CH37145)*, pp. 319–326. IEEE, 2000.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hamidreza Houshiar and Andreas Nüchter. 3d point cloud compression using conventional image compression for efficient data transmission. In *2015 XXV International Conference on Information, Communication and Automation Technologies (ICAT)*, pp. 1–8. IEEE, 2015.

Lila Huang, Shenlong Wang, Kelvin Wong, Jerry Liu, and Raquel Urtasun. Octsqueeze: Octree-structured entropy model for lidar compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1313–1323, 2020.

Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019.

Siwei Ma, Xinfeng Zhang, Chuanmin Jia, Zhenghui Zhao, Shiqi Wang, and Shanshe Wang. Image and video compression with neural networks: A review. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(6):1683–1698, 2019.

Fabian Mentzer, Eirikur Agustsson, Michael Tschannen, Radu Timofte, and Luc Van Gool. Practical full resolution learned lossless image compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10629–10638, 2019.

Aaron van den Oord, Nal Kalchbrenner, Oriol Vinyals, Lasse Espeholt, Alex Graves, and Koray Kavukcuoglu. Conditional image generation with pixelcnn decoders. *arXiv preprint arXiv:1606.05328*, 2016.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

Zizheng Que, Guo Lu, and Dong Xu. Voxelcontext-net: An octree based framework for point cloud compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6042–6051, 2021.

Ruwen Schnabel and Reinhard Klein. Octree-based point-cloud compression. In *PBG@ SIGGRAPH*, pp. 111–120, 2006.

Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2446–2454, 2020.

Dong Tian, Hideaki Ochimizu, Chen Feng, Robert Cohen, and Anthony Vetro. Geometric distortion metrics for point cloud compression. In *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 3460–3464. IEEE, 2017.

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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5306–5314, 2017.

James Townsend, Thomas Bird, Julius Kunze, and David Barber. Hilloc: Lossless image compression with hierarchical latent variable models. *arXiv preprint arXiv:1912.09953*, 2019.

Chenxi Tu, Eijiro Takeuchi, Alexander Carballo, and Kazuya Takeda. Point cloud compression for 3d lidar sensor using recurrent neural network with residual blocks. In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 3274–3280. IEEE, 2019.

Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pp. 1747–1756. PMLR, 2016.

Louis Wiesmann, Andres Milioto, Xieyuanli Chen, Cyrill Stachniss, and Jens Behley. Deep compression for dense point cloud maps. *IEEE Robotics and Automation Letters*, 6(2):2060–2067, 2021.

Wei Yan, Shan Liu, Thomas H Li, Zhu Li, Ge Li, et al. Deep autoencoder-based lossy geometry compression for point clouds. *arXiv preprint arXiv:1905.03691*, 2019.

Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018.

A Appendix

A.1 Details on Predictive Models

Model architecture For our deep intra-frame predictor, we adapt a modified structure of PointNet Qi et al. (2017). After concatenation of global and local features, we split the network into anchor-classification branch and residuals branch. Each branch is a MLP with layer sizes [128, 64, # of anchors]. For the previous valid value method, we predict the attribute $x_{i,j}$ at row i, column j to be $\hat{x}_{i,j} = x_{i,j-1}$, if $x_{i,j-1}$ is a valid attributes(not void due to empty laser return) else repeatedly decrement j by 1. For the linear interpolation baseline method, we predict $\hat{x}_{i,j} = x_{i,j-1} + x_{i-1,j} - x_{i-1,j-1}$. For the 12-layer CNN method, we adapt a similar structure as ResNet (He et al., 2016), with 12 convolutional layers in total, and each with filter size 3x3.

Training We learn the weights of the prediction model by training on the range images from the Waymo Open Dataset train set. We randomly crop the patches of shape $h \times w$ from the range images and train the deep network with batch size 128 and an Adam optimizer. We use the input size $h = 5, w = 10$ and loss weight $\gamma = 0.01$. The initial learning rate is 0.00005, and we decay learning rate by 10x at step 1500k and step 3000k. We normalize each attribute to range $[0, 1]$. To mimic the same setting in decoding, the training inputs are quantized. The ground truth attribute values are in full precision for more accurate supervision.

There are two strategies for inputs quantization. The first strategy is to train different models for different quantization precisions, and for each model we using a fixed quantization precision for the input. The second strategy is using mixed precisions to quantize the input during training. Specifically, we uniformly sample a quantization precision for a given input from 0.0001 to 0.5000 with sample bin size of 0.0001. By the second strategy, we only need to train one model for different compression rates. From our experiments, we observe that the second strategy won't harm the compression rates at individual quantization precision.

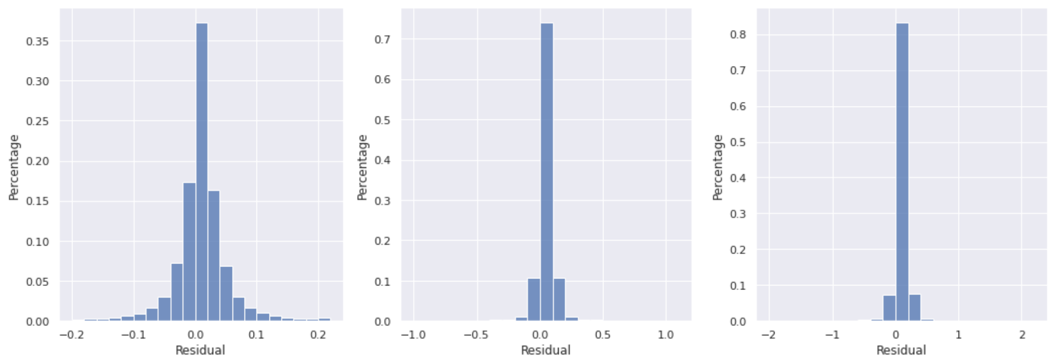
A.2 More Ablation Studies

Analysis on entropy encoders. Table. 4 shows that different entropy encoders have different compression rates of the residuals. For quantization precision of 0.1m, the residuals are more sparse, and the compression rate of using sparse representation (reprent non-zero residuals by specifying their row, col index and the residual values) with arithmetic encoding is higher than varints with LZMA. However, for quantization precision of 0.02m, the compression rate of Varints with LZMA is higher, due to the decrease in percentage of zero residuals in the residuals.

Table 4: Ablation study of the entropy encoder models.

Entropy encoder	**Quantization precision**	**Bit per point**
Sparse Representation + Arithmetic Encoding	0.1m	2.28
Varints+LZMA	0.1m	2.33
Huffman Encoding	0.1m	2.49
Arithmetic Encoding	0.1m	2.41
Sparse Representation + Arithmetic Encoding	0.02m	4.17
Varints+LZMA	0.02m	4.08
Huffman Encoding	0.02m	4.25
Arithmetic Encoding	0.02m	4.21

Analysis of the model latency. To achieve faster decompression speed, during compression, we split a range image into smaller blocks and run compression in parallel on this blocks. During decompression, we decode in parallel on these smaller blocks. Our experiments show that if we split a 64 by 2650 range image into 212 blocks with size 16 by 50, the bitrate will only increase by 0.5%, which is negligible. If we split it into 424 blocks with size 16 by 25, the bitrate will only increase by 5.13%.



(a) Residuals distribution (0.02m) (b) Residuals distribution (0.1m) (c) Residuals distribution (0.2m)

Figure 6: **Distribution of residuals at different quantization precisions**. Proposed deep prediction model is able to accurately model the joint distributions of range image pixel attributes, resulting in a concentrated distribution of residuals with low entropy.

A.3 Visualization of the Distributions of Residuals

Fig. 6 shows the distribution of the range residual maps after our deep delta encoding step. We can see that the larger the quantization interval the more concentrated are the residuals (lower entropy), which explains the lower bitrate after the compression. Note that for a quantization size of 0.1m, more than 70% of the prediction has zero error compared to the ground truth quantized range image.