LATENT SPACE SLICING FOR ENHANCED ENTROPY MODELING IN LEARNING-BASED POINT CLOUD GEOMETRY COMPRESSION

Nicolas Frank, Davi Lazzarotto, Touradj Ebrahimi

Multimedia Signal Processing Group (MMSPG)
École Polytechnique Fédérale de Lausanne (EPFL)

ABSTRACT

The growing adoption of point clouds as an imaging modality has stimulated the search for efficient solutions for compression. Learning-based algorithms have been reporting increasingly better performance and are drawing the attention from the research community and standardisation groups such as JPEG and MPEG. Learned autoencoder architectures based on 3D convolutional layers are popular solutions and have demonstrated higher performance when adopting latent space entropy modeling based on learned hyperpriors. We propose an enhanced entropy model that takes into account both the hyperprior and previously encoded latent features to estimate the mean and scale of compressed features. The obtained results show a large increase in performance, with a BD PSNR gain of 5.75dB when compared to the Octree coding module in G-PCC for the D2 PSNR metric. We also perform an ablation study to quantify the impact of network parameters in the performance of the model, drawing useful insights for future research.

Index Terms— Point cloud, compression, deep learning, entropy modeling

1. INTRODUCTION

The interest on point clouds has been growing in the last years, mainly thanks to the affordability of recent LiDAR scans, the popularization of devices adapted to the visualization of immersive content such as head mounted displays (HMD) and increasingly widespread use cases such as autonomous driving and wide area scanning. Due to the high amount of data associated to this type of content, efficient compression is paramount. Some compression algorithms use the octree to represent the points fitted to a regular grid [1], indicating the occupancy of each voxel by one bit in a tree structure. Other methods project the point cloud into multiple views and compress the obtained two dimensional maps as still images or video sequences [2].

The Moving Pictures Experts Group (MPEG) has been conducting an activity to produce two coding standards, namely G-PCC [3] and V-PCC [4], which are based on the aforementioned technologies, respectively. However, recently proposed solutions leveraging deep learning for point cloud compression have been drawing attention of both Joint Photographic Experts Group (JPEG) and MPEG due to their achieved performance and potential for improvement. This paradigm shift follows a trend previously initiated for the compression of two-dimensional images. Several proposed methods are based on the autoencoder architecture [5], which make an input image with three color channels go through an encoder with learned convolutional layers that produces a compressed representation with reduced height and width and an arbitrary number of channels to represent the latent features. The compressed features are then quantized and entropy coded in order to minimize the bitrate. This model has been later improved with a hyperprior [6] modeling the statistical dependencies on the latent space and an enhanced entropy model that conditions the probability of encoded features to previously compressed channels [7].

While 3D convolutional autoencoders and hyperpriors have already been incorporated to point cloud compression methods, little effort has been devoted to study the effect of channel-wise entropy modeling for point clouds. This paper aims to fill this gap by implementing an algorithm that slices channel-wise the latent space blocks in the bottleneck and entropy codes each slice sequentially while modeling their mean and scale parameters based on previously encoded channels. We obtain superior results when compared to G-PCC as well as to a state-of-the-art learning based compression method. Further ablation studies demonstrate the effectiveness of the channel-conditioning approach. The source code of the compression method is made publicly available to foster future research [1].

2. RELATED WORK

Numerous solutions for point cloud compression have been proposed in the literature. They are usually based on the representation of geometry in other data structures than a list of coordinates. Octrees have been widely employed for this pur-

This work was supported by the Swiss National Foundation for Scientific Research (SNSF) under the grant number 200021-178854.

[1]https://github.com/mmspg/pcc-geo-slicing

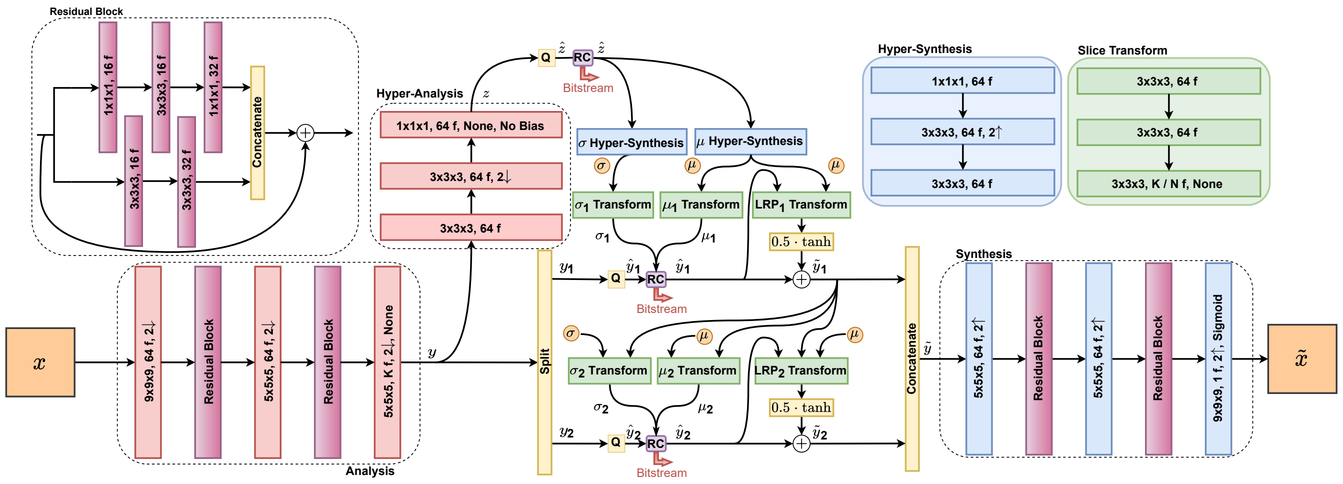


Fig. 1: Overview of the proposed architecture of the model for $N = 2$ slices. 3D Convolutional layers are specified by their kernel size and number of filters. Unless specified differently, no downsampling nor upsampling is applied, bias is used and the activation function is ReLU. "2↑" and 2↓" indicate an upsampling or downsampling stride of 2 respectively. Red blocks indicate regular 3D convolutional layers, while blue ones indicate transposed 3D convolutional layers. Q and RC indicate the quantization operation and entropy coding respectively. K represents the number of channels of y before slicing, and N the total number of slices. K must be divisible by N.

pose [1], representing the points by recursively partitioning the three dimensional space and encoding occupancy as nodes in a tree. Enhancement layers have been associated with this method by representing the geometry on the leaf nodes as planar [8] or triangular [9] primitives. The octree structure has also been leveraged by the recent compression standard G-PCC (Geometry-based Point Cloud Compression) [3] developed by MPEG, which also includes three alternative modules for color coding. Following another direction, [2] projected the point cloud into multiple views and encoded the obtained projections with a video codec. This approach is able to encode dynamic point clouds and achieves good results at low bitrates, serving as the basis of the parallel MPEG standardisation effort denominated as V-PCC (Video Point Cloud Compression) [4].

Compression algorithms using learning-based autoencoder architectures have also demonstrated interesting performance. While some operate directly on point coordinates [10], others employ voxelization as pre-processing, generating occupancy maps that serve as input to 3D convolutional layers. The work from [11] optimized the performance of an earlier algorithm [12] by adding adaptive thresholding and adjusting hyperparameters in the loss function according to point density. [13] employed an architecture with residual blocks, which was further developed by the authors in a multiscale approach [14] adopting sparse convolutions. Scalable solutions were proposed for resolution [15] and quality [16] progressive coding in fully end-to-end optimized networks, while [17] adopted a hybrid approach by combining a learned residual coding module with G-PCC.

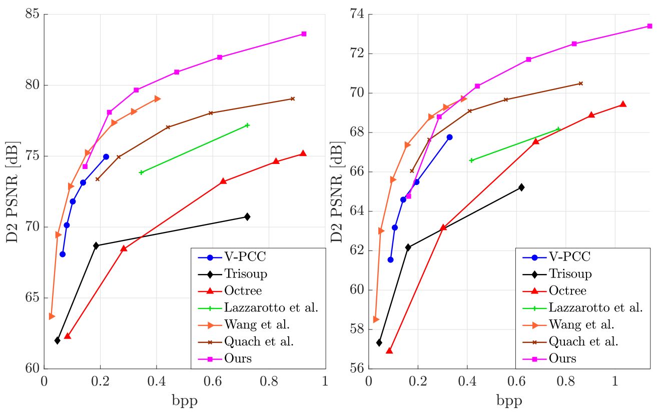
Although many approaches [11, 13, 14, 15, 16, 17] model the entropy on the latent space based only on hyperpriors, we explore redundancies on the latent space by partitioning channel-wise the learned feature block into slices and sequentially leveraging previously encoded slices to help model the scale and mean of the remaining features.

3. PROPOSED METHOD

3.1. Model architecture

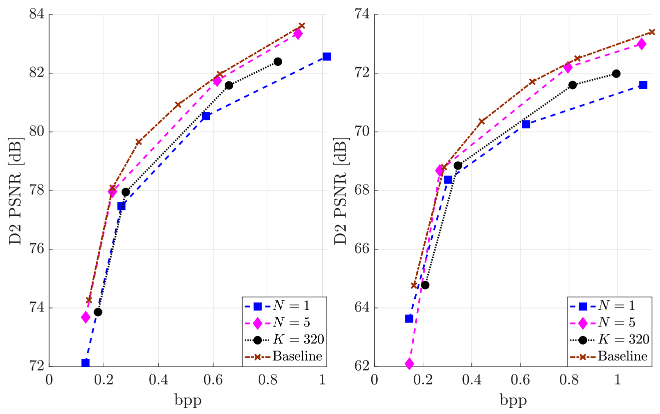
The proposed model consists in a 3D autoencoder architecture with latent entropy modeling. A block diagram of the architecture can be observed in Figure 1. The input x of the model is a cubic block with $k \times k \times k$ voxels that take the value of 1 when occupied and 0 otherwise. The latent representation y is obtained as an output of the encoder, which then undergoes a hyper-analysis transform yielding z. This hyperprior is passed to the bitstream as side information after quantization, and is used to model the entropy of the quantized latent features $\hat{y}$ after going through the hyper-synthesis.

While in other solutions for learning-based point cloud compression the hyperprior would be the only variable used to estimate the scale and mean of $\hat{y}$, we adapt the solution from [7] and additionally use previously decoded channels for entropy modeling. The latent tensor y is sliced along the channel dimension of size K into N non-overlapping, equally sized tensors y_i with $i \in \{1, ..., N\}$. The entropy parameters σ_i and μ_i of each quantized slice $\hat{y}_i$ are estimated using a learned transform taking as input the global entropy param-



(a) *longdress* (b) *phil*

Fig. 2: Comparison of our method against V-PCC, G-PCC (Octree, TriSoup) and other methods based on learned PCC.



(a) *longdress* (b) *phil*

Fig. 3: Rate-distortion plots for ablation study of K and N.

eters σ and μ generated by the hyper-synthesis, as well as the previously decoded slices $\tilde{y}_j\ \forall j \in \{1, ..., i-1\}$. Note that $\tilde{y}_j$ is generated from $\hat{y}_j$ after latent residual prediction, as explained in the next paragraph. Each slice is then independently entropy coded using its own entropy parameters.

At decoder side, we also employ latent residual prediction (LRP) in order to predict the quantization error $y_i - \hat{y}_i$ through a learned transform taking as input the global entropy parameters σ and μ as well as $\hat{y}_i$ and previously decoded slices $\tilde{y}_j\ \forall j \in \{1, ..., i-1\}$. A $\tanh$ non-linearity scaled by a factor 0.5 is applied to the output of the transform to keep the output of the LRP within the range of quantization error. The predicted residuals are then added to $\hat{y}_i$, generating $\tilde{y}_i$. These slices are then concatenated along the channel dimension before going through the synthesis transform. This last learned block finally generates the output block $\tilde{x}$ containing a probability estimation for the occupancy of each voxel.

While the quantization of the latent features y and the hyperprior z at the encoder stage is a necessary step before entropy coding, this operation cannot be performed during backpropagation since its gradient is zero almost everywhere. Traditionally [11, 16, 13], quantization by rounding is replaced by the addition of uniform noise as a proxy function during training to make the operation differentiable. Alternative approaches apply rounding and replace the true gradient by the identity function. We adopt a mixed method [7] which uses the noisy tensor while learning the entropy model and replaces it by a rounded version in subsequent operations. This technique is only used in training, and the rounding function is always applied during inference.

To train the model, a rate-distortion optimization problem represented by the loss function expressed as $\mathcal{L} = R + \lambda D$ is resolved, where R is the estimated rate and D the distortion. The trade-off parameter λ is used to balance the importance of compression rate against reconstruction quality. We leverage the focal loss [18] to compute the distortion, which is expressed by (1).

$$\mathrm{FL}(x_j, \tilde{x}_j) = \begin{cases} -\alpha(1-\tilde{x}_j)^{\gamma}\log(\tilde{x}_j), & \text{if } x_j = 1 \\ -(1-\alpha)\tilde{x}_j^{\gamma}\log(1-\tilde{x}_j), & \text{if } x_j = 0 \end{cases} \quad (1)$$

In this equation, x_j is the binary value of the j^{th} voxel of the input tensor and $\tilde{x}_j$ is its estimated occupancy probability after decoding, while α and γ are configurable hyperparameters.

As a step outside of the training loop, the probability map in $\tilde{x}$ is translated into point cloud blocks. This step is usually accomplished by applying a threshold t and rounding every value above t to 1, and 0 otherwise. While a naive thresholding would set $t = 0.5$ for every block, we adopt an adaptive approach and search for the optimal value $t \in [0, 1]$ which minimises the point-to-point MSE metric [19] between the reconstructed and the original block, finding its own t for each block. This technique was introduced by [11] and has proven to enhance reconstruction quality without significantly impacting the bitrate.

3.2. Dataset and training configuration

For training and testing purposes, the dataset assembled in [20] is used. It consists of 50 point clouds sampled from preexisting datasets, voxelized with a bit depth of 10 for most models, and bit depth of 9 for point clouds with lower point density. The dataset is separated into 44 point clouds for training and 6 for testing following the split originally proposed by the authors. The point clouds are partitioned into blocks of size $64 \times 64 \times 64$ for training, and $128 \times 128 \times 128$ for testing, yielding a total of 13084 and 383 blocks respectively. Furthermore, blocks with less than 500 points in the training set were discarded since they do not carry enough relevant information.

Compression	Metric	*bumbameuboi9*	*guanyin10*	*longdress10*	*phil9*	*rhetorician10*	*romanoillamp10*	Average
G-PCC Octree	D1 PSNR	-6.20	9.23	10.58	7.31	9.53	5.37	5.97
	D2 PSNR	-1.51	7.53	9.15	4.43	8.13	6.77	5.75
Quach et al. [11]	D1 PSNR	-0.97	3.09	3.60	1.77	3.20	1.74	2.07
	D2 PSNR	2.61	3.15	3.65	1.06	2.82	2.34	2.61

Table 1: BD PSNR gain in dB between our method and other compression algorithms.

The training process is conducted using the Adam optimizer with a learning rate of $5 \cdot 10^{-5}$. We employed sequential training to obtain different rate points, which has shown to save time without negative impact on the performance [11].

4. RESULTS

The proposed model was trained with several configurations. We adopted $K = 320$ as in the original work from [7] and the additional value of $K = 160$ for the latent depth. Both $N = 5$ and $N = 10$ were selected for the number of slices, as well as $N = 1$ in order to study a condition without channel-wise entropy modeling. In the focal loss, we employed $\gamma = 2$, and extensive experimentation allowed to define $\alpha = 0.6$ as the best performing value. Different rate-distortion trade-offs were obtained with $\lambda = 40, 200, 1000$ and 1750. All models were tested with both adaptive and fixed thresholding, the latter case with $t = 0.5$. Architectures both with and without the LRP were also trained and assessed.

For evaluation, all blocks from the testing set were compressed, decompressed and merged together to reconstruct the point cloud. The D1 PSNR (point-to-point) and D2 PSNR (point-to-plane) metrics were computed on the reconstructed point clouds using the MPEG software version 0.13.5 [19].

The comparison of the rate-distortion curves revealed that the best performing configuration adopted $K = 160$, $N = 10$, adaptive thresholding and the LRP. Additional models were trained with $\lambda = 400$ and 700 for the selected configuration. For performance comparison, the test set was also compressed with V-PCC intra, both geometry coding modules of G-PCC, namely TriSoup and Octree, as well as three distinct learning-based methods proposed in [11, 14, 17]. The plots of the D2 PSNR metric against the bitrate for two models of the test set are presented in Figure 2. The Bjontergaard-Delta (BD) PSNR comparison to both Octree and the method from [11] was computed for the D1 and D2 metrics and is reported in Table 1 for each test set model.

The proposed method overcomes both G-PCC and [11] in terms of quality in the evaluated test set, as indicated by the average BD PSNR gain and the rate distortion plots. This trend is specially observed for point clouds voxelized with higher bit depth, i.e. the entire set except *bumbameuboi* and *phil*. The comparison to V-PCC and to [14] poses a greater challenge due to the lower bitrate range achieved by those solution, but the plots from Figure 2 indicate that our solution can achieve comparable or superior results at similar bitrates.

In order to observe the impact that different factors exert in the performance of the model, we include in Figure 3 an ablation study for K and N. The baseline corresponds to the plots presented in Figure 2 for our model, while the remaining were tested with the exact same configurations except for the parameters indicated in the legend.

The ablation study demonstrates that the slice partitions play an important role for the obtained performance, since the model with $N = 1$ achieves consistently lower metric values at equivalent bitrates. However, the performance increase from $N = 10$ when compared to $N = 5$ is only marginal, indicating that the latter amount of partitions is already able to minimize inter-channel redundancies. The result for $K = 320$ also suggests that employing an excessively high amount of channels can be detrimental for the performance of the model, and fine tuning the latent depth for the specific case of point cloud compression is necessary.

Among other techniques, while we observed adaptive thresholding played a crucial role on the obtained performance, the changes provoked by the removal of the LRP were minimal, suggesting that equivalent transformations are already effectively learned by the synthesis block. Therefore, an optimal implementation of this method would not include the LRP block, due to the associated complexity increase.

5. CONCLUSION

This paper introduces sequential channel-wise entropy conditioning for learning-based point cloud compression. The method also adopts adpative thresholding and mixed training quantization methods in an end-to-end optimized architecture, outperforming both conventional approaches and other learning-based compression methods with BD PSNR gains of 5.75 and 2.61dB when compared to G-PCC Octree and [11], respectively. The ablation study also provides insights on the impact of latent space partitioning for point cloud coding. The performance gain from this approach indicates that the latent features generated by current learning-based compression methods carry redundancy that can be explored by the proposed architecture. Future studies can focus on other techniques to decorrelate channels in the learning process and naturally reduce channel-wise redundancy.

6. REFERENCES

[1] Julius Kammerl, Nico Blodow, Radu Bogdan Rusu, Suat Gedikli, Michael Beetz, and Eckehard Steinbach, “Real-time compression of point cloud streams,” in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 778–785.

[2] K. Mammou, A. M. Tourapis, D. Singer, and Y. Su, “Video-based and hierarchical approaches point cloud compression,” ISO/IEC JTC1/SC29/WG11 Doc. M41649, Oct. 2017.

[3] MPEG Systems, “Text of ISO/IEC DIS 23090-18 Carriage of Geometry-based Point Cloud Compression Data,” ISO/IEC JTC1/SC29/WG03 Doc. N0075, Nov. 2020.

[4] MPEG 3D Graphics Coding, “Text of ISO/IEC CD 23090-5 Visual Volumetric Video-based Coding and Video-based Point Cloud Compression 2nd Edition,” ISO/IEC JTC1/SC29/WG07 Doc. N0003, Nov. 2020.

[5] Johannes Ballé, Valero Laparra, and Eero P Simoncelli, “End-to-end optimized image compression,” in *International Conference on Learning Representations, ICLR 2017, Toulon, France*, 2017.

[6] Johannes Ballé, David Minnen, Saurabh Singh, Sung Jin Hwang, and Nick Johnston, “Variational image compression with a scale hyperprior,” in *International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada*, 2018.

[7] David Minnen and Saurabh Singh, “Channel-wise autoregressive entropy models for learned image compression,” in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 3339–3343.

[8] A. Dricot and J. Ascenso, “Hybrid octree-plane point cloud geometry coding,” in *2019 27th European Signal Processing Conference (EUSIPCO)*, 2019, pp. 1–5.

[9] E. Pavez, P. A. Chou, R. L. de Queiroz, and A. Ortega, “Dynamic polygon clouds: representation and compression for VR/AR,” *APSIPA Transactions on Signal and Information Processing*, vol. 7, pp. e15, 2018.

[10] Xuanzheng Wen, Xu Wang, Junhui Hou, Lin Ma, Yu Zhou, and Jianmin Jiang, “Lossy geometry compression of 3d point cloud data via an adaptive octree-guided network,” in *2020 IEEE International Conference on Multimedia and Expo (ICME)*, 2020, pp. 1–6.

[11] M. Quach, G. Valenzise, and F. Dufaux, “Improved Deep Point Cloud Geometry Compression,” in *IEEE International Workshop on Multimedia Signal Processing (MMSP’2020)*, Sep. 2020.

[12] M. Quach, G. Valenzise, and F. Dufaux, “Learning convolutional transforms for lossy point cloud geometry compression,” in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 4320–4324.

[13] Jianqiang Wang, Hao Zhu, Haojie Liu, and Zhan Ma, “Lossy point cloud geometry compression via end-to-end learning,” *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 1–1, 2021.

[14] Jianqiang Wang, Dandan Ding, Zhu Li, and Zhan Ma, “Multiscale point cloud geometry compression,” 2020.

[15] André F. R. Guarda, Nuno M. M. Rodrigues, and Fernando Pereira, “Deep learning-based point cloud geometry coding with resolution scalability,” in *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, 2020, pp. 1–6.

[16] André F. R. Guarda, Nuno M. M. Rodrigues, and Fernando Pereira, “Point cloud geometry scalable coding with a single end-to-end deep learning model,” in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020, pp. 3354–3358.

[17] Davi Lazzarotto and Touradj Ebrahimi, “Learning residual coding for point clouds,” in *Applications of Digital Image Processing XLIV*, Andrew G. Tescher and Touradj Ebrahimi, Eds. International Society for Optics and Photonics, 2021, vol. 11842, pp. 223 – 235, SPIE.

[18] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, “Focal loss for dense object detection,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, Los Alamitos, CA, USA, oct 2017, pp. 2999–3007, IEEE Computer Society.

[19] 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)*, 2017, pp. 3460–3464.

[20] Evangelos Alexiou, Kuan Tung, and Touradj Ebrahimi, “Towards neural network approaches for point cloud compression,” *Proceedings Applications of Digital Image Processing XLIII*, vol. 115100, pp. 20. 1151008, 2020.