# Evaluating the effect of sparse convolutions on point cloud compression

Davi Lazzarotto, Touradj Ebrahimi
Multimedia Signal Processing Group (MMSPG)
École Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Switzerland
davi.nachtigalllazzarotto@epfl.ch, touradj.ebrahimi@epfl.ch

***Abstract*—The use of point clouds as an imaging modality has been rapidly growing, motivating research on compression methods to enable efficient transmission and storage for many applications. While compression standards relying on conventional techniques such as planar projection and octree-based representation have been standardized by MPEG, recent research has demonstrated the potential of neural networks in achieving better rate-distortion performance for point cloud geometry coding. Early attempts in learning-based point cloud coding mostly relied on autoencoder architectures using dense convolutional layers, but the majority of recent research has shifted towards the use of sparse convolutions, which are applied only to occupied positions rather than the entire space. Since points are usually distributed on underlying surfaces rather than volumes, such operations allow to reduce the computational complexity required to compress and decompress point clouds. Moreover, recent solutions also achieve better compression efficiency, allocating fewer bits at similar levels of geometric distortion. However, it is not clear to which extent this gain in performance is due to the use of sparse convolutions, if any at all, since the architecture of the model is often modified. In this paper, we conduct an evaluation of the effect of replacing dense convolutions with sparse convolutions on the rate-distortion performance of the JPEG Pleno Point Cloud Verification Model. Results show that the use of sparse convolutions allows for an average BD-rate reduction of approximately 9% for both D1 and D2 PSNR metrics based on similar training procedures, with an even bigger reduction in point clouds featuring reduced point density.**

***Index Terms*—Point cloud compression, sparse convolutions**

## I. Introduction

Immersive imaging modalities have grown in popularity in recent years due to their potential to offer more natural ways to interact with visual content. Applications such as virtual and augmented reality have become increasingly accessible and widely employed in numerous industries such as entertainment, education, training, and healthcare. These modalities enable users to explore and interact with virtual landscapes and objects in ways previously impossible, resulting in more impactful experiences. Because they allow for the correct collection and representation of 3D geometry in a scene, point clouds are a key component of many immersive imaging modalities. Point clouds provide a detailed and high-fidelity representation of the geometry that may be utilized for a range of applications such as autonomous navigation and visualization, by expressing the surface of objects as a collection of 3D points in space.

The vast amount of data needed to represent point clouds is however a major drawback for their use in mainstream applications. For that reason, effective solutions for compression have been heavily researched in recent years. Such efforts have led to the standardisation of two compression algorithms by MPEG, namely geometry-based point cloud compression (G-PCC) [1] and video-based point cloud compression (V-PCC) [2]. While the latter obtain planar projections of both point cloud attributes and geometry as color, depth, as well as occupancy maps and compresses them with conventional video codecs, G-PCC uses an octree to encode voxel occupancy and encodes color with either a region-adaptive hierarchical transform (RAHT) or a lifting transform.

Despite the usefulness demonstrated by conventional techniques, deep learning is receiving increased attention as an alternative approach for point cloud compression. Learning-based solutions have displayed even better rate-distortion performance for geometry data when compared to conventional techniques, allowing for better compression efficiency while maintaining a similar reconstruction quality. Early works in this direction were inspired by architectures previously used for the compression of 2D images, which relied on an autoencoder composed of convolutional layers. The input tensor is downsampled multiple times at the encoder, entropy coded using a probability distribution learned during training, and finally upsampled back to the original resolution at the decoder side. Initial efforts [3], [4] to adapt this algorithm for point clouds represented blocks as dense occupancy maps where all spatial positions are processed by dense 3D convolutions that operate similarly to their 2D counterparts.

However, these approaches fail to take into consideration the nature of most point clouds, which contain points sampled from an underlying surface and therefore occupy a small fraction of the space. Sparse convolutions, on the other hand, allow to better take advantage of these characteristics. In particular, sparse convolutions convolve a 3D kernel over a set of coordinates and apply the weights only at the occupied voxel positions from the input set of coordinates, differently from dense convolutions that convolve the kernel over all

This work was supported by the Swiss National Foundation for Scientific Research (SNSF) under the grant number 200020_207918.

979-8-3503-4218-5/23/$31.00 © 2023 IEEE

indices of a three-dimensional grid and also consider all input positions to produce the input value. Sparse convolutions were leveraged in later compression methods [5], showing not only reduced computational complexity, but also increased rate-distortion performance, and have replaced dense convolutions on recent learning-based compression methods [6], [7].

The majority of recently proposed methods also include modifications to the architecture being used by previous models based on dense convolutions. Therefore, despite the rapid adoption of sparse convolutions as a de facto standard for learning-based voxelized point cloud compression, it is not possible to conclude to which extent the obtained improvements in rate-distortion performance are due to the use of sparse convolutions. The goal of this paper is therefore to assess the isolated impact on compression performance of replacing dense convolutions by these operations. The geometry-only pipeline of the JPEG Pleno Point Cloud verification model [8] is used as a baseline, and the evaluation is conducted using a test set composed of point clouds with different sparsity levels.

While early designs of the verification model also contained joint coding for both geometry and color, recent versions use a separate method for color coding. Moreover, the state of the art contains a much larger number of compression algorithms based both on dense and sparse convolutions for geometry-only coding than for color coding. For those reasons, sparse convolutions are evaluated only for geometry coding in this paper. Since the architecture of the baseline model is in many ways similar to a significant number of works in the state of the art [3]–[5], it is also considered that the results presented in this paper could be similar if other compression methods based on autoencoders were used.

## II. Related work

Early works on point cloud geometry compression used an octree representation as data structure rather than a list of coordinates [9]. The octree became later prevalent with the addition of similar compression algorithms in widespread open-source libraries such as the Point Cloud Library (PCL) [10]. Later works explored pruning the octree, and then representing the leaf nodes as triangular primitives rather than singular points [11]. Both techniques were adopted in the geometry coding module of the G-PCC standard [1]. Other methods aimed to take advantage of the progress made in video compression during the last decades, and projected points onto multiple planes to represent point cloud geometry as two-dimensional maps. Such methods were later explored by the V-PCC standard [2], achieving high rate-distortion performance for dense point clouds, but struggling to effectively compress models with smaller point density.

Point cloud compression algorithms using neural networks were later introduced, with the first works [3], [4] mainly adapting a previous method designed for image compression [12] for three-dimensional representation. Several additional techniques were later studied [13] using previous works as baseline, such as entropy modeling using a hyperprior, adding residual convolutional layers to the encoder and decoder, employing an adaptive threshold selection to translate output occupancy probability into voxels, as well as a sequential training method to allow for coding at different bitrates at reduced training time. Similar techniques were also employed by other authors [14], with results surpassing the rate-distortion performance by G-PCC for dense point cloud models. An autoregressive entropy coding model was also explored with comparable architecture [15], producing even better results that outperformed V-PCC for the evaluated test set. Other techniques, such as block prediction [16] and residual coding [17] explored further extension of similar techniques. Recently, the JPEG standardisation committee launched a call for proposals for learning-based point cloud coding, and a compression method based on dense convolutions was selected as the starting point known by the term verification model [8].

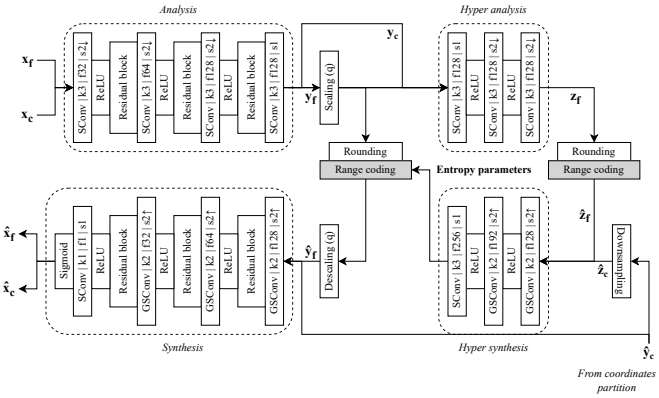



Fig. 1: Block diagram of the model using sparse convolutions. *SConv* denominates regular sparse convolutions and *GSConv* denominate generative sparse transposed convolutions. $k$ denominates kernel size, $f$ the number of features and $s$ the convolution stride. The residual block was implemented with the same parameters as [8]. Scaling consists of the division of the latent features by the quantization step $q$, provided as an encoding parameter and added to the bitstream. Blocks highlighted in gray produce compressed representation added to the bitstream.

Sparse convolutions were first adopted [5] with an architecture similar to the same authors' previous work [14], with the addition of classification and pruning layers for progressive decoding. Another method, denominated as SparsePCGCv1 [6], improved upon the architecture by exploiting cross-scale and same-scale redundancies, allowing for both lossless and lossy compression. Moreover, GRASP-Net [7] proposed a heterogeneous architecture combining sparse convolutions with point-based MLP layers to recover fine details during decoding.

## III. Evaluation conditions

The evaluated compression model is created by replacing all dense convolutions from the baseline model with sparse

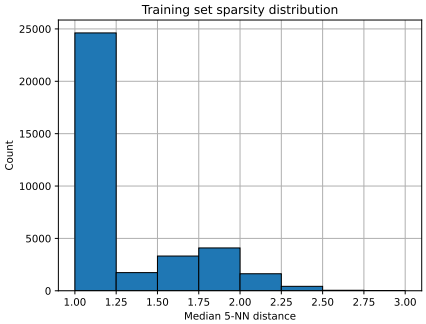



Fig. 2: Histogram of sparsity values of the training set

(a) *Annibal* from CfP original
Median 5-NN distance = 1.17

(b) *kinfudesk* from CfP supplemental
Median 5-NN distance = 2.26

(c) *Lausanne* from swissSURFACE3D — Median 5-NN distance = 3.36

Fig. 3: Point clouds from test set

convolutions, and its diagram block can be observed in Figure 1. The framework used in [8] includes downsampling prior to compression as a strategy to achieve lower bitrates, with learning-based upsampling being applied as a post-processing method. The present evaluation focuses only on the end-to-end autoencoder architecture, therefore setting the sampling factor to 1 and ignoring the advanced block upsampling network.

Prior to compression, the input point cloud is first partitioned into blocks. In the baseline model, the blocks are represented as dense tensors $x$ of dimension $K \times K \times K \times N$, where $K$ corresponds to the block size and $N$ is the number of channels, which is set to 1 for geometry-only coding. The value at a given coordinate of $x$ is set to 1 if the coordinate is present at the input block and to 0 otherwise. On the other hand, the sparse tensors used in the evaluated model are represented by a coordinate tensor $x_c$ of dimensions $N_i \times 3$ set to the coordinates of the input point cloud block, and by a feature tensor $x_f$ of size $N_i \times 1$ with all values set to 1, with $N_i$ being the number of points in the input block.

The output of the analysis transform is a tensor $y$ with its coordinates $y_c$ being equivalent to $x_c$ downsampled three times by a factor of 2. While the features $y_f$ are encoded to the bitstream in a lossy manner by the range coding module and serve to build the input of the synthesis transform $\hat{y}$, the coordinates $y_c$ are losslessly encoder and retrieved at the decoder side. The proposed compression algorithm downsamples the input point cloud geometry prior to block partition by a factor of 8 and compresses it using the lossless settings of the G-PCC codec. During decompression, the G-PCC bitstream is decoded and the obtained coordinates are partitioned into blocks in order to obtain $\hat{y}_c$, which is equivalent to $y_c$.

The synthesis transform takes $\hat{y}$ as input and passes it through three upsampling layers and residual blocks prior to a final sparse convolutional layer that produces a reconstructed tensor $\hat{x}$. Generative layers are employed when upsampling in order to generate new points, which, with kernel size 2, creates 8 output coordinates for each input coordinate corresponding to all possible positions that could have generated the point at the corresponding downsampling layer at the analysis transform.

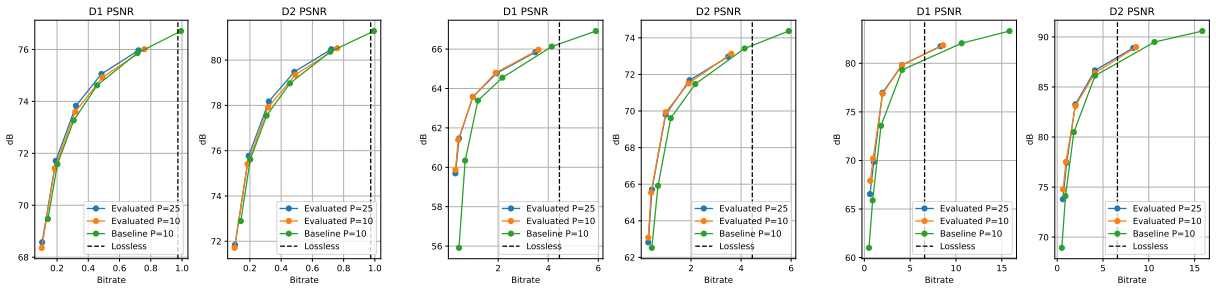
As a result, the decoded tensor $\hat{x}$ usually contains many additional points when compared to the input $x$. Similarly to the baseline model, the coordinates $\hat{x}_c$ are sorted according to the values of $\hat{x}_f$, which represent the estimated probability of occupancy for each coordinate. The decoded point cloud block will then contain the $N_o$ points with the highest occupancy probability, with $N_o$ being defined during compression as the value that maximizes a similarity metric between decoded and input block. During this experiment, the D1 PSNR metric is employed for this purpose.

This model is trained end-to-end to minimize a loss function equivalent to a weighted sum between the estimated bitrate of the compressed features and the distortion between the decoded point cloud block and the input. The rate $R$ is estimated by the sum between the entropy of $\hat{y}_f$ and the entropy of $\hat{z}_f$, while the estimated distortion $D$ is given by the sparse focal loss between $\hat{x}$ and $x$. The latter term is represented in Equation 1, where $\hat{x}_{f_j}$ and $\hat{x}_{c_j}$ correspond to the $j^{th}$ row from $\hat{x}_f$ and $\hat{x}_c$, respectively.

$$\text{FL} = \begin{cases} -\alpha(1-\hat{x}_{f_j})^{\gamma}\log(\hat{x}_{f_j}), & \text{if } \hat{x}_{c_j} \in x_c \\ -(1-\alpha)\hat{x}_{f_j}^{\gamma}\log(1-\hat{x}_{f_j}), & \text{if } \hat{x}_{c_j} \notin x_c \end{cases} \quad (1)$$

The hyperparameter $\alpha$ can be configured to control the weight given to unoccupied voxels relative to occupied ones, while assigning a higher value of $\gamma$ increases the importance given to voxels difficult to classify. The final loss value is given by $L = \lambda R + D$, with the hyperparameter $\lambda$ setting the trade-off between rate and distortion.

The sequential procedure proposed by [13] was used to train the evaluated models in order to obtain different quality levels. In particular, the model with the lowest $\lambda$ is first




(a) *Annibal* (b) *kinfudesk* (c) *Lausanne*

Fig. 4: Rate-distortion plots

trained from scratch, and the obtained weights are used to initialize the training for higher $\lambda$ values. The baseline model was trained with $\lambda$ values following the sequence $\{0.00025, 0.0005, 0.001, 0.002, 0.004, 0.008\}$. For the model with sparse convolutions, higher $\lambda$ values had to be selected to allow for similar bitrates, since the focal loss is computed only on voxels in the neighborhoods of the input points rather than the entire block, given how voxels are generated at the decoder. Such voxels are harder to classify than the vastly empty zones of dense tensors, driving up the relative importance of the $D$ term in the final loss value. Therefore, the sequence $\{0.0025, 0.005, 0.01, 0.025, 0.05\}$ was adopted.

Moreover, a patience parameter $P$ was used to detect convergence of the model during training: if the loss function on the validation set does not decrease after $P$ epochs, then training is stopped and the weights yielding the lowest loss value are selected. Both the baseline and the evaluated models were trained with $P = 10$ for all $\lambda$ values. An additional model with sparse convolutions was trained using a learning rate scheduler, which decreased the learning rate by a factor of 10 whenever the validation loss was not reduced after 10 epochs. In this case, a patience value of $P = 25$ was set. All compression models are coded in PyTorch and were trained with an initial learning rate of $10^{-4}$ using the Adam optimizer, with values of $\alpha = 0.7$ and $\gamma = 2$ in the focal loss.

## IV. Training and Testing Datasets

In order to train both the baseline and the evaluated compression models, the training and validation datasets presented in [8] were used, containing 35861 blocks of size $64 \times 64 \times 64$ obtained from 24 point clouds for training and 3822 blocks from 4 point clouds for validation. Recent works indicate that the performance of compression models depends heavily on the sparsity of the point clouds being compressed. One possible reason for this difference in performance is the distribution of the density values of the training set. In order to evaluate the impact of this factor, a sparsity metric is computed for each block in the training set by measuring the average distance from each point to its 5 nearest neighbors. The median value across all points is selected, denominated as the median 5-NN distance.

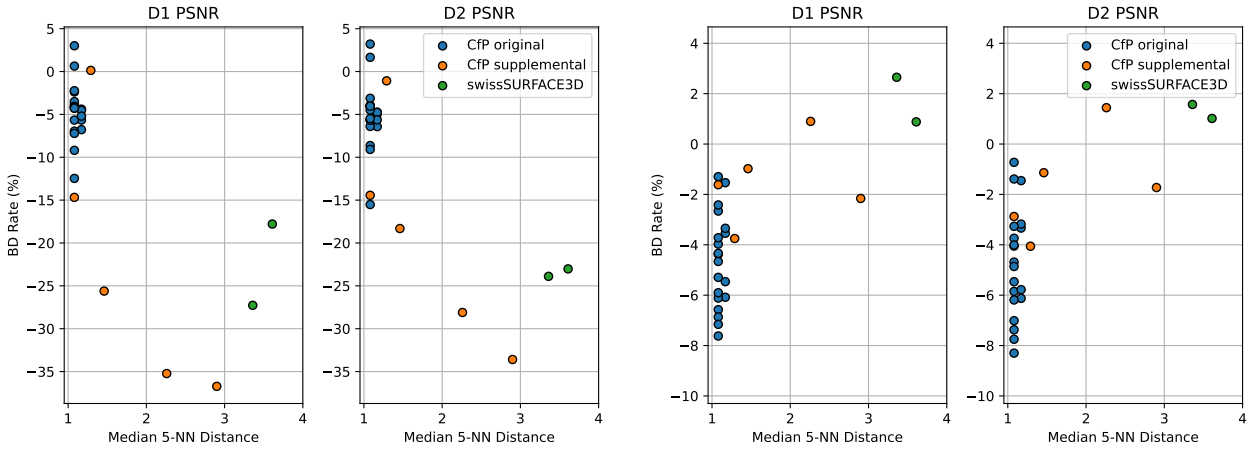
The distribution of this metric for the training set is illustrated in Figure 2. The majority of the point cloud blocks are highly dense, with more than 68% having a median 5-NN distance under 1.25. On the opposite side, less than 6% of the training set is between 2 and 3 in the histogram, and no block with sparsity higher than 3 was employed in the training set. Although the use of a more diverse training set would probably improve the performance of the model at higher sparsity levels, the same dataset was kept to ensure a fair comparison.

In order to test both the baseline and the evaluated compression models, the original test set from the JPEG Pleno Call for Proposals on Point Cloud Coding (CfP) [18] was used. The 20 point clouds were sampled with high density from meshes generated from the acquisition of real-world objects, all of them presenting a median 5-NN distance under 1.25. Additionally, the five point clouds from the supplemental dataset were employed, some of them presenting considerably smaller point densities. Finally, two point clouds obtained from swissSURFACE3D [19] were also included in the test set. This dataset was obtained with airborne LiDAR, with the entire set currently covering more than half of Switzerland. Two geographical regions were selected and the coordinates were voxelized with precision of 13 bits, leading to median 5-NN distances larger than 3. The entire test dataset contained a total of 27 point clouds, three examples of which are presented in Figure 3.

The test set was compressed and decompressed with the baseline model and the two versions of the evaluated model, using a block size of 128 and a latent quantization step of 1. They were additionally compressed using G-PCC software version 21 with lossless settings for comparison. Both point-to-point PSNR (D1 PSNR) and point-to-plane (D2 PSNR) metrics were computed on the decompressed models.

## V. Results and Discussion

The metric values for the evaluated and baseline models were plotted against their bitrates, with a dashed vertical line indicating the bitrate for lossless encoding with G-PCC. The plots for the point clouds illustrated in Figure 3 are presented in Figure 4. For the *Annibal* point cloud, modest gains in performance from the use of sparse convolutions are observed




(a) Comparison between evaluated model with $P = 10$ and the baseline

(b) Difference between values obtained with $P = 25$ and $P = 10$

Fig. 5: BD-Rate reductions for D1 PSNR and D2 PSNR

across the evaluated range. Also, a difference in performance is observed when training the model at a higher patience value, with longer training leading to higher quality. The plots for *kinfudesk* indicate that even larger gains can be obtained for sparser point clouds. Moreover, it is also observed that the difference between different patience levels is reduced, possibly due to the lack of point cloud blocks with similar sparsity levels in the training set. It is also observed that the highest rate of the plot is significantly above the lossless line. Finally, the results obtained for *Lausanne* demonstrate that the model with sparse convolutions outperforms again the baseline, with a higher difference in performance for *Annibal*, but lower than for *kinfudesk*. Moreover, performance between models trained with $P = 10$ and $P = 25$ is again very similar, with even a slight advantage of the former at lower bitrates. Since this point cloud is even sparser than *kinfudesk*, these findings reveal that decreasing point density does not necessarily lead to an increased advantage of the sparse convolutions.

In order to better evaluate the effect of point cloud sparsity on the rate-distortion performance, the BD-Rate between the evaluated model trained with $P = 10$ and the baseline was obtained for each point cloud, ignoring quality levels above the lossless rate for G-PCC. The evaluated model achieved an average bitrate saving of approximately 9%. A consistent increase in performance from the use of sparse convolutions is observed while using the exact same training strategy.

The BD-Rate results were plotted against the median 5-NN distance of each point cloud and are displayed in Figure 5a, the color of each point of the plot indicating the original source of each point cloud. The use of sparse convolutions allowed for lower bitrates for the majority of test set. All point clouds of the original test set of the CfP are grouped at the left of the plot, achieving a BD-rate difference varying between 3% and -15%. Since BD-rate values can vary at a range of approximately 18 percentage points at almost identical point density levels, these results show that sparsity is not the only factor that determines the performance of sparse convolutions. However, the analysis of the CfP supplemental set indicates that sparsity is indeed among the most influential factors, with a high correlation between BD-Rate reduction and median 5-NN distance. In particular, the point exhibiting the highest rate reduction is the most sparse model from this set, achieving more than 36 % savings for the D1 PSNR metric at 2.9 median 5-NN distance. While this strong correlation would suggest that even higher savings should be possible on the sparser models from swissSURFACE3D, this trend was not observed, with overall rate difference remaining between $-17\%$ and -27%. Such results indicate that other factors such as homogeneity or voxelization precision can affect the compression performance as well.

The BD-Rate between the evaluated model trained with $P = 25$ and the baseline model was also computed for the entire set. The difference between these values and those from the previous comparison is presented in Figure 5b. It is observed that including a mechanism for progressively decreasing the learning rate and waiting more epochs prior to stop of the training induced an increase in performance for the majority of the tested point clouds. Indeed, the model trained with $P = 25$ achieved an average BD-Rate difference of approximately -12.5% when compared to the baseline. In particular, point clouds with higher density were more favored by the higher patience value. However, a longer training process was slightly detrimental to the efficiency of the compression of sparser point cloud models, likely because it caused the neural network to specialize for the sparsity values better represented in the training set. Since blocks with a median 5-NN distance higher than 2 account for only a small portion of the training data, higher patience values are not beneficial. Rather than encouraging earlier stops of the training process, these results

indicate the importance of including point cloud models with a wider range of sparsity values in the training set.

Naturally, the obtained results depend also on the evaluation conditions. For instance, the lack of sparse point cloud blocks in the training set probably hinders the performance of both the baseline and the evaluated model. As a matter of fact, no blocks with the same sparsity as *kinfubooks*, nor any models from swissSURFACE3D were used for the optimization of the models. Yet, the models are still capable of encoding such point clouds at rates below lossless with acceptable quality. While these results show the generalization capacity of the neural network to unseen examples, using a more diverse training set would probably increase the rate-distortion performance of such learning-based methods.

Aside from the rate reduction in sparse convolutions, a reduction in computational complexity is also inherently obtained since the convolution operations need to compute at fewer spatial locations. While this feature is already an advantage in itself, it would also allow the use of larger blocks both during training and testing. Indeed, one major limitation of using dense convolutions is their memory usage, which restricts the size of the point cloud blocks that can feed the neural network. While the point cloud size used by compression models based on sparse convolutions is not limitless, higher dimensions could certainly be used due to their smaller memory footprint, likely enabling better performance as previously demonstrated for models using dense convolutions.

Moreover, the hyperparameters for the loss function selected for the training of both the baseline and evaluated model were established by [8], considering only the characteristics of dense convolutions. In particular, the $\alpha$ parameter is set to $0.7$ in order to give a higher weight on the correct classification of occupied voxels due to the fact that most spatial positions of the dense input tensor $x$ are empty. Giving instead the same weight to both occupied and empty voxels would skew the network into producing lower probabilities for the occupancy of most positions due to class imbalance. However, the decoder of the evaluated model produces sparse tensors with coordinates only in the neighborhoods of occupied voxels, not considering regions that are totally empty. Therefore, the optimal $\alpha$ value is likely different from that of the baseline, and adapting this hyperparameter could lead to even better results. These experiments were considered out of the scope of this paper and are deferred to future work.

## VI. Conclusion

In this paper, an evaluation of the performance of sparse convolutions for point cloud geometry compression is conducted by replacing the dense convolutions of an existing compression model with sparse layers, with minimal additional changes. Aside from the intrinsic complexity reduction, an increase in the rate-distortion performance is also observed, with an average BD-Rate reduction of approximately 9% in the evaluated test set. An improved training process also allowed to increase the rate savings to nearly 12.5%. While the improvement of rate-distortion performance is observed for the majority of the point clouds, the sparse convolutions are particularly effective for test models with lower density, with rate savings going up to 35%. The fact that such results were obtained without major adaptations indicates that sparse convolutions are more suitable for point cloud compression in most cases, corroborating the recent shift in research trends.

## References


[1] 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.

[2] 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.

[3] A. F. Guarda, N. M. Rodrigues, and F. Pereira, "Deep learning-based point cloud coding: A behavior and performance study," in *2019 8th European Workshop on Visual Information Processing (EUVIP)*. IEEE, 2019, pp. 34–39.

[4] 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)*. IEEE, 2019, pp. 4320–4324.

[5] J. Wang, D. Ding, Z. Li, and Z. Ma, "Multiscale point cloud geometry compression," in *2021 Data Compression Conference (DCC)*. IEEE, 2021, pp. 73–82.

[6] J. Wang, D. Ding, Z. Li, X. Feng, C. Cao, and Z. Ma, "Sparse tensor-based multiscale representation for point cloud geometry compression," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[7] J. Pang, M. A. Lodhi, and D. Tian, "Grasp-net: Geometric residual analysis and synthesis for point cloud compression," in *Proceedings of the 1st International Workshop on Advances in Point Cloud Compression, Processing and Analysis*, 2022, pp. 11–19.

[8] A. F. Guarda, N. M. Rodrigues, M. Ruivo, L. Coelho, A. Seleem, and F. Pereira, "It/ist/ipleiria response to the call for proposals on jpeg pleno point cloud coding," *arXiv preprint arXiv:2208.02716*, 2022.

[9] R. Schnabel and R. Klein, "Octree-based point-cloud compression," in *Symposium on Point-Based Graphics 2006*, Jul. 2006.

[10] J. Kammerl, N. Blodow, R. B. Rusu, S. Gedikli, M. Beetz, and E. Steinbach, "Real-time compression of point cloud streams," in *2012 IEEE International Conference on Robotics and Automation*, 2012, pp. 778–785.

[11] 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, p. e15, 2018.

[12] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end optimized image compression," *arXiv preprint arXiv:1611.01704*, 2016.

[13] M. Quach, G. Valenzise, and F. Dufaux, "Improved deep point cloud geometry compression," in *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2020, pp. 1–6.

[14] J. Wang, H. Zhu, H. Liu, and Z. Ma, "Lossy point cloud geometry compression via end-to-end learning," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 12, pp. 4909–4923, 2021.

[15] N. Frank, D. Lazzarotto, and T. Ebrahimi, "Latent space slicing for enhanced entropy modeling in learning-based point cloud geometry compression," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 4878–4882.

[16] D. Lazzarotto, E. Alexiou, and T. Ebrahimi, "On block prediction for learning-based point cloud compression," in *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2021, pp. 3378–3382.

[17] D. Lazzarotto and T. Ebrahimi, "Learning residual coding for point clouds," in *Applications of Digital Image Processing XLIV*, vol. 11842. SPIE, 2021, pp. 223–235.

[18] WG1, "Final Call for Proposals on JPEG Pleno Point Cloud Coding," ISO/IEC JTC1/SC29/WG1 Doc. N100097, Jan 2022.

[19] "swisssurface3d," [Accessed: 2022-05-11] https://www.swisstopo.admin.ch/en/geodata/height/surface3d.html.