NF-PCAC: Normalizing Flow based Point Cloud Attribute Compression

Rodrigo Borba Pinheiro, Jean-Eudes Marvie, Giuseppe Valenzise, Frédéric Dufaux

NF-PCAC: NORMALIZING FLOW BASED POINT CLOUD ATTRIBUTE COMPRESSION

Rodrigo B. Pinheiro[1,2] *Jean-Eudes Marvie*[1] *Giuseppe Valenzise*[2] *Frédéric Dufaux*[2]

[1] InterDigital, Inc.
[2]Université Paris-Saclay, CNRS, CentraleSupélec, L2S 91190 Gif-sur-Yvette, France

ABSTRACT

Learning-based point cloud (PC) compression is a promising research avenue to reduce the transmission and storage costs for PC applications. Existing learning-based methods to compress PCs have mainly focused on geometry and employ variational autoencoders to learn compact signal representations. However, autoencoders leverage low-dimensional bottlenecks that limit the maximum reconstruction quality, even at high bitrates. In this paper, we propose a different and novel approach to compress PC *attributes* by using *normalizing flows*. Since normalizing flows model invertible transforms, the proposed approach can achieve better reconstruction quality than variational autoencoders over a large range of bitrates. Our Normalizing Flow-based Point Cloud Attribute Compression (NF-PCAC) outperforms previous learning-based methods for attribute compression, and has comparable performance as G-PCC v.14, showing the potential of this scheme for PC compression.

Index Terms— Point clouds, Learning-Based, Compression, Attributes, Normalizing Flow

1. INTRODUCTION

In the past few years, video content consumption has been evolving towards immersive formats [1], in particular for entertainment. For example, the most recent Olympic Games in Tokyo (2021) featured a 3D replay that brought the spectator closer to the action. Video games new trend is using virtual reality goggles and devices to reproduce 3D worlds and make the experience as immersive as possible. In this context, Point Clouds (PCs) are one of the most popular volumetric representations. PCs are a set of unordered points in space that contain the geometry information, the spatial location of the points in the x, y and z axes, and the associated attribute information, in most cases, color. However, PCs can contain millions of points and are thus expensive to store and transmit. For this reason, to make PCs a viable option to diffuse 3D content, compression is necessary.

The field of PC compression has significantly advanced in the past few years, notably thanks to the standardization efforts in MPEG [2]. In addition to conventional coding techniques, recently a number of learning-based techniques have been proposed for PCs [3]. However, the majority of these techniques focus on the compression of the PC geometry, but not the attributes. In addition, existing methods use variational autoencoders to learn low-dimensional representations for compression. While this is very effective at low bitrates, the intrisically lossy nature of autoencoders limits the maximum reconstruction quality at higher bitrates.

In this paper, we propose the first end-to-end learning based approach that makes use of a *normalizing flow architecture* to encode the PC attributes. We call the proposed approach NF-PCAC for Normalizing Flow Point Cloud Attribute Compression. Normalizing flows are neural networks that model invertible transforms. In contrast to variational autoencoders (VAE), these architectures do not have a low-dimensional bottleneck and can, in principle, achieve lossless reconstructions. We adapt the 2D normalizing flow architecture to take into consideration the 3D nature and the sparsity of PCs and we add some approximations to obtain a better trade off between quality and bitrate. Thus our architecture is no longer fully invertible, but produces state-of-the-art performance and provides higher coding gains than existing learning based attribute compression approaches. It is also the first learning based approach that achieves comparable and in some cases, even better results than G-PCC for attributes.

2. RELATED WORK

The baseline for today's research on PC static compression is the Geometry-based PC Compression (G-PCC) algorithm standardized by MPEG [2]. G-PCC encodes the geometry and attributes separately. Geometry coding uses an octree approach to code the PC, whereas attribute coding is performed by a Region Adaptive Hierarchical Transform (RAHT) [5].

Following the recent success of learning-based coding for 2D images (e.g., [6]), variational autoencoders have been also applied to the compression of PCs, in particular for geometry. In [7], the authors use 3D voxel convolutions, and cast the decoding problem as one of classifying which voxels are non-empty. This initial architecture led to some extensions and improvements, [8, 9]. In particular, [10] introduces the use of sparse convolutions for PC compression, allowing the input to be the entire PC instead of just a partition.

Learning-based coding of PC attributes has been less ex-

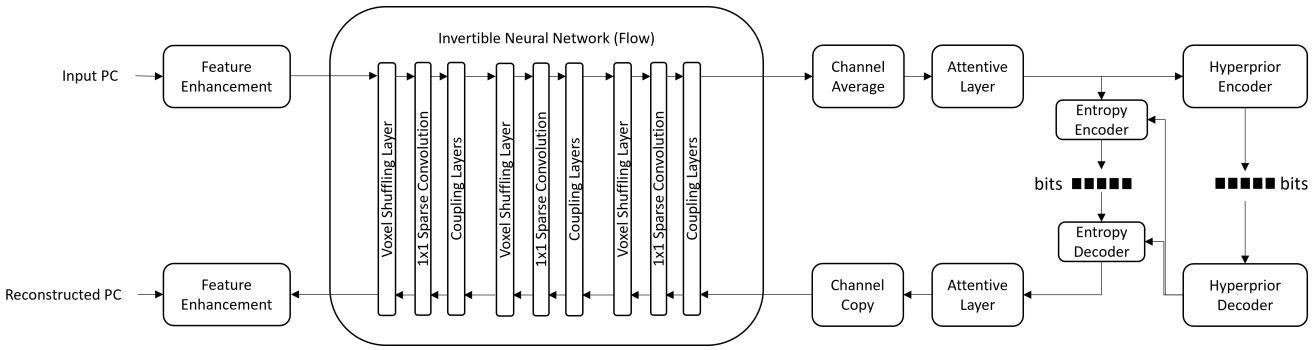


Fig. 1: Proposed PC attribute coding architecture based on normalizing flows. Compared to a 2D normalizing flow architecture for image coding [4], our scheme is adapted to compress PC attributes: all the convolutions are replaced by sparse convolutions, the squeeze layer is replaced by our novel 3D squeeze layer. The PC goes through a feature enhancement block, a normalizing flow module, a channel average and the attention layers before being coded.

plored. In [11], attributes are interpreted as a signal over a 2D manifold and are coded using a conventional image codec. However, this mapping fails when the underlying PC geometry is complex, limiting the coding performance. DeepPCAC [12] employs second-order point convolutions instead of voxel-based convolution, inspired by [13], and a similar architecture to [6] to code the color attributes of PCs. A limitation of point-based convolution is the inability to capture spatial dependencies, which leads to relatively poor coding performance. A recent method proposed in [14] consists of an extended version of [10] using a similar architecture and sparse voxel convolutions to compress the attributes of PCs. The use of sparse convolutions is interesting for PCs learning based tasks because it can help overcome the limitations of available memory as well as avoid the dilation of the features to empty spaces [15]. This work is currently achieving state-of-the-art results for learning-based attribute coding.

Normalizing flow architectures have initially been proposed as generative models [16] and later applied to image compression [17]. An interesting property of normalizing flows is that they produce invertible representations, similar to the orthogonal transforms used in conventional coding. To obtain competitive performance at low bitrates, the authors of [4] introduce attentive squeezing layers to reduce the dimensionality of the latent representations and increase the compression ratio of images. In this paper, we adapt this architecture to the case of 3D point cloud attributes.

3. PROPOSED METHOD

3.1. Normalizing Flow PC Attribute Compression

We propose a novel method to compress PC attributes using a normalizing flow based architecture. Normalizing flow architectures transform the input signal using a diffeomorphic and orientation preserving function [18]. A function is diffeomorphic if it is completely invertible and differentiable. In other words, the transformation necessary to decode the original data from the latent space consists in inverting the operations of the encoder, with the same trained weights. To obtain a bijective mapping, the latent space must have the same dimension as the original data space. This is in contrast with variational autoencoders, where an information bottleneck forces the latent space to be low-dimensional. Initially proposed for image generation [16], normalizing flows have been less popular for lossy image compression [17, 4], where the main paradigm is still VAE [6]. Nevertheless, normalizing flows have the potential to provide better performance than VAE at higher bitrates, avoiding or reducing the typical quality saturation in autoencoders.

To achieve good compression performance over a large range of bitrates, we propose the NF-PCAC scheme depicted in Figure 1, which extends the architecture presented in [4] to point cloud attributes. It contains a feature enhancement block, an invertible neural network (INN), a channel average block and an attentive layer, followed by a hyperprior model [19] for the entropy coding of the coefficients in the latent space.

The feature enhancement block, composed of dense blocks as defined in [20], has the goal to help extract more non-linear features from the original PC. The use of dense blocks with skip connections helps preserve the original features in the output of the block, and also transforms the signal so that it can be better encoded. We extend this block to use sparse 3D convolutions instead of 2D regular convolutions to increase memory efficiency and avoid noise in feature computation due to convolutions with empty voxels.

The INN is the core of the architecture as it computes the latent features. It is composed by completely invertible operations, meaning that the output of this block when processed by the reverse operations will result in the original input in a lossless fashion. It is constituted of 3 repetitions of the sequence: 3D voxel squeeze layer, sparse 1×1 convolution and sparse coupling layers. The coupling layers consist on a sequence of operations as defined in [16], designed to obtain a bijective function that is flexible and easy to invert.

Following the INN and the average channel layer (see be-

low), we have an attentive layer [21] that consists of convolutions followed by a sigmoid function with the goal of helping the architecture focus on more important areas in the PC. For the entropy modeling, we use a hyperprior to generate a mean and a scale for the distribution such as [6, 19].

3.2. 3D squeeze layer

The normalizing flow architecture needs to be adapted to take into account the sparse nature of 3D PCs. A key module in the architecture is the *squeeze layer* to effectively reduce the spatial dimension and increase the number of channels, without changing the overall dimensionality of the feature space. This step is essential to increase the receptive field for the convolutions, and the squeeze operation serves this purpose in the normalizing flow, converting spatial information into channel information and using it for subsequent convolutions.

The squeezing operation trades spatial size for channels by implementing a masking scheme for 2D images [16], where the features in a 2×2 neighborhood are all aggregated as channels of the first spatial location, reducing the number of pixels by a factor of 4 and increasing the number of features also by a factor of 4. The extension for 3D data comes naturally by just increasing the neighborhood to a $2 \times 2 \times 2$ region, reducing the number of pixels by a factor of 8 and increasing the number of channels by the same factor (see Figure 2). However, this naive extension in a sparse PC would generate empty channels, impacting subsequent convolutions, and increasing the number of coefficients, since we would add new voxels to the original data. To reduce the number of coefficients, a channel averaging layer is typically used after the normalizing flow blocks. This layer reduces the number of channels by a factor of 2 by taking the average values of these channels. However, by using this strategy alone in the architecture we worsen the problems caused by the empty channels produced in the squeezing layers. These empty channels cause the learning process to suffer a big slowdown, since they do not contribute any information.

To solve this problem we develop a *3D voxel squeeze layer*, derived from the squeezing operation presented in [16]. The newly proposed layer uses artificially filled points in the PC to replace the zeros and improve the convergence of the algorithm, reducing at the same time the artifacts due to filtering across empty voxels. Specifically, we propose to fill the empty voxels using the average attribute values of all the occupied voxel neighbors in the region to be squeezed. An example illustration of empty voxel filling is given in Figure 2b, where we have an average value of $(1+4+6+7)/4 = 4.5$ and the output of the squeeze layer is displayed on the right side of the arrow in Figure 2b.

4. EXPERIMENTS AND RESULTS

In this section we detail the training procedure and present a comparison between our method and other learning-based attribute compression methods, as well as the last available version of G-PCC (TMC13v14). For all the cases we assume the geometry is losslessly coded in a different bitstream.

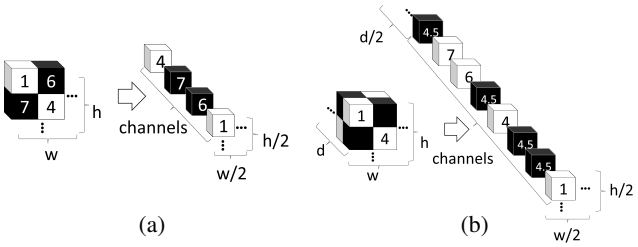


Fig. 2: Squeeze Layer implementation; (a) The original squeeze layer implemented in 2D. (b) The 3D extension of the squeeze layer on sparse data. Here, the voxels in black are empty and are replaced by the average of the non-empty neighbors after the squeezing.

4.1. Training

The training set is a mix of different PCs that represent real characters from 8i [22], owlii [23], volucap and xdprod [24], as well as PCs generated from axyz [25] textured mesh bundles, sampled to produce PCs of roughly 1,000,000 points, a comparable density to the other PCs in our dataset.

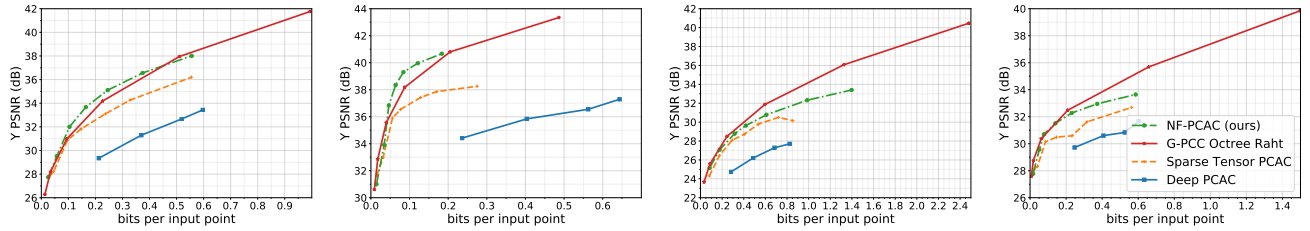
We retain for tests and validation the PCs Soldier [22], Basketball Player [23], Thomas [24], and Facade [2] while all the other sequences are used for training. We repeat the same protocol using Longdress and Redandblack [22], as test set for a broader validation. We choose a percentage of the frames in each sequence to be used in our framework. To compensate for the lack of available data for training, an octree partition of the chosen frames was performed, generating blocks of size $128 \times 128 \times 128$. With this block size we manage to produce a dataset with enough variety of data and the training process becomes less memory consuming, allowing for larger batches. We obtained 32809 training blocks and 5413 validation blocks. We trained our architecture with no data augmentation for 20 epochs with a learning rate of $1e-4$ and a batch size of 8. The training was performed in a Tesla V100 with 32 GB of memory.

4.2. Validation

We trained a network with the same architecture in the same conditions for one bitrate point, using a naive squeeze layer instead of our solution to validate our contribution. Our 3D squeeze layer converges to a smaller loss, meaning we obtain better PSNR results with fewer bits per occupied voxel, as seen in Table 1.

	Soldier		Basketball		Facade		Thomas	
	Naive	Ours	Naive	Ours	Naive	Ours	Naive	Ours
PSNR	35.20	**37.90**	37.26	**40.56**	33.00	**33.65**	32.62	**34.38**
Bits per input point	0.59	**0.56**	0.30	**0.18**	0.69	**0.59**	0.43	**0.39**

Table 1: Ablation study: 3D squeeze layer vs. naive approach



(a) Soldier - 8iVFB (b) Basketball Player - Owlii (c) Longdress - 8iVFB (d) Facade - MPEG

Fig. 3: PSNR curves for four different PCs.

(a) Original PC (b) GPCC 30.95 dB, **0.094 bpp** (c) Deep-PCAC 29.35 dB, 0.213 bpp (d) Sparse Tensor PCAC 30.89 dB, 0.097 bpp (e) NF-PCAC (ours) **32.00 dB**, 0.104 bpp

Fig. 4: Visual results of all the different architectures tested

During inference, to avoid blocking artifacts that occur when coding the divided PC, and since we have a network that is completely convolutional, the input of the network is the complete PC, instead of the blocks as used for training.

We compare the performance of the proposed method with those obtained with a VAE-type architecture similar to the one presented in [14]. For a fair comparison, we employ the same hyperprior as used in our architecture. We call this method Sparse Tensor PCAC. This baseline was trained with the same dataset, learning rate and batch size. We also present the results of Deep-PCAC, the architecture used in [12] with the available weights in their official repository. The PSNR results were generated using the MPEG metric software [26].

Rate-distortion results are presented in Figure 3 for 4 PCs, with the Y-PSNR as quality metric. As observed, our method outperforms all others learning-based methods for similar bitrates. However, when it comes to higher bitrates, G-PCC is still the best performing approach. We also present BD-Rate and BD-PSNR in Table 2, some qualitative results in Figure 4 and the inference time on CPU and GPU in Table 3. G-PCC manages to better represent higher frequencies in smaller areas, such as in the soldier's helmet, which might explain the weaker performance on point clouds Facade and Longdress. This behavior might be due to the average calculations that are performed in our architecture and the channel average layer that is used to reduce the number of channels. However, our method obtains better visual results in the more accentuated contours such as the fingers or the end of the sleeves, whereas in G-PCC these areas are more noisy.

	G-PCCv14		Sparse Tensor PCAC		Deep PCAC	
	BD-R-Y	BD-PSNR-Y	BD-R-Y	BD-PSNR-Y	BD-R-Y	BD-PSNR-Y
Soldier	10.05	-0.3	45.18	-1.17	308.52	-5.22
Basketball Player	-2.15	0.07	40.73	-1.43	870.11	-6.84
Longdress	-20.96	0.96	26.93	-0.67	276.22	-3.85
Facade	-27.64	0.58	80.35	-0.9	402.75	-2.53
Average	-10.18	0.33	**48.30**	**-1.04**	**464.40**	**-4.61**

Table 2: BD-Rate (%), BD-PSNR (dB) - other methods against ours

PC size	G-PCCv14 CPU		Sparse Tensor CPU		Sparse Tensor GPU		NF-PCAC (Ours) CPU		NF-PCAC (Ours) GPU	
	Enc	Dec	Enc	Dec	Enc	Dec	Enc	Dec	Enc	Dec
10-bits	3,23	3,15	5,08	4,06	0,40	0,26	49,19	48,75	1,36	1,25
11-bits	14,19	11,67	20,47	16,73	1,13	0,76	172,89	171,08	4,08	3,69

Table 3: Average inference time for test models on CPU (Intel®. i9-9900K - 3.6GHz, 64GB RAM) and on GPU (Nvidia®. Tesla ®. V100, 32 GB)

5. CONCLUSION AND FUTURE WORKS

We explore the use of normalizing flows as a tool for compressing point cloud attributes. We first adapt a 2D image coding architecture based on normalizing flows to the case of 3D PCs. Then, we propose a simple yet effective 3D squeeze layer that avoids filtering over empty voxels by filling the non-occupied space with local attribute averages. More sophisticated designs for the squeeze layer are currently under study.

Despite being a first attempt to replace the widely used VAE with normalizing flows in PC compression, the proposed approach obtains good coding gains compared to previous learning-based compression methods for PC attributes. These preliminary results support the conclusion that invertible transforms such as normalizing flows have a significant potential for PC coding and represent a promising research direction for future work.

6. REFERENCES

[1] G. Valenzise, M. Alain, E. Zerman, and C. Ozcinar, *Immersive Video Technologies*, Elsevier, Oct. 2022.

[2] 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 Trans. Signal and Inf. Process.*, 2020.

[3] M. Quach, J. Pang, T. Dong, G. Valenzise, and F. Dufaux, "Survey on Deep Learning-based Point Cloud Compression," *Frontiers in Signal Processing*, 2022.

[4] Y. Xie, K. L. Cheng, and Q. Chen, "Enhanced invertible encoding for learned image compression," *29th ACM International Conference on Multimedia*, 2021.

[5] R. L. de Queiroz and P. A. Chou, "Compression of 3d point clouds using a region-adaptive hierarchical transform," *IEEE Transactions on Image Processing*, 2016.

[6] D. Minnen, J. Ballé, and G. D. Toderici, "Joint autoregressive and hierarchical priors for learned image compression," in *Advances in Neural Information Processing Systems*, 2018.

[7] M. Quach, G. Valenzise, and F. Dufaux, "Learning convolutional transforms for lossy point cloud geometry compression," *2019 IEEE International Conference on Image Processing (ICIP)*, 2019.

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

[9] J. Wang, D. Ding, Z. Li, and Z. Ma, "Multiscale point cloud geometry compression," in *2021 Data Compression Conference (DCC)*, 2021.

[10] J. Wang, D. Ding, Z. Li, X. Feng, C. Cao, and Z. Ma, "Sparse tensor-based multiscale representation for point cloud geometry compression," 2021.

[11] M. Quach, G. Valenzise, and F. Dufaux, "Folding-based compression of point cloud attributes," in *2020 IEEE International Conference on Image Processing (ICIP)*, 2020.

[12] X. Sheng, L. Li, D. Liu, Z. Xiong, Z. Li, and F. Wu, "Deep-PCAC: An end-to-end deep lossy compression framework for point cloud attributes," *IEEE Transactions on Multimedia*, 2021.

[13] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems*, 2017.

[14] J. Wang and Z. Ma, "Sparse tensor-based point cloud attribute compression," in *2022 IEEE 5th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, 2022.

[15] B. Graham, M. Engelcke, and L. van der Maaten, "3d semantic segmentation with submanifold sparse convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[16] L. Dinh, J. Sohl-Dickstein, and S. Bengio, "Density estimation using real NVP," in *International Conference on Learning Representations*, 2017.

[17] L. Helminger, A. Djelouah, M. Gross, and C. Schroers, "Lossy image compression with normalizing flows," in *Neural Compression: From Information Theory to Applications – Workshop @ ICLR 2021*, 2021.

[18] L. Ruthotto and E. Haber, "An introduction to deep generative modeling," *GAMM-Mitteilungen*, 2021.

[19] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," 2018.

[20] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[21] Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Learned image compression with discretized Gaussian mixture likelihoods and attention modules," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[22] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, "8i voxelized full bodies - a voxelized point cloud dataset," ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006, Geneva, 2017.

[23] Y. Xu, Y. Lu, and Z. Wen, "Owlii dynamic human mesh sequence dataset," ISO/IEC JTC1/SC29/WG11 m41658, 120th MPEG Meeting, Macau, 2017.

[24] R. Schaefer, P. Andrivon, J. Ricard, and C. Guede, "Volucap and XD Productions Datasets," Tech. Rep. M56192, ISO/IEC JTC1/SC29/WG7(MPEG), 2021.

[25] "Axyz design 3d people models and character animation software," https://secure.axyz-design.com - Acessed Mar. 03, 2023.

[26] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, "Updates and integration of evaluation metric software for PCC," Tech. Rep. M40522, ISO/IEC JTC1/SC29/WG11(MPEG), 2017.