Improved Deep Point Cloud Geometry Compression

Maurice Quach*, Giuseppe Valenzise*, Frederic Dufaux*

*Université Paris-Saclay, CNRS, CentraleSupélec, Laboratoire des signaux et systèmes

91190 Gif-sur-Yvette, France

Abstract—**Point clouds have been recognized as a crucial data structure for 3D content and are essential in a number of applications such as virtual and mixed reality, autonomous driving, cultural heritage, etc. In this paper, we propose a set of contributions to improve deep point cloud compression, i.e.: using a scale hyperprior model for entropy coding; employing deeper transforms; a different balancing weight in the focal loss; optimal thresholding for decoding; and sequential model training. In addition, we present an extensive ablation study on the impact of each of these factors, in order to provide a better understanding about why they improve RD performance. An optimal combination of the proposed improvements achieves BD-PSNR gains over G-PCC trisoup and octree of 5.50 (6.48) dB and 6.84 (5.95) dB, respectively, when using the point-to-point (point-to-plane) metric. Code is available at https://github.com/mauriceqch/pcc_geo_cnn_v2.**

Index Terms—**point clouds, compression, neural networks, geometry, octree**

I. INTRODUCTION

Due to recent advances in visual capture technology, point clouds have been recognized as a crucial data structure for 3D content. In particular, point clouds are essential for numerous applications such as virtual and mixed reality, sensing for autonomous vehicle navigation, architecture and cultural heritage, etc. Point clouds are sets of 3D points identified by their coordinates, which constitute the geometry of the point cloud. In addition, each point can be associated with attributes like colors, normals and reflectance. Point clouds can have a massive number of points, especially in high precision or large scale captures. This entails a huge storage and transmission cost. As a result, Point Cloud Compression (PCC) is fundamental in practice.

The Moving Picture Experts Group (MPEG) is planning to release two PCC standards [1]: Geometry-based PCC (G-PCC) and Video-based PCC (V-PCC). G-PCC approaches PCC from a 3D perspective and compresses point clouds in their native form using 3D data structures such as octrees. On the other hand, V-PCC approaches PCC from a 2D perspective, projects 3D data onto a 2D plane and makes use of video compression technology. In order to evaluate test models, common test conditions (CTCs) [2] were designed. In this context, the point-to-point (D1) and the point-to-plane [3] quality metrics (D2) are used for quantitative evaluation. Recently, deep point cloud compression (DPCC) methods have been proposed and shown to provide significant coding gains compared to traditional methodologies [4], [5].

In this paper, we focus on lossy compression of static point cloud geometry using deep convolutional networks. Specifically, we propose a set of contributions to improve RD performance and accelerate model training. We then present an ablation study identifying key performance factors for DPCC. In particular, we start from a baseline DPCC model [4] and we consider the following improvements:

- *Entropy modeling*: we consider an hyperprior model to improve entropy coding.
- *Deeper transforms* that compensate downsampling with progressively higher numbers of filters.
- *Changing the balancing weight in the focal loss*: similar to [4], we cast decoding as an unbalanced classification problem by optimizing a focal loss [6]. Hence, we study the RD performance impact of the focal loss α parameter.
- *Optimal thresholding for decoding*: in order to classify voxels as occupied or not, we propose an optimal thresholding approach that minimizes a given distortion metric (instead of a fixed threshold as in [4]).
- *Sequential training*: in order to reduce the computational complexity of training a network for each RD tradeoff, we propose a sequential training procedure. That is, we train a network corresponding to a given RD point by fine tuning the network trained from the previous RD point. This makes training times up to 8 times faster compared to training independently and improves RD performance.
- *Ablation study* An extensive ablation study evaluating the impact of each factor mentioned above on RD performance. The evaluated conditions are detailed in Table I.
- *Octree partitioning* An efficient octree partitioning algorithm that is significantly faster compared to recursive octree partitioning.

II. RELATED WORK

Our research is related most closely to three research areas: static point cloud geometry compression, deep image compression and deep point cloud compression.

Static point cloud geometry compression methods are usually based on the octree structure [7]. Indeed, octrees provide an efficient way of partitioning the 3D space and representing point clouds. In particular, they are especially suitable for lossless coding in combination with octree entropy models [8]. However, lossy compression using octrees alone has poor performance as pruning octree levels decreases the number of points exponentially resulting in significant distortion. To alleviate this issue, many solutions have been proposed such as triangle [9] surface models, planar [10] surface models,

Experimental conditions evaluated in this study. Each
condition is an improvement over the previous one.

Name	Model	Transforms	α	Threshold	Training
c1	Baseline	Shallow	0.90	Fixed	Independent
c2	Hyperprior	—	—	—	—
c3	—	Deep	—	—	—
c4	—	—	0.75	—	—
c5	—	—	—	Optimal	—
c6	—	—	—	—	Sequential

(a) Baseline model. (b) Hyperprior model.

Fig. 1: Entropy models considered in this work. The f functions are learned transforms, Q refers to quantization and AC to arithmetic coding with its associated density model.

graph-based enhancement layers [11] and volumetric functions [12]. The core idea is that by encoding approximations along a coarse octree, we can alleviate the shortcomings of the octree structure. Different from previous work in this area, we study learned approximation models based on deep neural networks.

Deep image compression considers the use of deep neural networks for image compression. An end-to-end image compression solution with joint RD optimization along with a learned entropy model has been proposed in [13], which also replaces (non-differentiable) quantization with uniform noise at training time. As a follow-up of that work, a scale hyperprior model has been proposed in [14]. The scale hyperprior enables the modeling of spatial correlations in the latent space; for each element, it uses a Gaussian distribution whose standard deviation is predicted by a dedicated network. We design models for PCC using these learning-based entropy modeling techniques.

DPCC is a recent research avenue exploring the use of deep neural networks for PCC. For lossy geometry coding, voxel-based DPCC methods have been shown to outperform traditional methods significantly [4], [5], [15]. For lossless geometry coding, deep neural networks have been used to improve entropy modeling [16]. Also, DPCC for attributes has been explored by interpreting point clouds as a 2D discrete manifold in 3D space [17]. Closely related to our study, the behavior and performance of DPCC methods has been investigated in [5]. However, this particular study investigates the characteristics and RD impact of the latent space. In contrast, we seek to understand and identify key performance factors for rate-distortion (RD) performance on a larger scale.

III. PROPOSED IMPROVEMENTS

In this section we present different strategies to improve DPCC. We consider as baseline the network proposed in our preliminary work [4] (denoted as **c1** in the following). In that work, we relied on shallow transforms to compress entire point clouds at once. However, this has a fundamental limitation in terms of memory usage, as it does not allow to compress large point clouds as those commonly used in MPEG CTCs. Therefore, in this work we make use of octree partitioning to partition point clouds into blocks of size $64 \times 64 \times 64$ voxels, which we have found to be a good trade-off between memory usage and coding performance. In the rest of the paper, we denote the different considered improvements with **c2**,..., **c6**, which are summarized in Table I.
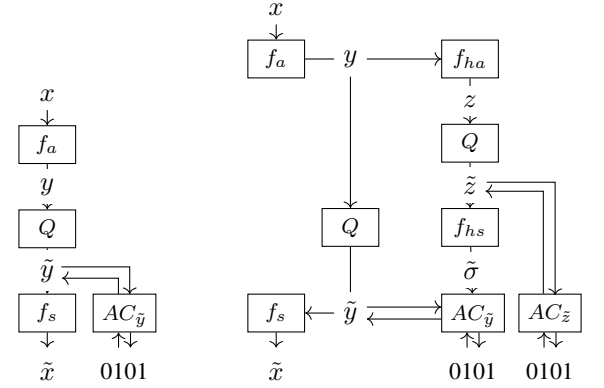
A. Entropy modeling (c2)

We consider models that take the voxelized point clouds x and $\tilde{x}$ as input and output. In particular, we consider a baseline model (Fig. 1a) and an hyperprior model (Fig. 1b).

The baseline model is based on an autoencoder architecture with an analysis f_a and a synthesis transform f_s [13]. y is modeled using a learned entropy model for each feature map. The baseline model is expressed as follows

$$y = f_a(x) \qquad \tilde{y} = Q(y) \qquad \tilde{x} = f_s(\tilde{y}). \qquad (1)$$

We consider a scale hyperprior model [14] as a better entropy model for $\tilde{y}$. Specifically, we model y with a zero-mean gaussian density model $\mathcal{N}(0, \tilde{\sigma}^2)$ where standard deviations $\tilde{\sigma}^2$ are predicted from y with $\tilde{\sigma} = f_{hs}(Q(f_{ha}(y)))$. As a result, the spatial dependencies can be modeled better compared to the learned entropy model. The hyperprior model is expressed as follows

$$y = f_a(x) \qquad \tilde{y} = Q(y) \qquad \tilde{x} = f_s(\tilde{y}) \qquad (2)$$
$$z = f_{ha}(y) \qquad \tilde{z} = Q(z) \qquad \tilde{\sigma} = f_{hs}(\tilde{z}) \qquad (3)$$

where z is modeled with a learned density model for each feature map.

The compression model is trained using joint RD optimization with the loss function $R + \lambda D$. For each RD tradeoff, we train a model with the corresponding λ value resulting in transforms and entropy models specialized for this particular tradeoff. The entropy R is computed on $\tilde{y}$, and $\tilde{z}$ for the hyperprior model, using their associated entropy models. Since the quantization operation Q is not differentiable, we use additive uniform noise during training in place of quantization as originally proposed in [13].

B. Deeper transforms (c3)

We compare shallow and deep transforms for analysis and synthesis, as illustrated in Fig. 2. Specifically, we focus on

C, N, 9, ↓ 2	AB, N_1, ↓ 2	C, N, 3
C, N, 5, ↓ 2	AB, N_2, ↓ 2	C, N, 3, ↓ 2
C, N, 5, ↓ 2	AB, N_3, ↓ 2	C, N, 3
	C, N_3, 3	

(a) Shallow Analysis (b) Deep Analysis (c) Hyper analysis

C^T, N, 5, ↑ 2	SB, N_3, ↑ 2	C^T, N, 3
C^T, N, 5, ↑ 2	SB, N_2, ↑ 2	C^T, N, 3, ↑ 2
C^T, N, 9, ↑ 2	SB, N_1, ↑ 2	C^T, N, 3
	C, 1, 3	

(d) Shallow Synthesis (e) Deep Synthesis (f) Hyper synthesis

C, N, 3, ↓ 2
C, N, 3
C, N, 3
+

C^T, N, 3, ↑ 2
C^T, N, 3
C^T, N, 3
+

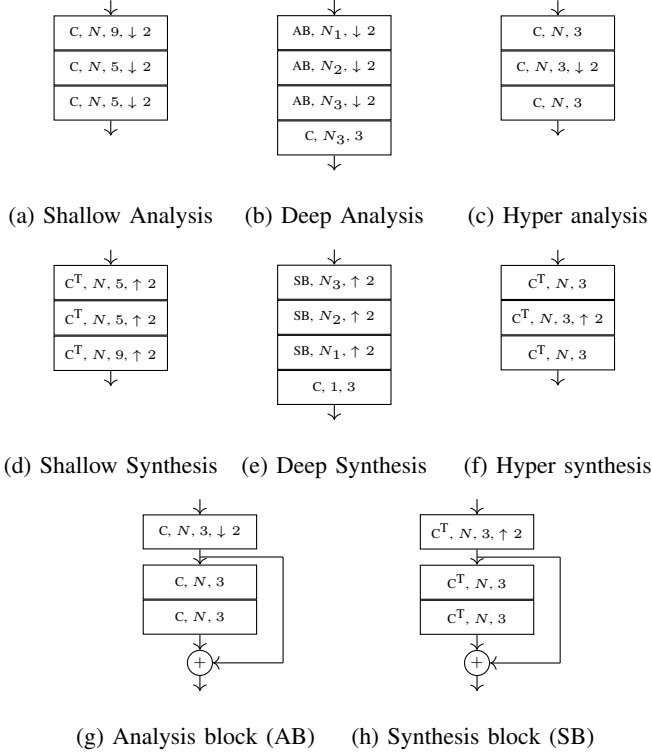(g) Analysis block (AB) (h) Synthesis block (SB)

Fig. 2: Transform types. Each layer is specified as follows: convolution type (C refers to convolution, C^T to transposed convolution), number of filters, filter size and strides.

analysis and synthesis transforms and use shallow hyper-analysis and hyper-synthesis transforms (Fig. 2c and 2f).

The transforms based on 3D convolutions and 3D transpose convolutions introduced in [4] are referred to as shallow transforms (Fig. 2a and 2d). We introduce *deeper* variants of shallow transforms (Fig. 2b and 2e), which we refer to as deep transforms. These transforms are composed of residual [18] blocks (Fig. 2g and 2h) which use skip-connections to prevent issues such as exploding or vanishing gradients. The skip-connections act as "shortcuts" in the network allowing gradients to backpropagate through shorter paths. We also make them *progressive* by increasing the number of filters progressively as $N_1/4 = N_2/2 = N_3$. The rationale behind this choice is that the number of filters should compensate the downsampling along the spatial dimensions. In that way, the capacity at a given layer $W \times H \times D \times N$ decreases more slowly which allows the network to compress information more easily. In our experiments, we set $N_3 = 64$.

C. Changing the balancing weight in the focal loss (c4)

When considering point clouds as voxel grids, we observe that most of the space is empty (usually $> 95\%$). This large class imbalance between occupied voxels and unoccupied voxels is a barrier to effective training. Indeed, without any countermeasures, the network would converge towards empty outputs only. In order to resolve this class imbalance issue, we adopt the focal loss [6] as our distortion loss.

The focal loss is well suited for point clouds since it addresses the class imbalance issue with α-balancing. Moreover, the focal loss differentiates between easy and hard examples using the γ parameter. Specifically, the higher γ is, the more hard examples are emphasized. With $\gamma = 0$, the focal loss becomes equivalent to the weighted binary cross-entropy.

For conciseness, we adopt the following notation. If $x = 1$, then $x_t = x$, $\alpha_t = \alpha$ and $\tilde{x}_t = \tilde{x}$; otherwise $x_t = 1 - x$, $\alpha_t = 1 - \alpha$ and $\tilde{x}_t = 1 - \tilde{x}$. We then define the focal loss as

$$\text{FL}(x, \tilde{x}) = \alpha_t x_t (1 - \tilde{x}_t)^\gamma \log(\tilde{x}_t). \tag{4}$$

We study the impact of the focal loss α parameter on RD performance. The α parameter governs the attention given to occupied voxels and empty voxels. A high α value makes marking occupied voxels as empty more costly than marking empty voxels as occupied and results in denser reconstructions. Originally, we picked the same α value (0.90) as in [4]. This was motivated by the fact that point clouds are often comprised of more than 95% of empty space.

However, we found experimentally that lower α values can actually provide better coding gains. We hypothesize that this is due to the fact that the default $\gamma = 2$ in the focal loss emphasizes hard examples (occupied voxels) more than easy examples (empty voxels). Thus, $\gamma = 2$ already alleviates the class imbalance issue which explains this phenomenon.

D. Optimal thresholding for decoding (c5)

For each block, after decoding y (and z for the hyperprior model) into $\tilde{x}$, we need to convert $\tilde{x}$ into binary values in order to obtain the decompressed point cloud. The baseline method (c1) employs a fixed threshold $t = 0.5$. In contrast, we perform this conversion by finding optimal thresholds for each block of voxels. This threshold is transmitted as side information in the bitstream with a small overhead in terms of bitrate.

We formulate optimal thresholding as the problem of finding an optimal threshold $t^\star$ such that

$$t^\star = \arg\min_t d(x, H(\tilde{x} - t)) \tag{5}$$

where d is a distortion metric and $H(x)$ is the heaviside step function (equal to 1 when $x \geq 0$ and 0 otherwise).

E. Sequential training (c6)

We train compression models for each RD tradeoff using a corresponding λ value. This allows for transforms and entropy models to be specialized for this particular tradeoff resulting in better RD performance. Unfortunately, using this independent training scheme, we need to train one model for each tradeoff.

To alleviate this issue, we propose a novel sequential training scheme that speeds up training significantly and improves RD performance. The core idea of this scheme is to use previously trained neural network weights as a starting point for new neural networks. Essentially, given a set of λ tradeoffs, we first train λ_1. Then, for each subsequent model, we train λ_i using the trained weights of λ_{i-1}.

RD performance for each experimental condition. We specify BD-PSNR values (dB) compared to G-PCC trisoup and octree in each cell (trisoup BD-PSNR / octree BD-PSNR). The greatest values for trisoup and octree are indicated in **bold** and the second greatest in *italic*. **c6** consistently outperforms all other conditions.

Point cloud	Metric	Experimental conditions					
		c6	**c5**	**c4**	**c3**	**c2**	**c1**
loot	D1	**5.91 / 7.00**	*5.84 / 6.89*	4.05 / 5.06	2.03 / 3.67	−0.27 / 2.26	−0.72 / 1.88
	D2	*6.85 / 6.12*	**6.90 / 6.11**	4.10 / 3.33	1.44 / 1.23	−1.81 / −0.81	−2.60 / −1.40
redandblack	D1	**5.02 / 6.50**	*4.81 / 6.30*	3.28 / 4.71	1.43 / 3.45	−0.19 / 2.58	−0.59 / 2.01
	D2	**5.93 / 5.65**	*5.74 / 5.46*	2.91 / 2.62	0.55 / 0.79	−1.73 / −0.46	−2.42 / −1.18
longdress	D1	**5.54 / 6.94**	*5.41 / 6.79*	3.75 / 5.10	1.81 / 3.82	−0.26 / 2.64	−0.79 / 2.10
	D2	**6.59 / 6.01**	*6.52 / 5.91*	3.90 / 3.30	1.41 / 1.36	−1.37 / −0.34	−2.20 / −1.09
soldier	D1	**5.55 / 6.91**	*5.49 / 6.88*	3.76 / 5.11	1.88 / 3.88	−0.28 / 2.60	−0.77 / 2.13
	D2	**6.54 / 6.02**	*6.52 / 6.02*	3.86 / *3.36*	1.39 / 1.45	−1.54 / −0.40	−2.31 / −1.06
Average	D1	**5.50 / 6.84**	*5.39 / 6.71*	3.71 / 5.00	1.79 / 3.71	−0.25 / 2.52	−0.72 / 2.03
	D2	**6.48 / 5.95**	*6.42 / 5.87*	3.69 / 3.15	1.20 / 1.21	−1.61 / −0.50	−2.38 / −1.18

Impact of the focal loss α parameter on RD performance. We specify BD-PSNR values (dB) compared to G-PCC trisoup for different α values. The greatest values are indicated in **bold** and the second greatest in *italic*. $\alpha = 0.75$ outperforms all other α values.

Point cloud	Metric	α			
		0.90	0.75	0.50	0.25
loot	D1	2.03	*4.05*	**4.41**	1.24
	D2	1.44	4.10	**6.47**	*4.19*
redandblack	D1	1.43	**3.28**	*2.24*	−3.70
	D2	0.55	*2.91*	**5.19**	1.63
longdress	D1	1.81	**3.75**	*3.66*	−0.11
	D2	1.41	*3.90*	**6.28**	3.88
soldier	D1	1.88	*3.76*	**4.48**	1.68
	D2	1.39	3.86	**6.32**	*4.45*
Average	D1	1.79	**3.71**	*3.70*	−0.22
	D2	1.20	*3.69*	**6.07**	3.54

and reduces memory footprint when training. We perform training with λ values ranging from 5×10^{-6} to 3×10^{-4}.

We evaluate the different conditions using G-PCC trisoup and octree as baselines. Specifically, we use G-PCC v10.0 (released in May 2020) with the included configurations, "mpeg-pcc-dmetric" v0.12.3 for D1 and D2 metrics, Python 3.6.9 and TensorFlow 1.15.0 with the Adam optimizer [20].

B. Experimental results

In Fig. 3 and Table II, we observe that each condition is a net improvement over previous ones. **c6** outperforms G-PCC trisoup with an average BD-PSNR of 5.50 dB on D1 and 6.48 dB on D2 and outperforms G-PCC octree with an average BD-PSNR of 6.84 dB on D1 and 5.95 dB on D2. Note that the lowest bitrate point for **c6** is not included in BD-PSNR computations in order to keep integration intervals consistent and keep BD-PSNRs comparable across different conditions.

We also observe that **c5** (optimal thresholding) is especially beneficial for the point-to-plane metric (D2) with an improvement of 1.68 dB for D1 and 2.73 dB for D2 compared to **c4**. Indeed, optimal thresholding provides optimal sets of thresholds for D1 and D2 yielding two separate reconstructions.

In Fig. 4, we provide qualitative results on "soldier_vox10_0690". We observe that shapes and local point densities are reproduced more accurately compared to G-PCC trisoup. Overall, our method results in lower distortions at similar bitrates.

C. Ablation study

In this subsection, we present BD-PSNR values when compared to G-PCC trisoup. The hyperprior model (**c2**) results in an improvement of 0.47 dB for D1 and 0.77 dB for D2 compared to **c1**. Adding deep transforms (**c3**) further improves D1 by 2.04 dB and D2 by 2.81 dB compared to **c2**.

In Table III, we observe that setting $\alpha = 0.75$ for D1 and $\alpha = 0.50$ for D2 increases RD performance significantly for all point clouds. The average BD-PSNR for $\alpha = 0.75$ is 3.71 dB for D1 and 3.69 dB for D2. Also, the average BD-PSNR for $\alpha = 0.50$ is 3.70 dB for D1 and 6.07 dB for D2. Indeed, higher α values lead to denser reconstructions which are favored by D1 and lower α values to sparser ones which
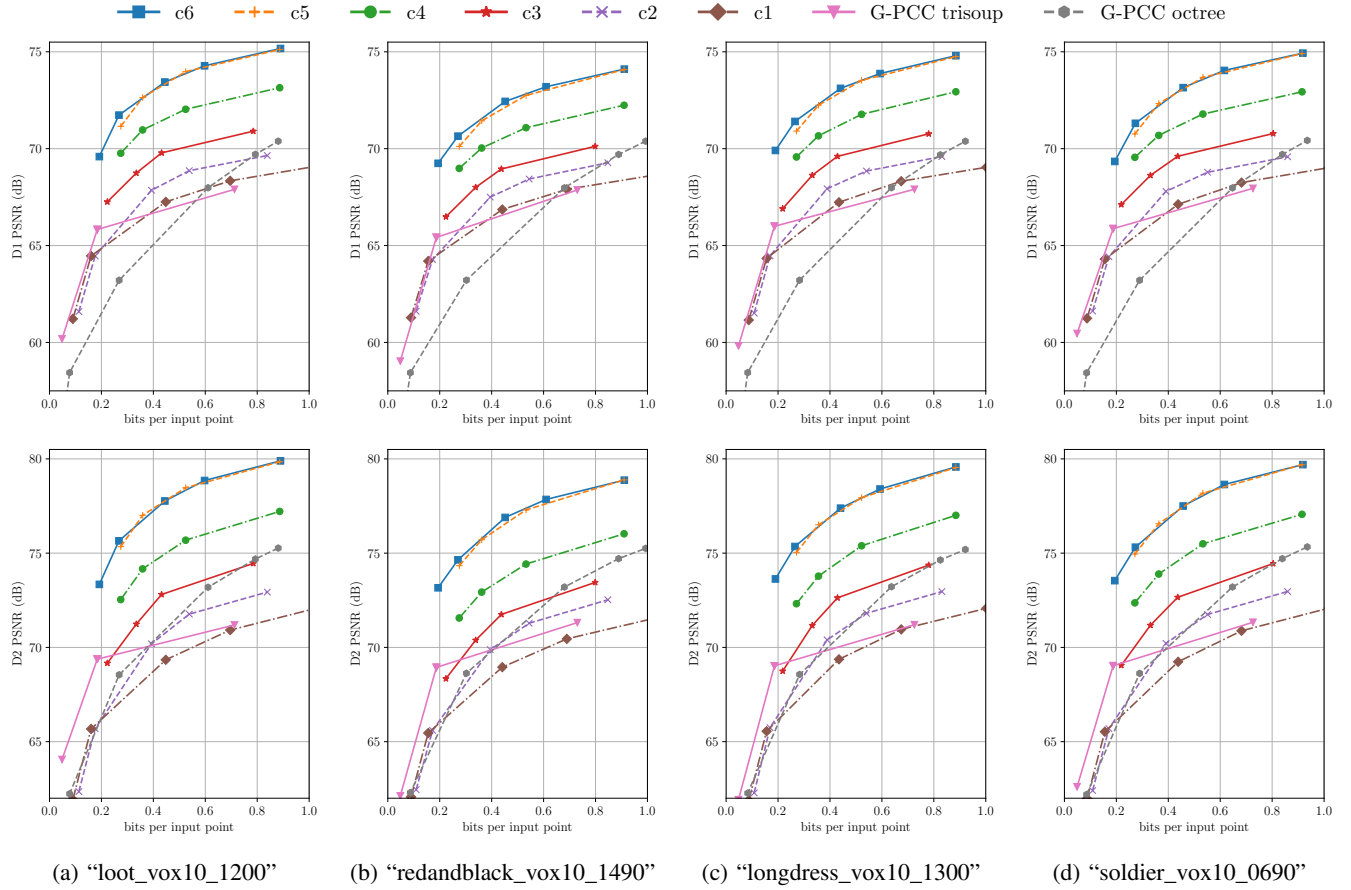
In this training scheme, we proceed to train the different tradeoffs in descending order. That is, we first train a low distortion, high bitrate model. Then, for each subsequent model, we progressively lower the bitrate while trying to minimize the increase in distortion.

IV. EXPERIMENTS

In this paper, we evaluate the six different improvement strategies described in Section III and summarized in Table I. The BD-PSNR gains are reported in Table II.

A. Experimental setup

We perform our experiments on four point clouds specified in the MPEG CTCs [2], [19]. Namely, "longdress_vox10_1300", "loot_vox10_1200", "redandblack_vox10_1490", "soldier_vox10_0690" which we refer to as "longdress", "loot", "redandblack" and "soldier".

We train our models on a subset of the ModelNet40 dataset. First, we sample the dataset into voxelized point clouds with resolution 512 and select the 200 largest point clouds. Then, we divide these point clouds into blocks with resolution 64 and select the 4000 largest blocks. This produces a small dataset containing rich point clouds, accelerates dataset loading time

(a) "loot_vox10_1200" (b) "redandblack_vox10_1490" (c) "longdress_vox10_1300" (d) "soldier_vox10_0690"

Fig. 3: RD curves for each condition in Table II. **c6** consistently outperforms G-PCC trisoup and G-PCC octree.

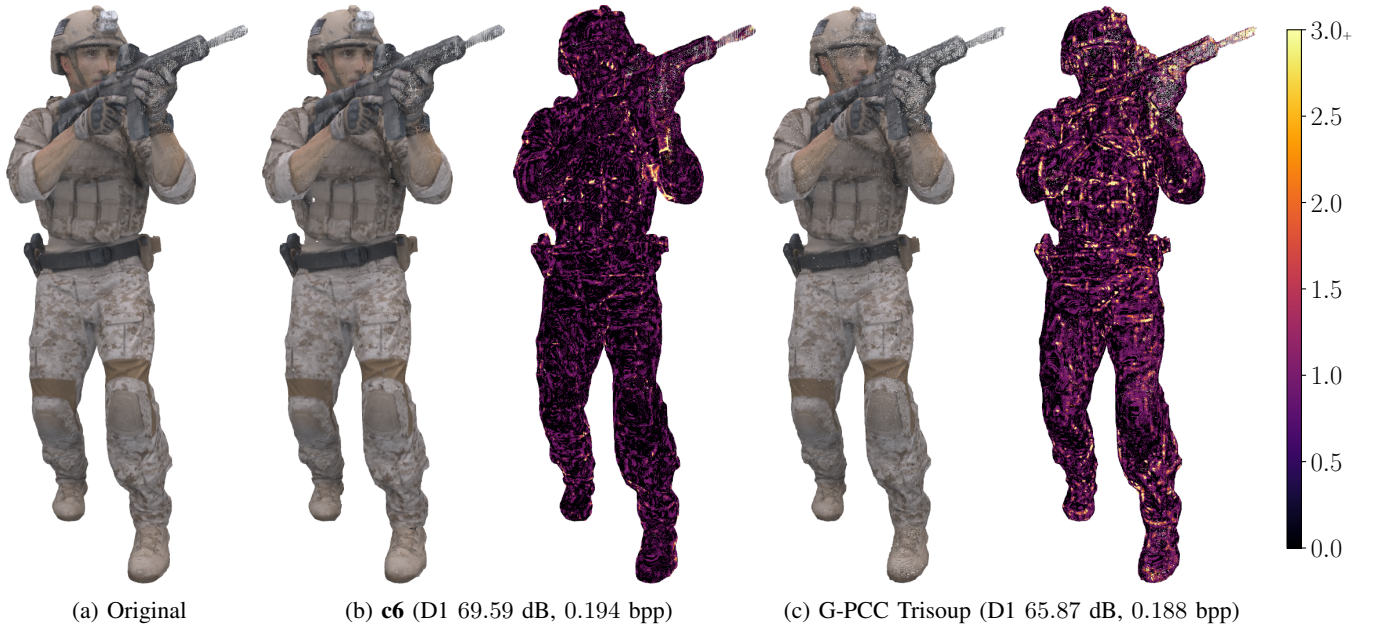(a) Original (b) **c6** (D1 69.59 dB, 0.194 bpp) (c) G-PCC Trisoup (D1 65.87 dB, 0.188 bpp)

Fig. 4: Qualitative evaluation on "soldier_vox10_0690". For **c6** and G-PCC Trisoup, we show the decompressed point cloud and its D1 squared errors. The errors are displayed according to the color scale on the right and are truncated to the 99th percentile (3.0). In parentheses, we specify the D1 PSNR along with the number of bits per input point (bpp).

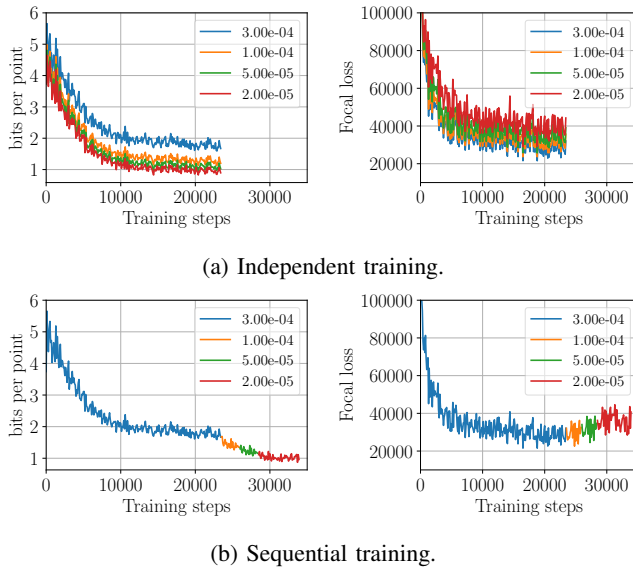(a) Independent training.

(b) Sequential training.

Fig. 5: Bits per point and focal loss when training independently and sequentially. Sequential training is more efficient as it reuses previously trained models to train subsequent ones.

are favored by D2. We select $\alpha = 0.75$ (**c4**) as we have found experimentally that it performs better when associated with optimal thresholding. Compared to $\alpha = 0.90$ (**c3**), $\alpha = 0.75$ brings an improvement of 1.92 dB for D1 and 2.49 dB for D2 .

Then, we use optimal thresholding (**c5**) with the point-to-point (D1) and point-to-plane (D2) objective metrics. As a result, we obtain two point clouds respectively optimized with D1 and D2. Also, we encode thresholds on 8 bits with 256 uniformly distributed threshold values between 0 and 1. Optimal thresholding (**c5**) results in an improvement of 1.68 dB for D1 and 2.73 dB for D2 compared to **c4**.

Training DPCC models is time consuming as shown in Fig. 5. Indeed, the **c5** condition requires 4 hours of training resulting in a total of 16 hours for four models on an Nvidia GeForce GTX 1080 Ti. With sequential training (**c6**), these models train in 30 to 60 minutes instead of 4 hours which is up to 8 times faster. In addition, this results in an improvement of 0.11 dB for D1 and 0.06 dB for D2 compared to **c5**.

V. CONCLUSION

We propose a set of key performance factors for DPCC and we present an extensive ablation study on the individual impact of these factors. More precisely, we provide insights on the individual impact of scale hyperprior models, deep transforms, the focal loss α value, optimal thresholding and sequential training. We analyze each of these factors in order to provide a better understanding about why they improve RD performance. The final model (**c6**) outperforms G-PCC trisoup with an average BD-PSNR of 5.50 dB on D1 and 6.48 dB on D2 and outperforms G-PCC octree with an average BD-PSNR of 6.84 dB on D1 and 5.95 dB on D2.

REFERENCES

[1] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuca, S. Lasserre, Z. Li, J. Llach, K. Mammou, R. Mekuria, O. Nakagami, E. Siahaan, A. Tabatabai, A. M. Tourapis, and V. Zakharchenko, "Emerging MPEG Standards for Point Cloud Compression," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, pp. 1–1, 2018.

[2] "Common test conditions for point cloud compression," in *ISO/IEC JTC1/SC29/WG11 MPEG output document N19084*, Feb. 2020.

[3] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, "Geometric distortion metrics for point cloud compression," in *2017 IEEE International Conference on Image Processing (ICIP)*. Beijing: IEEE, Sep. 2017, pp. 3460–3464.

[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)*, Sep. 2019, pp. 4320–4324, iSSN: 1522-4880.

[5] A. F. R. Guarda, N. M. 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)*, Oct. 2019, pp. 34–39, iSSN: 2164-974X.

[6] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 2999–3007, iSSN: 2380-7504.

[7] R. Schnabel and R. Klein, "Octree-based Point-cloud Compression," in *Proceedings of the 3rd Eurographics / IEEE VGTC Conference on Point-Based Graphics*, ser. SPBG'06, 2006, pp. 111–121. [Online]. Available: http://dx.doi.org/10.2312/SPBG/SPBG06/111-120

[8] D. C. Garcia and R. L. d. Queiroz, "Intra-Frame Context-Based Octree Coding for Point-Cloud Geometry," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, Oct. 2018, pp. 1807–1811.

[9] A. Dricot and J. Ascenso, "Adaptive Multi-level Triangle Soup for Geometry-based Point Cloud Coding," in *2019 IEEE 21st International Workshop on Multimedia Signal Processing (MMSP)*, Sep. 2019, pp. 1–6, iSSN: 2163-3517.

[10] ——, "Hybrid Octree-Plane Point Cloud Geometry Coding," in *2019 27th European Signal Processing Conference (EUSIPCO)*, Sep. 2019, pp. 1–5, iSSN: 2219-5491.

[11] P. de Oliveira Rente, C. Brites, J. Ascenso, and F. Pereira, "Graph-Based Static 3D Point Clouds Geometry Coding," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 284–299, Feb. 2019.

[12] M. Krivokuća, M. Koroteev, and P. A. Chou, "A Volumetric Approach to Point Cloud Compression," *arXiv:1810.00484 [eess]*, Sep. 2018, arXiv: 1810.00484. [Online]. Available: http://arxiv.org/abs/1810.00484

[13] J. Ballé, V. Laparra, and E. P. Simoncelli, "End-to-end Optimized Image Compression," in *2017 5th International Conference on Learning Representations (ICLR)*, 2017.

[14] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston, "Variational image compression with a scale hyperprior," in *2018 6th International Conference on Learning Representations (ICLR)*, Jan. 2018.

[15] J. Wang, H. Zhu, Z. Ma, T. Chen, H. Liu, and Q. Shen, "Learned Point Cloud Geometry Compression," *arXiv:1909.12037 [cs, eess]*, Sep. 2019, arXiv: 1909.12037. [Online]. Available: http://arxiv.org/abs/1909.12037

[16] L. Huang, S. Wang, K. Wong, J. Liu, and R. Urtasun, "OctSqueeze: Octree-Structured Entropy Model for LiDAR Compression," *arXiv:2005.07178 [cs, eess]*, May 2020, arXiv: 2005.07178. [Online]. Available: http://arxiv.org/abs/2005.07178

[17] M. Quach, G. Valenzise, and F. Dufaux, "Folding-based compression of point cloud attributes," in *2020 IEEE International Conference on Image Processing (ICIP)*, to be published. [Online]. Available: http://arxiv.org/abs/2002.04439

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778, iSSN: 1063-6919.

[19] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, "8i Voxelized Full Bodies - A Voxelized Point Cloud Dataset," in *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006*, Geneva, Jan. 2017.

[20] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *2015 3rd International Conference on Learning Representations*, Dec. 2014, arXiv: 1412.6980. [Online]. Available: http://arxiv.org/abs/1412.6980