# DEEP LEARNING-BASED POINT CLOUD GEOMETRY CODING: RD CONTROL THROUGH IMPLICIT AND EXPLICIT QUANTIZATION

*André F. R. Guarda*[1], *Nuno M. M. Rodrigues*[2], *Fernando Pereira*[3]

[1,3]Instituto Superior Técnico, Instituto de Telecomunicações, Lisbon, Portugal
[2]ESTG, Instituto Politécnico de Leiria, Instituto de Telecomunicações, Leiria, Portugal
[1]andre.guarda@lx.it.pt, [2]nuno.rodrigues@co.it.pt, [3]fp@lx.it.pt

## ABSTRACT

Deep learning is becoming more and more relevant for multiple multimedia processing tasks, and lately it has raised much interest in the coding arena notably for images and point clouds. While offering near state-of-the-art compression performance, current deep learning-based point cloud coding solutions have a shortcoming since they require training and storing multiple models in order to obtain different rate-distortion trade-offs. This paper proposes a solution that effectively reduces the number of deep learning models that need to be trained and stored by applying explicit quantization to the latent representation, which can be controlled at coding time, to generate varying rate-distortion trade-offs. The proposed implicit-explicit quantization combination achieves a compression performance that is equivalent or better than the alternative, while significantly reducing the model storage memory requirements.

***Index Terms***— Point cloud coding, deep learning, explicit quantization, multiple models

## 1. INTRODUCTION

Recent advances in virtual reality related technologies have been boosting the usage of 3D visual representation formats, notably point clouds (PCs) due to their great advantages in terms of user immersion and interaction. A PC is an unordered set of points in 3D space, usually represented by their coordinates, corresponding to the geometry; moreover, each PC point may have other associated attributes, such as color, which will not be addressed in this paper. As more and more PC content is made available, the development of appropriate coding solutions becomes essential since non-compressed PCs require a huge amount of data, thus preventing efficient transmission and storage. Recognizing the importance of PC data for major emerging 3D-based applications, MPEG has just concluded the specification of two PC coding standards for static and dynamic PCs [1], [2], while JPEG has recently launched a Call for Evidence on Point Cloud Coding, especially addressing scalability and random access [3], [4].

After considerable success in several multimedia processing domains, deep learning (DL) has made its way into the media coding arena, with recent proposals achieving close to state-of-the-art performance for image [5] and point cloud [6] coding. The few available DL-based PC coding solutions have some relation with the traditional transform coding paradigm since an autoencoder (AE) is used to learn a non-linear transform. In DL-based coding solutions, the rate-distortion (RD) control is performed during the training process by using a loss function, which embeds a trade-off between the reconstruction error (distortion) and the entropy of the latent representation (rate). Different RD trade-offs are obtained by tuning the Lagrangian multiplier value in the training loss function, unlike the use of explicit quantization, as in conventional transform-based encoders. In fact, the DL-based coding solutions adapt the scale of the latent values (coefficients) resulting from the learned transform, depending on the Lagrangian multiplier value, thus operating as an *implicit quantization* to reach different RD trade-offs. This approach presents a major disadvantage as it requires training and storing a different DL model for each relevant quality/rate level, i.e. RD point, which may not even be known in advance.

In this context, this paper proposes an alternative lower-complexity quantization approach for DL-based PC geometry coding, which targets reducing the number of trained DL models while offering the same or even a larger number of quality/bitrate trade-offs (RD points). The novel PC geometry coding solution combines the implicit and *explicit quantization* of the latent representation to obtain multiple RD points from each trained DL model, thus reducing the associated training and memory requirements. In practice, RD control is performed in two ways: first through the Lagrangian parameter associated to each trained DL models; and second, through the quantization step (QS) parameter associated to the explicit quantization. Experimental results show that, by carefully designing the implicit-explicit quantization combinations, this approach can be used to replace some of the trained DL models without penalizing the overall RD performance.

The remainder of this paper is organized as follows: Section 2 briefly reviews the state-of-the-art on PC geometry coding, notably in terms of conventional and DL-based methods. Section 3 describes the proposed DL-based solution and introduces the implicit and explicit quantization approaches. Section 4 presents and discusses the experimental results and, finally, Section 5 concludes the paper.

## 2. PC GEOMETRY CODING: BACKGROUND REVIEW

The specific characteristics of PC data have brought new coding needs and challenges. The key differentiating characteristic is the unorganized nature of PCs, which means that they do not have a regularly filled structure as pixels in a 2D image. As such, most traditional PC geometry coding solutions proposed in the literature [7] can be classified in two categories: i) 2D projection, where the PC data is mapped into 2D maps, which may be coded with well-established image/video codecs; and ii) 3D geometry, where the PC data 3D structure is directly exploited as with octrees or the more recent DL-based coding solutions, in the following separately addressed due to the relevance of the second type for this paper.


978-1-7281-1485-9/20/$31.00 ©2020 IEEE

### 2.1. 2D projection-based coding solutions

This type of PC geometry coding solutions aims at taking advantage of the decades of progress in image and video coding to obtain a good PC compression efficiency by essentially turning the PC coding problem into a mapping/projection problem followed by 2D coding. In [8], the PC is clustered into several patches, with each patch being resampled and projected into a regular 2D grid, forming a height map. Then, each height map is coded with a shape-adaptive wavelet image coder. In [9], the PC is first projected into a panorama image using equirectangular projection, and the panorama image is then coded with a traditional image codec, e.g. JPEG [10].

The state-of-the-art 2D projection-based PC geometry coding solution is the MPEG Video-based Point Cloud Compression (V-PCC) standard [11], which was recently finalized. V-PCC starts by segmenting the PC into patches according to the normal vector associated with each point. Patches are then projected and packed onto 2D images, generating depth images (for geometry), texture images (for color) and binary occupancy maps, which signal the 2D pixels corresponding to actual 3D points. The 2D images can then be coded by any image/video codec, notably the highly efficient HEVC video coding standard [12].

### 2.2. 3D geometry-based coding solutions

The most popular 3D geometry coding solutions are based on octrees. To build an octree data structure, starting from the root node, each node occupied with at least one point is recursively subdivided into eight child nodes, until a desired depth is reached [13]. In [14], an extension for dynamic PCs is presented, which first builds an octree for each PC frame, and then encodes only the octree differences (via exclusive disjunction) for each successive frame.

The MPEG Geometry-based Point Cloud Compression (G-PCC) standard [1] is the current state-of-the-art for 3D geometry-based coding solutions. G-PCC firstly computes an octree up to a target depth, and then applies a technique known as *trisoup* to estimate the surface inside each occupied octree leaf node using a number of triangles.

### 2.3. Deep learning-based coding solutions

Following their success in image coding, DL-based solutions have been recently adopted and adapted to PC coding [6], [15], [16]. These solutions typically replicate the principles behind the conventional transform-based coding paradigm, as used in the popular JPEG image coding standard [10]. The DL-based coding solutions commonly use convolutional neural networks (CNNs) to exploit the correlations between neighboring pixels/points. But whereas an image is a structured representation with rectangular blocks that convolutional layers can easily process, a PC is unorganized by definition. As such, the natural ($x$, $y$, $z$) PC geometry representation needs to be converted into a 3D binary (empty/filled) block with a regular size. In [15] and [16], an autoencoder (AE) is used to transform the input PC data into a feature space, the so-called *latent representation*. The latent representation is then quantized, with a given quantization step (QS), and entropy coded, resulting in different RD trade-offs depending on the QS value. In [6], the RD trade-off is achieved during training, by using an RD loss function, with entropy coding also considered during training to estimate the rate. While this approach performs substantially better than the first, with a compression performance comparable to G-PCC, it requires training and storing a model for every desired RD point.

## 3. DL-BASED PC GEOMETRY CODING: COMBINING IMPLICIT AND EXPLICIT QUANTIZATION

This section proposes a DL-based PC geometry coding solution attaining overall lower-complexity by combining implicit with explicit quantization. In this case, multiple RD points are obtained at the cost of a small number of trained DL models, thus reducing both the training complexity and DL models storage requirements.

### 3.1. Codec architecture and walkthrough

This section describes the adopted PC geometry coding architecture, which is shown in Fig. 1. As usual for DL-based solutions, the training and the testing/inference (in this case coding) processes have to be addressed.

#### *3.1.1. Training*

To have a functional codec with efficient compression performance following a specific DL-based architecture, the DL model needs to be trained using an appropriate loss function, which should be able to balance the distortion and the rate. This is commonly accomplished using an RD loss function, with a Lagrangian multiplier λ, defined as:

$$Loss = dist + \lambda \times rate, \quad (1)$$

where *dist* is the distortion between the input and reconstructed PCs, measured here as the mean Focal Loss [17] across the entire PC volume, and *rate* is the estimated rate, measured here as the entropy of the quantized latent representation. By training DL models with various λ values, different target qualities/bitrates, i.e. RD points, are obtained, without involving explicit quantization; the DL models can then be used for coding any new PC.

#### *3.1.2. Testing/Coding*

The coding process involves three main modules as follows:

- **Transform** – As for previous DL-based PC coding solutions [6], [15], [16], the transform is learned with a convolutional AE, which is typically not perfectly reversible. The adopted AE encoder consists of three convolutional layers, each applying 32 filters with a 5×5×5 support, using a sigmoid activation function. To reduce the representation dimensionality, a stride of two allows down-sampling the feature maps, thus forcing the AE to learn only important features and allowing compression. As common in AE architectures, the decoder has a symmetric structure.

- **Quantization** – After obtaining the latent representation, some information may be discarded by applying quantization. As in traditional (image) coding, a quantization parameter is selected depending on the desired target quality or rate, usually known as the Quantization Step (QS). Since quantization is not differentiable, this operation is replaced by the addition of uniform noise during training, [18]; during coding, it performs as a normal quantization.

- **Entropy coding** – The final bitstream is obtained by entropy coding the quantized latent representation. In this paper, the entropy coding with a scale hyperprior method from [19] is used. The main idea is to transmit some side information to better adapt the entropy model to the current latent data, so that entropy coding is more efficient. To this end, this method uses a variational autoencoder (VAE), which estimates the entropy model parameters for the latent representation, which are then used by a range coder to entropy code the quantized latent values. Since the VAE is trained together with the AE transform as an end-to-end model, and the entropy coding side information is included in the estimated *rate* used in the training loss function (1), the DL model balances the entropy side information with the corresponding overall improvement in terms of compression efficiency.

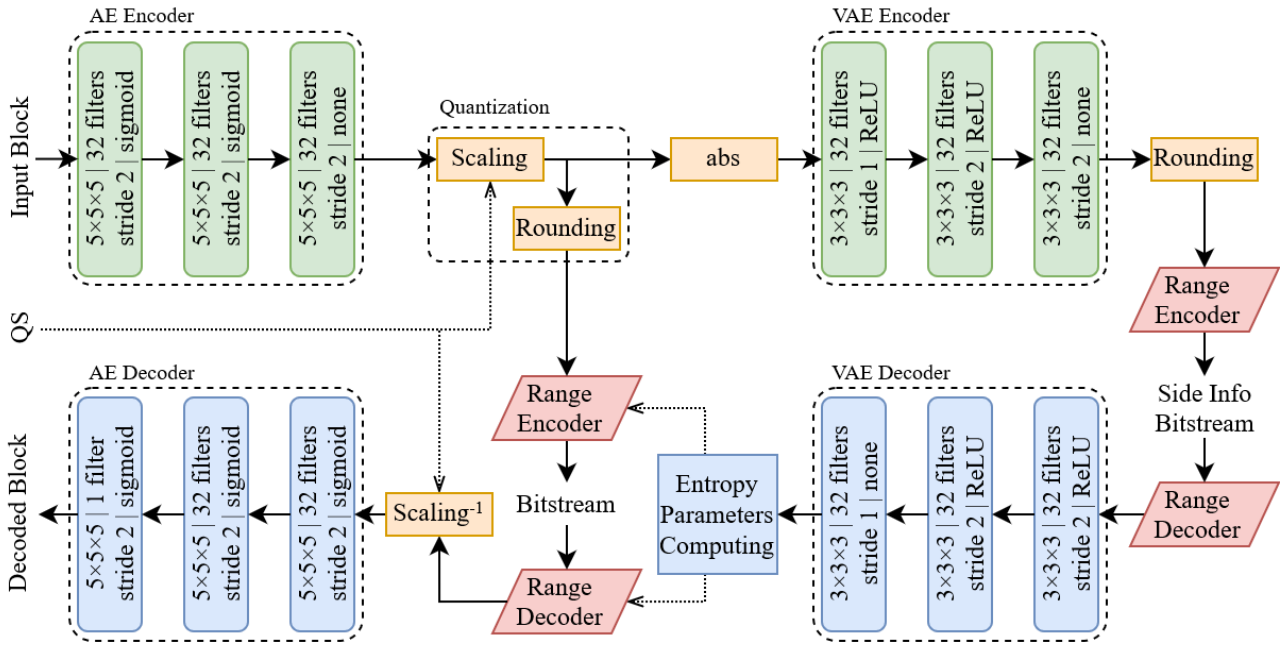


**Fig. 1.** DL-based PC coding architecture using explicit quantization with Quantization Step QS. If QS=1, quantization becomes implicit.

### 3.2. Implicit quantization: RD control via loss function

Unlike for conventional transform-based coding, in DL-based coding, both for PCs [6] as well as images [5], RD control is accomplished during the DL training process by tuning the Lagrangian multiplier λ in the loss function (1), and thus not at coding time. By changing the value of λ, different DL models are learned, with varying dynamic ranges for the latent representation, which achieve different quality and rate combinations, thus offering multiple RD points. This approach is referred in this paper as *implicit quantization* since no QS-controlled process is performed at the encoding stage. It is important to notice that in this coding approach, the floating-point latents are only rounded to the closest integer since this is required by the entropy coder [18].

### 3.3. Explicit quantization: RD control via quantization

As mentioned in the previous section, with implicit quantization, a new DL model must be trained for every single RD point, thus for each target quality/rate trade-off; however, this results in high training complexity and memory requirements. To avoid this drawback, this paper proposes combining the usage of implicit and explicit quantization to eliminate the need for some DL trained models, thus reducing the complexity and memory requirements, eventually without compression efficiency penalties.

When performing explicit quantization, models are still trained with QS=1 (thus performing, in practice, implicit quantization). However, at coding time, a larger QS may be used to create more RD points from the same DL model, thus allowing to perform RD control through low-complexity explicit quantization. This explicit quantization is applied as follows: at the encoder, after obtaining the AE latent representation, the latent values are (down-)scaled by QS, as represented in Fig. 1 with the *Scaling* block. The scaled latent representation is then fed to the VAE for side information entropy coding estimation and also entropy coded after rounded to integers. At the decoder side, after entropy decoding, the latent values are inverse quantized, i.e. (up-)scaled, by the same QS, as represented in Fig. 1 by the *Scaling*$^{-1}$ block. Naturally, the used QS value needs to be transmitted to the decoder.

By using this simple Scaling at the encoder/decoder, no other changes are required regarding the DL-codec with implicit quantization and, thus, the same entropy coding solution [19] can be applied. Since the entropy model is now learned from the scaled latent representation, the entropy coding process adapts to the selected QS for which scaling was performed.

In this context, each single trained DL model can be used for obtaining several target RD points, one with implicit quantization (equivalent to QS=1) and others with explicit quantization (QS>1). This allows saving on the number of DL models that are required to be trained and stored and offers flexibility on the number of available RD points. Since the DL models are originally trained for QS=1, larger QS values may introduce artifacts in the reconstruction, which intensify as QS increases. This makes important to study the implicit-explicit balance, notably how many explicit quantization RD points is reasonable to derive from each single DL model before the quality degradation is too obvious. The next section will assess the performance of implicit-explicit quantization DL-based coding method in comparison with implicit-only quantization DL-based coding.

## 4. PERFORMANCE ASSESSMENT

In this section, the performance of DL-based PC geometry coding with implicit-explicit quantization versus implicit-only quantization is compared using as benchmark the state-of-the-art MPEG G-PCC standard. The experimental test conditions are described in the following.

### 4.1. Experimental test conditions

For both implicit-only and implicit-explicit quantization, the adopted DL-based PC coding solution was trained and evaluated using the MPEG PCC dataset [20]. Prior to encoding, all PCs were down-sampled to their respective coding precision defined in the PCC test conditions [20], which is used as reference when measuring the reconstruction quality. The PCs *Statue Klimt, Queen, Longdress*

and *House without Roof* were selected for evaluation, while the remaining PCs in the MPEG test set were used for training only. PCs are first transformed into 3D binary blocks, namely of size 64×64×64 voxels, which were then coded independently.

Several DL models have been trained using implicit-only quantization for different RD trade-offs, by changing the λ parameter in the loss function (1). Using each of these trained DL models, the test PCs were encoded and decoded with explicit quantization, using integer QS values between 1 and 20.

The coding benchmark for the DL-based coding solutions is, naturally, the MPEG G-PCC standard, more precisely the reference software version 7.0, which represents the state-of-the-art for static PC compression. To evaluate the RD performance, the PSNR of the point-to-point MSE distance, also known as D1 [20], is used as the objective quality metric while the rate is measured in bits-per-point (bpp) of the original PC.

### 4.2. Implicit-explicit quantization with a single DL model

The first experiment studies the impact of explicit quantization by comparing the performance of the implicit-only quantization DL coding solution, which requires training as many DL models as the number of RD points offered, with implicit-explicit quantization DL coding solutions using a single DL trained model. For the implicit-only quantization results, multiple DL models were trained with different λ values, ranging from 50 to 20000, to generate RD points at a variety of quality and rate levels. The implicit-explicit quantization results were obtained using a single one of the previously trained DL models, with the created latents quantized using increasing QS values, to obtain multiple RD points at a variety of quality and rate levels.

Fig. 2 shows the RD performance comparison between the implicit-only and implicit-explicit quantization coding solutions. Three examples are shown for the implicit-explicit quantization case, notably using three different DL models trained with different λ values (50, 500 and 1500). The points in each of these curves marked with an "x" correspond to the implicit quantization RD points, while the remaining ones (from right to left) correspond to progressively higher QS values for the explicit quantization. The results show that, overall, the explicit quantization RD points are initially close to the implicit quantization RD points; however, as QS increases, the differences between the two quantization solutions increase. Naturally, as expected, the implicit-explicit quantization combination fails to reach the wide variety of bitrates that is offered by multiple DL trained models associated to implicit-only quantization. When using a DL model trained for higher bitrates (λ=50), the implicit-explicit quantization cannot reach the lower bitrates, with the quality decreasing more rapidly as QS increases. On the other hand, if the DL model is trained for lower bitrates (λ=1500), the implicit-explicit quantization can closely match the implicit-only quantization results, but it cannot reach the higher qualities. This is expectable as the single used DL model was trained for a specific RD point and did not learn how to behave for a large range of qualities and rates as opposed to the multiple, specific DL models created with implicit-only quantization.

As can be seen in Fig. 2, for the lower QS values, there tends to be a marginal RD performance gain by the implicit-explicit quantization over the implicit-only quantization; while this effect could be unexpected, it be explained by the reduction of the number of non-zero latent values, e.g. happening when increasing QS from 1 to 2, notably up to 50% of the non-zero values are now quantized to 0, thus leading to a substantial bitrate reduction. Whereas for smaller QSs the quantization error does not significantly impact the quality associated to the most important latent values, when QS is larger, it severely degrades the latent representation, thus causing a rapid decrease in quality.

These RD performance results provide a good indication that using implicit-explicit quantization is a valuable alternative to training and storing (many) DL implicit-only quantization models, notably for all target RD points. However, the results also show that multiple DL models are still required to cover a wide range of qualities/rates as one single DL model is not enough.

### 4.3. Visual effects of explicit quantization

With explicit quantization, it is expected that some quality artifacts appear in the reconstructed PCs, notably as QS increases, since the DL models are trained to consider only the effect of rounding the latent values (QS=1). To better understand the quality impacts of performing explicit quantization, Fig. 3 shows a detail comparison for a reconstructed PC view in different coding situations. Fig. 3 (b) and (c) compare implicit and explicit quantization results for a RD point with similar quality level at *medium to high bitrates*. For these rates, even for a QS as high as 5, the explicit quantization reconstruction quality is nearly identical to the implicit-only quantization, with only minor artifacts being visible. However, when making the same comparison for an RD point at *lower rates*, as in Fig. 3 (d) and (e), the reconstruction quality for explicit quantization with the same QS of 5 is now noticeably deteriorated, showing artifacts such as holes and lumps in the surface, even though the objective quality metric has similar scores for both situations. This may be explained by the fact that the latent representation dynamic range at lower rates is significantly smaller than at higher rates, implying that latent values close to zero are more important overall for the lower rates and cannot be as easily eliminated as for higher rates. When using the same large QS at lower rates, most of the latent values are eliminated (i.e. quantized to zero), whereas the latent representation is still meaningful for the QS at higher rates. As such, when using explicit quantization to replace DL models with implicit-only quantization, it is preferable to use only small QSs at lower rates, whereas higher bitrates allow adopting larger QSs to obtain more RD points.

### 4.4. Implicit-explicit quantization with multiple models

Taking into account the previous experiments, a final comparison is made to evaluate the performance of using fewer trained DL models with implicit-explicit quantization. For implicit-only quantization, 10 trained DL models with a meaningful range of qualities were selected, notably λ = 100, 250, 500, 600, 750, 1000, 2000, 3000, 10000, and 20000. For implicit-explicit quantization, only three of these DL models were selected, notably λ = 100, 1000 and 10000. The remaining seven points were obtained by applying explicit quantization to the three trained DL models, using a larger QS; taking into account the results from the previous section, the following QS values have been used: QS=2, 3, 4 and 5 for λ = 100, QS=2 and 3 for λ = 1000, and QS=2 for λ = 10000.

Fig. 4 shows the RD performance comparing both quantization approaches, and Table I shows the Bjontegaard-Delta bitrate and PSNR gains for implicit-explicit quantization (with three trained DL models) over implicit-only quantization (with 10 trained DL models). Overall, it can be seen that the RD performance is similar, although there are some slight BD improvements when using implicit-explicit quantization, contrary to what could be expected. This is mostly due to the rate savings achieved when initially increasing the QS to 2 or even 3, as previously explained. When using three DL models, the slight RD gains associated to each DL

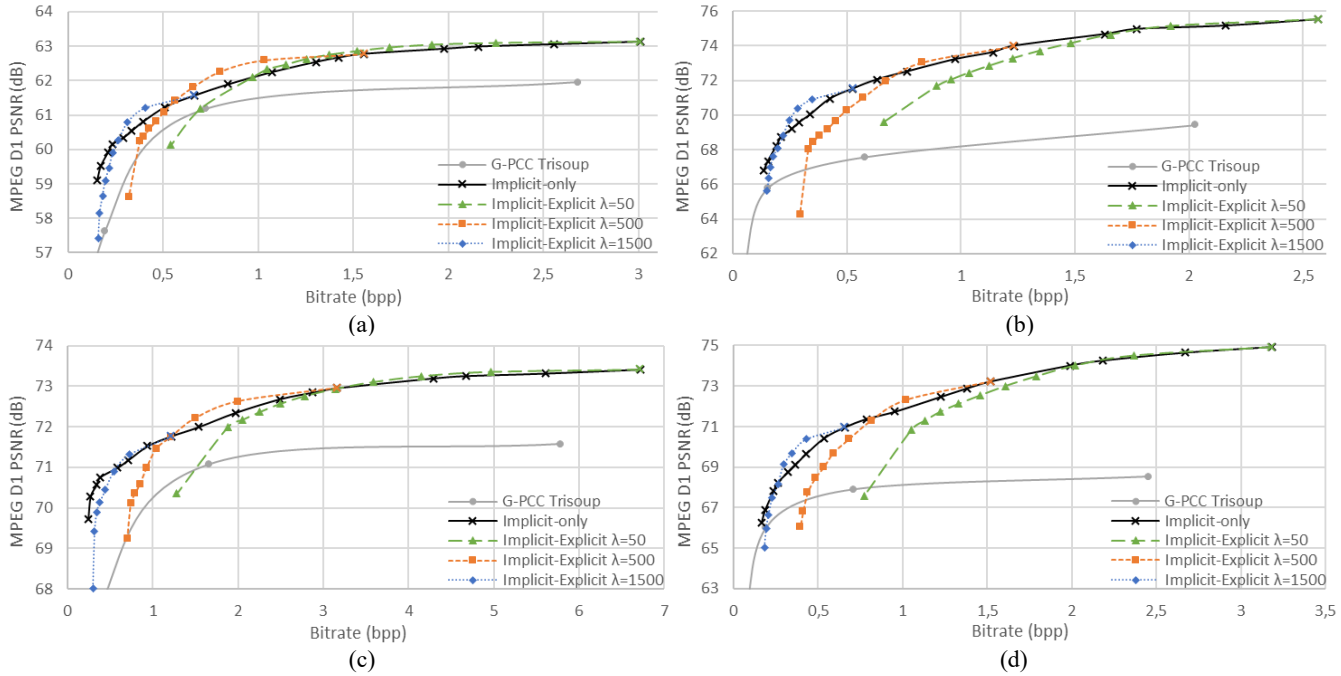


**Fig. 2.** RD performance for implicit quantization *versus* single model implicit-explicit quantization, for different RD trade-offs: (a) *Statue Klimt;* (b) *Queen;* (c) *House without Roof;* and (d) *Longdress*. Points with an "x" marker correspond to the trained DL models.

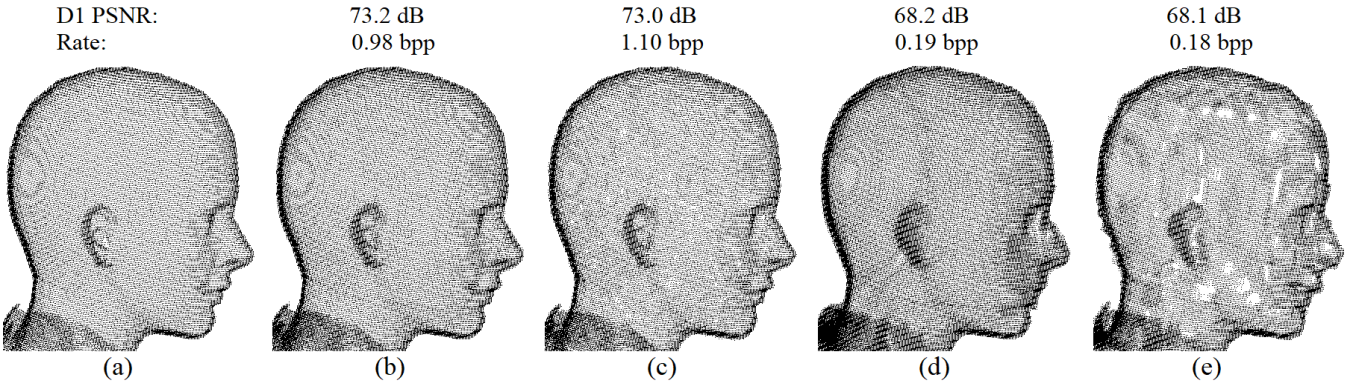


**Fig. 3.** Detail comparison of view reconstruction quality for *Queen*: (a) original PC; (b) implicit quantization for λ=750; (c) explicit quantization for λ=100 with QS 5; (d) implicit quantization for λ=10000; and (e) explicit quantization for λ=2000 with QS 5.

model add up, thus eventually surpassing the quality losses associated to the higher QS.

These results demonstrate that it is indeed beneficial to use implicit-explicit quantization to obtain intermediate bitrate/quality levels, instead of training and storing more DL models. As demonstrated above, similar RD performance can be achieved even when using less than one third of the number of trained DL models. It is worth noting that many other combinations of DL models and QS values may be used, thus offering a trade-off between complexity (measured by the number of trained and stored DL models) and RD performance.

## 5. CONCLUSIONS

As for almost all areas of multimedia data processing, deep learning tools are becoming relevant for PC coding. State-of-the-art DL-based PC coding solutions commonly adopt an implicit quantization method, where RD control is performed through the training loss function parameter. This requires multiple DL models to be trained and stored at both the transmitter and receiver sides to achieve different target RD points, thus increasing the complexity and memory requirements. This paper proposes the use of implicit quantization combined with explicit quantization to reduce the number of required DL models, while still allowing to obtain multiple target RD points. Experimental results demonstrate that the number of trained DL models can be significantly reduced, e.g. at least a factor of 3, while maintaining RD performance or even obtaining RD performance gains.

**Table 1.** RD performance gains for implicit-explicit quantization over implicit-only quantization.

| | **BD-Rate** | **BD-PSNR** |
|---|---|---|
| ***Statue Klimt*** | -11.29 % | 0.16 dB |
| ***Queen*** | -3.48 % | 0.11 dB |
| ***House without Roof*** | -3.89 % | 0.05 dB |
| ***Longdress*** | -3.03 % | 0.10 dB |
| **Average** | -5.42 % | 0.11 dB |

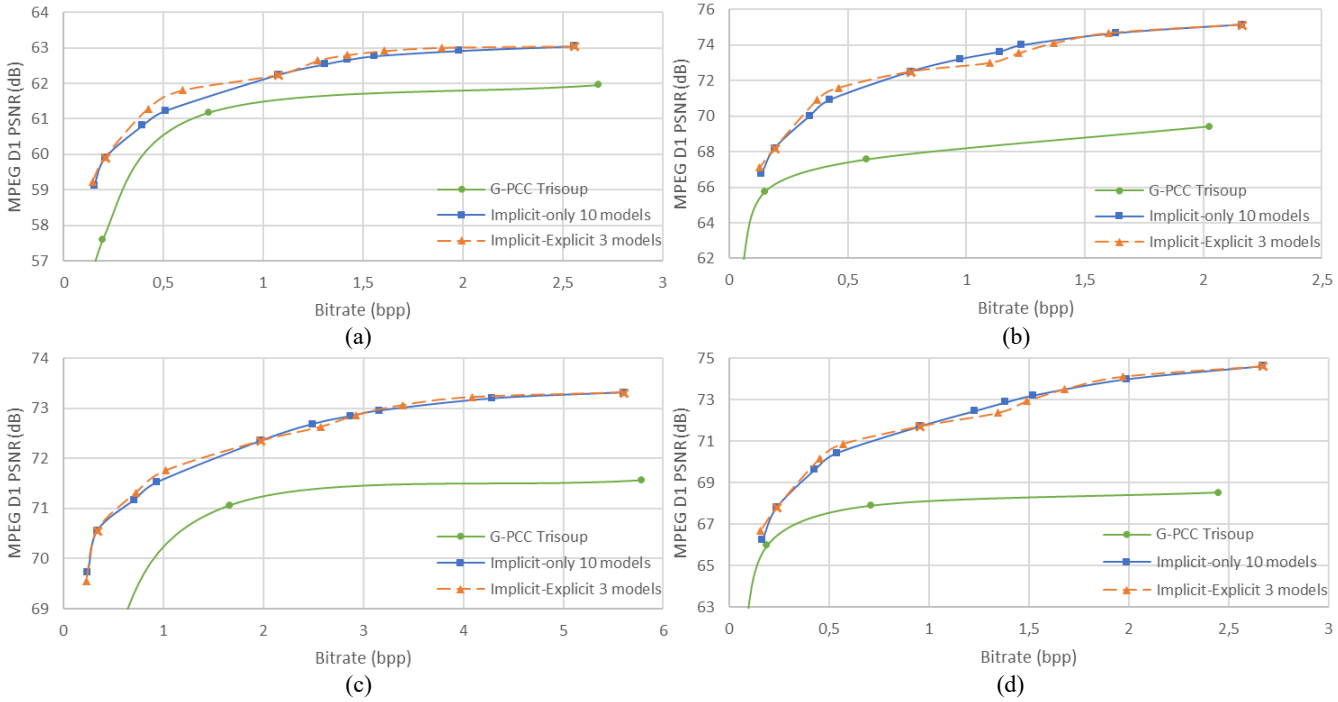


**Fig. 4.** RD performance for implicit-only quantization *versus* implicit-explicit quantization with multiple trained DL models: (a) *Statue Klimt;* (b) *Queen;* (c) *House without Roof;* and (d) *Longdress*. Points with an "x" marker correspond to the trained DL models.

## 6. ACKNOWLEDGMENTS

This work was funded by Fundação para a Ciência e Tecnologia (FCT), Portugal, Ph.D. Grant SFRH/BD/ 118218/2016, by FCT/MEC through national funds and when applicable co-funded by EU funds under the project UIDB/EEA/50008/2020.

## 7. REFERENCES

[1] S. Schwarz *et al*., "Emerging MPEG Standards for Point Cloud Compression," IEEE Trans. Emerg. Sel. Topics Circuits Syst., vol. 9, no. 1, pp. 133–148, Mar. 2019.

[2] L. Cui *et al.*, "Point-Cloud Compression: Moving Picture Experts Group's New Standard in 2020," IEEE Consum. Electron. Mag., vol. 8, no. 4, pp. 17-21, July 2019.

[3] T. Ebrahimi *et al.,* "JPEG Pleno: Toward an Efficient Representation of Visual Reality," IEEE MultiMedia, vol. 23, no. 4, pp. 14-20, Oct.-Dec. 2016.

[4] ISO/IEC JTC 1/SC29/WG1 N86013, "First Call for Evidence on JPEG Pleno Point Cloud Coding," Sydney, Australia, Jan. 2020.

[5] D. Minnen, J. Ballé and G. Toderici, "Joint Autoregressive and Hierarchical Priors for Learned Image Compression," Advances in Neural Inf. Process. Syst., Montreal, Canada, Dec. 2018.

[6] M. Quach, G. Valenzise and F. Dufaux, "Learning Convolutional Transforms for Lossy Point Cloud Geometry Compression," IEEE Int. Conf. on Image Process., Taipei, Taiwan, 2019.

[7] C. Cao, M. Preda, and T. Zaharia, "3D Point Cloud Compression: A Survey," Int. Conf. 3D Web Technol., Los Angeles, CA, USA, July 2019.

[8] T. Ochotta and D. Saupe, "Compression of Point-Based 3D Models by Shape-Adaptive Wavelet Coding of Multi-Height Fields", Eurographics Symp. Point-Based Graph., Zurich, Switzerland, June 2004.

[9] H. Houshiar and A. Nüchter, "3D Point Cloud Compression Using Conventional Image Compression for Efficient Data Transmission," Int. Conf. Inf. Commun. Automat. Technol., Sarajevo, B&H, Oct. 2015.

[10] G. K. Wallace, "The JPEG Still Picture Compression Standard," Commun. ACM, vol. 34, pp. 30-44, Apr. 1991.

[11] E. S. Jang et al., "Video-Based Point-Cloud-Compression Standard in MPEG: From Evidence Collection to Committee Draft [Standards in a Nutshell]," IEEE Signal Process. Mag., vol. 36, no. 3, pp. 118-123, May 2019.

[12] G. J. Sullivan *et al.,* "Overview of the High Efficiency Video Coding (HEVC) Standard," IEEE Trans. Circuits Syst. Video Technol., vol. 22, no. 12, pp. 1649-1668, Dec. 2012.

[13] R. Schnabel and R. Klein, "Octree-Based Point-Cloud Compression," Eurographics/IEEE VGTC Conf. Point-Based Graph., Boston, MA, USA, July 2006.

[14] J. Kammerl *et al*., "Real-Time Compression of Point Cloud Streams," IEEE Int. Conf. Robot. Automat., Saint Paul, MN, USA, May 2012.

[15] A. F. R. Guarda, N. M. M. Rodrigues and F. Pereira, "Point Cloud Coding: Adopting a Deep Learning-based Approach," Picture Coding Symp., Ningbo, China, Nov. 2019.

[16] A. F. R. Guarda, N. M. M. Rodrigues and F. Pereira, "Deep Learning-Based Point Cloud Coding: A Behavior and Performance Study," Eur. Workshop Vis. Inf. Process., Roma, Italy, Oct. 2019.

[17] T. Lin *et al.*, "Focal Loss for Dense Object Detection," IEEE Int. Conf. Computer Vision, Venice, Italy, Oct. 2017.

[18] J. Ballé, V. Laparra, E. P. Simoncelli, "End-to-end Optimized Image Compression," Int. Conf. Learning Representations, Toulon, France, Apr. 2017.

[19] J. Ballé *et al*., "Variational Image Compression with a Scale Hyperprior," Int. Conf. Learning Representations, Vancouver, Canada, Apr. 2018.

[20] ISO/IEC JTC1/SC29/WG1 N17345, "Common Test Conditions for Point Cloud Compression," Gwangju, China, Jan. 2018.