


RESEARCH

Open Access



# Adaptive bridge model for compressed domain point cloud classification

Abdelrahman Seleem<sup>1,2,3\*</sup> , André F. R. Guarda<sup>2</sup>, Nuno M. M. Rodrigues<sup>2,4</sup> and Fernando Pereira<sup>1,2</sup>

\*Correspondence:  
a.seleem@lx.it.pt

<sup>1</sup> Instituto Superior Técnico-  
Universidade de Lisboa, Lisbon,  
Portugal

<sup>2</sup> Instituto de Telecomunicações,  
Lisbon, Portugal

<sup>3</sup> Faculty of Computers  
and Information, South Valley  
University, Qena, Egypt

<sup>4</sup> ESTG, Politécnico de Leiria,  
Leiria, Portugal

## Abstract

The recent adoption of deep learning-based models for the processing and coding of multimedia signals has brought noticeable gains in performance, which have established deep learning-based solutions as the uncontested state-of-the-art both for computer vision tasks, targeting machine consumption, as well as, more recently, coding applications, targeting human visualization. Traditionally, applications requiring both coding and computer vision processing require first decoding the bitstream and then applying the computer vision methods to the decompressed multimedia signals. However, the adoption of deep learning-based solutions enables the use of compressed domain computer vision processing, with gains in performance and computational complexity over the decompressed domain approach. For point clouds (PCs), these gains have been demonstrated in the single available compressed domain computer vision processing solution, named Compressed Domain PC Classifier, which processes JPEG Pleno PC coding (PCC) compressed streams using a PC classifier largely compatible with the state-of-the-art spatial domain PointGrid classifier. However, the available Compressed Domain PC Classifier presents strong limitations by imposing a single, specific input size which is associated to specific JPEG Pleno PCC configurations; this limits the compression performance as these configurations are not ideal for all PCs due to their different characteristics, notably density. To overcome these limitations, this paper proposes the first Adaptive Compressed Domain PC Classifier solution which includes a novel adaptive bridge model that allows to process the JPEG Pleno PCC encoded bit streams using different coding configurations, now maximizing the compression efficiency. Experimental results show that the novel Adaptive Compressed Domain PC Classifier allows JPEG PCC to achieve better compression performance by not imposing a single, specific coding configuration for all PCs, regardless of its different characteristics. Moreover, the added adaptability power can achieve slightly better PC classification performance than the previous Compressed Domain PC Classifier and largely better PC classification performance (and lower number of weights) than the PointGrid PC classifier working in the decompressed domain.

**Keywords:** Point cloud, Classification, Coding, Compressed domain, Deep learning

## 1 Introduction

Point clouds (PCs) have emerged as an attractive 3D visual representation model, offering immersive experiences for humans and providing rich spatial information for computer vision (CV) tasks. A PC consists of a set of points in the 3D space represented by

their coordinates  $(x, y, z)$ , referred to as the PC geometry. Besides geometry, a PC may include additional information about the points, the so-called attributes, such as color and normal vectors. For realistic and immersive applications, PCs usually have a massive number of points to model the surfaces. Due to the substantial number of points, efficient PCC is crucial for practical applications, notably those involving transmission or storage. Recognizing the critical need for efficient PCC in application scenarios where interoperability is essential, the Moving Picture Experts Group (MPEG) has already established two PCC standards: geometry-based PCC (G-PCC) for static PC coding; and video-based PCC (V-PCC) for dynamic PC coding [1]. These coding solutions may be classified as ‘conventional’ since they follow the usual hand-crafted design approach adopted for decades in multimedia coding solutions.

Following the success of deep learning (DL)-based solutions for multimedia signal processing and coding, also the Joint Photographic Experts Group (JPEG) has taken the PCC challenge by launching the JPEG Pleno PCC Call for Proposals (CfP) in April 2021, aiming to specify the first DL-based PCC standard. In contrast with MPEG PCC standards which mostly target compression for human visualization, the scope of the JPEG Pleno PCC project targets both human visualization and machine consumption [2]. Despite its relatively recent development, the current JPEG Pleno PCC Verification Model (VM), which specifies the standard under development, referred to from now on as JPEG PCC, has achieved state-of-the-art compression results when compared with the MPEG PCC standards, notably for geometry-only compression of static and dense PCs, showcasing the capabilities of DL-based coding solutions [3]. However, for sparse PCs, the JPEG PCC rate-distortion (RD) performance is not so competitive with G-PCC Octree still performing better. In parallel, JPEG is also developing a DL-based image coding standard, known as JPEG AI [4] which although still under development, is already achieving more than a 30% reduction in rate compared to the powerful Versatile Video Coding (VVC) standard in its intra-coding mode [5].

Regarding the coding approach, JPEG PCC is based on the recent advancements in DL-based coding and processing for multimedia signals [6, 7]. These DL-based coding solutions adopt convolutional DL models that extract features from the spatial-temporal multimedia signal resulting in a rich latent representation. These convolutional DL models are at the core of DL-based solutions used for both multimedia coding and CV tasks. For human visualization applications, the latent representation is compressed using a DL-based entropy coding scheme (e.g., for JPEG PCC, a hyperprior model is used to estimate the probability distribution of the latent representation and improve the entropy coding compression efficiency). For CV tasks, the latent representation is processed by additional layers which are trained to perform the target task, e.g., segmentation, classification, and recognition, among others.

In PC classification scenarios, existing solutions apply the classification models to the original (uncompressed) PC. If compression is applied, due to storage or bandwidth resource limitations, the PC classifier is applied to the decompressed version of the PC, meaning that the PC content must be first decoded before being processed. This option, referred to in this paper as decompressed domain PC classification, has two main disadvantages: the first and most serious is the penalty in the classification accuracy due to the artifacts that are introduced by coding/decoding the original PC [8]; additionally,

the decoding stage computational complexity must be considered on top of the (decompressed domain) classification computational complexity.

An interesting alternative to overcome the decompressed domain CV processing drawbacks associated to decompressed domain PC classification, is to perform the processing inherent to the CV task in the compressed domain [9–16]. The use of compressed domain approaches with conventional (usually transform-based or hybrid) coding methods has been unsuccessful, since these methods generate compressed domain signal representations which are not powerful enough both for efficient compression and CV tasks processing. However, the emergence of DL-based processing and coding models have brought a bright new perspective to this field, which has been recognized by JPEG while designing the JPEG AI and JPEG PCC scopes.

To validate the JPEG Pleno PCC scope vision for compressed representations targeting both man and machines, a so-called Compressed Domain PC Classifier (CD-PCCL) solution has been proposed for compressed domain PC classification by the same authors of this paper [17]. This solution uses a DL-based classification model which is partly coincident/compatible with a highly performing spatial domain geometry-only classifier, named PointGrid PC classifier [18], applied to the JPEG PCC geometry latent representation. In practice, CD-PCCL re-uses some layers and corresponding weights from the PointGrid classifier, referred to as the partial classifier to offer a high degree of compatibility between CD-PCCL and the PointGrid classifier. This compatibility requirement results from acknowledging that there are application scenarios where both compressed and decompressed domain processing may be simultaneously needed and thus the memory footprint should be reduced; moreover, it also makes sense to adapt very high-performing spatial domain CV processors to also offer this high performance in the compressed domain.

The PointGrid PC classifier [18] has been selected for CD-PCCL as the reference spatial domain DL-based PC classifier, due to adopting a voxel-based approach, which is a mandatory requirement to match the JPEG PCC voxel-based latent representation; moreover, it is a highly performing solution with gains over the alternative voxel-based classifiers, like VoxNet [19] and OctNet [20]. The performance results in [17] demonstrate that CD-PCCL achieves gains in classification performance with a lower number of parameters when compared with decompressed domain PC classification using PointGrid, most notably for the lower rates. Despite the good classification performance and compatibility degree with the original PointGrid classifier, the CD-PCCL model accepts a single, specific JPEG PCC latent size ( $8 \times 8 \times 8 \times 128$ ) which is associated to a specific JPEG PCC coding configuration, notably regarding the sampling factor (SF) applied to the input PC. The SF parameter is the most impactful JPEG PCC coding parameter in terms of the compression performance, since it allows to achieve good compression efficiency for both dense and sparse PC across a large range of rates. By constraining the JPEG PCC coding configuration whatever the PC characteristics, the use of CD-PCCL compressed domain classification is achieved at a cost of a reduced compression efficiency, which is a severe limiting factor for practical deployment of compressed domain CV processors.

In this context, this paper proposes a novel Adaptive Compressed Domain PC Classifier (ACD-PCCL) solution that overcomes the CD-PCCL limitations, while offering the same

advantages as well as compression and classification gains. The main technical novelty regards the classification model architecture, now including a novel adaptive bridge which can accept and automatically adapt to different-size input latent representations, notably generated by different JPEG PCC coding configurations. The proposed adaptive bridge model allows ACD-PCCL to process compressed bit streams generated by different encoders or different coding configurations of a given encoder (e.g., JPEG PCC), as well as different PC precisions.

It is important to note that the proposed ACD-PCCL solution specifically uses the future JPEG PCC standard and is intended to offer some degree of compatibility with an existing spatial domain PC classifier, in this case PointGrid, notably to reduce the memory footprint. In this context, the novelty of this work is not the design of the PC codec or the original PC classifier but rather the design of a novel, adaptive compressed domain PC classifier, compatible with two highly relevant state-of-the-art DL-based solutions for PC coding and classification, notably the selected JPEG PCC codec and the PointGrid PC classifier. As a consequence, the ACD-PCCL solution proposed in this paper is aligned with the JPEG Pleno PCC vision to achieve a unified and efficient learning-based PC representation for both human visualization and machine consumption.

The experimental results show that ACD-PCCL offers advantages in the two main aspects of compressed domain PC classification: PC coding/compression and PC classification. First, ACD-PCCL allows JPEG PCC to offer PC compression performance gains when compared with CD-PCCL, since no constraints are imposed on the JPEG PCC coding configurations and, thus, the best compression configuration for each PC may be selected, depending on its characteristics. This means that the benefits on compressed domain PC classification come at no cost in compression performance, which is critical for the deployment of applications that target both machine vision and human visualization. Second, ACD-PCCL achieves PC classification performance gains over PointGrid applied to the decompressed PC, while maintaining good compatibility with the original spatial domain model, by reusing a relevant number of layers and weights from the original PointGrid model (82.91%), which has the advantage of reducing the memory footprint. Moreover, ACD-PCCL also offers slightly better PC classification performance than the previous CD-PCCL. Finally, ACD-PCCL offers complexity gains over decompressed domain PC classification, notably in terms of the total number of weights in comparison with the PointGrid model, and avoids the additional complexity associated with PC decoding.

This paper is structured as follows: after this introduction, Section 2 briefly reviews the relevant background work regarding PC geometry-only coding and classification. Section 3 describes the non-adaptive CD-PCCL classification solution, while Sect. 4 proposes the novel ACD-PCCL classification solution and the design of its adaptive bridge model. Section 5 presents the test conditions and experimental setup and Sect. 6 reports and discusses the experimental results. Finally, Sect. 7 concludes the paper and presents future work.

## 2 Background work

This section presents a brief review of the relevant background work for this paper, notably the two key components, i.e., the PC codec and the spatial domain PC classifier. Section 2.1 addresses the JPEG PCC VM [21], a DL-based PC codec which will become soon the future JPEG Pleno PCC standard. Section 2.2 addresses the selected spatial domain

DL-based PC classifier, i.e., PointGrid [18], which is used as the reference and starting point for the design of the compatible compressed domain PC classification solutions discussed in this paper, i.e., the existing CD-PCCL and the novel proposed ACD-PCCL.

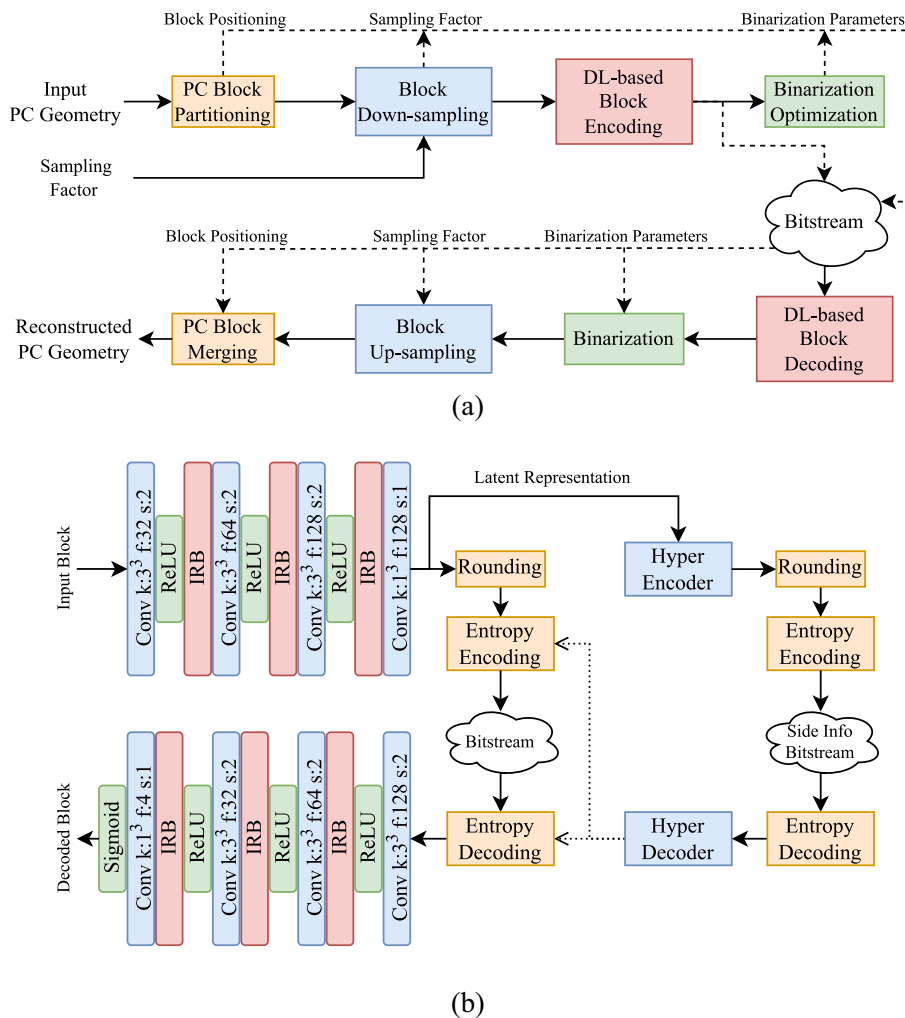
### 2.1 Deep learning-based point cloud coding: JPEG Pleno PCC verification model

The JPEG Pleno PCC VM [21] is the current coding solution under development which will soon become the JPEG Pleno PCC standard. JPEG PCC performs the coding of geometry and attributes using two different DL-based models. To foster a JPEG learning-based ecosystem, the color attributes are coded with the JPEG AI image codec, after 3D to 2D projection. Since this paper will perform PC classification using only the PC geometry, the following review will concentrate on JPEG PCC geometry coding.

The JPEG PCC high-level architecture for geometry-only coding is presented in Fig. 1a. JPEG PCC uses a DL-based model to code 3D blocks of binary voxels from the input PC, where each voxel may be either occupied (1) or empty (0). When the input PC is too large for coding with the available resources or random access is a requirement, JPEG PCC uses a PC block partitioning module to divide the input PC into fixed-size 3D blocks. After, JPEG PCC uses a block down-sampling module, which subsamples the input PC block by a given SF, which ideally should be dependent on the PC characteristics; if  $SF = 1$ , no down-sampling is performed. Down-sampling is especially important to reach lower rates or to efficiently code sparser PCs. After decoding, JPEG PCC uses a block up-sampling module, notably when SF is larger than 1, which includes an optional DL-based super-resolution model, creating the reconstructed version of the input coded PC block.

The DL-based block encoding and decoding modules are based on an autoencoder architecture, shown in Fig. 1b, which generate the latent representation for a given PC block and subsequently transform this latent representation back into the lossy reconstructed (decompressed) PC block. For this purpose, the autoencoder model is end-to-end trained using a RD-driven loss function, which effectively balances the decompressed PC distortion (estimated by the focal loss [22]) and the entropy-estimated coding rate required for the latent representation. The latent representation is encoded using a DL-based entropy coding model which uses an auxiliary hyperprior model to estimate the distribution of the latents generated by the autoencoder, thus reducing the rate required for their entropy coding. The JPEG PCC DL models are trained under the PCC Common Training and Test Conditions (CTTC) set by JPEG [23]; the JPEG CTTC defines a training dataset including 28 static PCs, each with distinctive features, notably in terms of resolution and sparsity. A Lagrangian multiplier, denoted as  $\lambda$ , controls the trade-off between quality and rate in the training process. By adjusting the  $\lambda$  value, JPEG PCC performs training to obtain a different coding model for each target quality/rate trade-off, thus defining multiple RD points. In this paper, six trained DL coding models are used, each corresponding to one of six different  $\lambda$  values, notably 0.008, 0.004, 0.002, 0.001, 0.0005 and 0.00025. For a more in-depth description of the JPEG PCC codec, refer to [21].

When comparing the geometry compression performance between JPEG PCC and the well-known MPEG G-PCC and V-PCC standards [1], the results reveal superior performance for JPEG PCC, particularly when coding static, dense PCs [21]. However,



**Fig. 1** JPEG PCC geometry codec. **a** High-level architecture. **b** DL-based block coding model architecture [21]

for sparse PCs, such as those in the ModelNet40 PC dataset adopted in this paper for classification, JPEG PCC offers a worse compression performance, notably compared to G-PCC. The previously mentioned down/up-sampling modules are critical to improve the compression performance for this type of PCs since using an appropriate SF densifies the input 3D cubic block (with a specific block size (BS) where BS is the edge size, i.e.,  $BS \times BS \times BS$ ) to offer the DL-based coding model a 3D block more similar to those used for training which are denser (see Fig. 1a) [24]. Later, at the decoder side, up-sampling is applied to restore the original PC precision using the same SF. The compression gains for this flexible down/up-sampling strategy are also associated to the use of smaller size latent representations, notably for the higher SF values.

Depending on the PC density/sparsity, the ideal SF to maximize the compression efficiency will vary and consequently also the size of the latent representation. The key technical novelty of this paper is precisely associated to the design of an adaptive PC classification model, notably a so-called bridge, which is able to accept different size latent PC representations to perform compressed domain PC classification.

### 2.2 Deep learning-based point cloud classification: PointGrid

DL-based PC classification methods may consider a variety of PC representation approaches, notably multi-view-based, point-based and voxel-based approaches as follows:

- Multi-view-based approaches: Use multiple projected 2D views of the PC to extract features and classify the PC objects (as in [25]). While projection-based approaches enhance interpretability and maintain compatibility with traditional 2D methods, they may suffer from information loss and limited spatial information compared to direct 3D processing approaches.
- Point-based approaches: Operate directly on the raw PC data, extracting features at the point level (as in [26]). Point-based approaches offer advantages such as the direct preservation of raw PC data, the ability to extract global features, and a generally lower computational complexity; however, the unstructured nature of the PC data makes it difficult to apply regular CNNs to extract local features.
- Voxel-based approaches: Represent the PC data as a binary signal in 3D space using a uniform volumetric grid (as in [18]). While voxel-based approaches offer a structured representation enabling the use of regular CNNs, they may consume significantly more memory than point-based approaches and lose fine-grained details.

These diverse representation approaches for DL-based methods offer flexibility and effectiveness in handling PC classification tasks for different use cases with varying constraints and requirements. Given the use of the JPEG PCC codec, which adopts a voxel-based approach, a voxel-based PC classifier was selected. Among the voxel-based PC classifiers, PointGrid [18] was chosen due to its better classification performance on the ModelNet40 PC dataset while maintaining low memory consumption when compared with other voxel-based PC classifiers, like VoxNet [19] and OctNet [20].

The detailed architecture of the PointGrid classifier model is presented in Fig. 2. It is composed of nine convolutional neural network (CNN) layers, each including a LeakyReLU activation function, followed by batch normalization, and three fully connected (FC) layers. These layers result in a model using a total of 10,492,072 weights. The PointGrid classifier model was trained using the ModelNet40 PC training dataset and the cross-entropy between the ground truth and predicted classes as loss function. Throughout the training process, all ModelNet40 PCs were resampled from 2048 to 1024 points each. Although PointGrid is a voxel-based classifier, it also integrates a

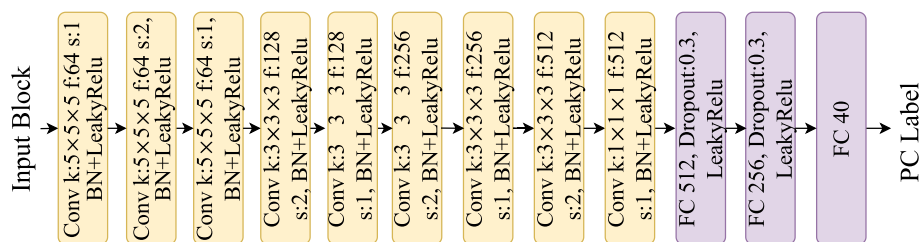


Fig. 2 PointGrid classifier architecture [18]

point-based representation by considering multiple points per voxel (cell). A unique advantage of the PointGrid classifier over other voxel-based PC classifiers is the integration of the voxel-based and point-based representations, considering larger voxels with multiple points per voxel (cell). Since the PointGrid classifier processes a given PC as a single block of a specific size with multiple points per voxel (cell), it was reported in [18] that the PointGrid classifier achieves the best classification performance for the ModelNet40 PC test dataset using blocks of size  $32 \times 32 \times 32$  with 4 points per voxel (cell). Hence, this top performing PointGrid classification configuration will be adopted in this paper.

### 3 Non-adaptive compressed domain point cloud classification: CD-PCCL

This section briefly presents the so-called DL-based Compressed Domain PC Classifier (CD-PCCL) solution, which has been previously proposed in [17], and will be used as the reference for the design of the novel ACD-PCCL solution proposed in this paper.

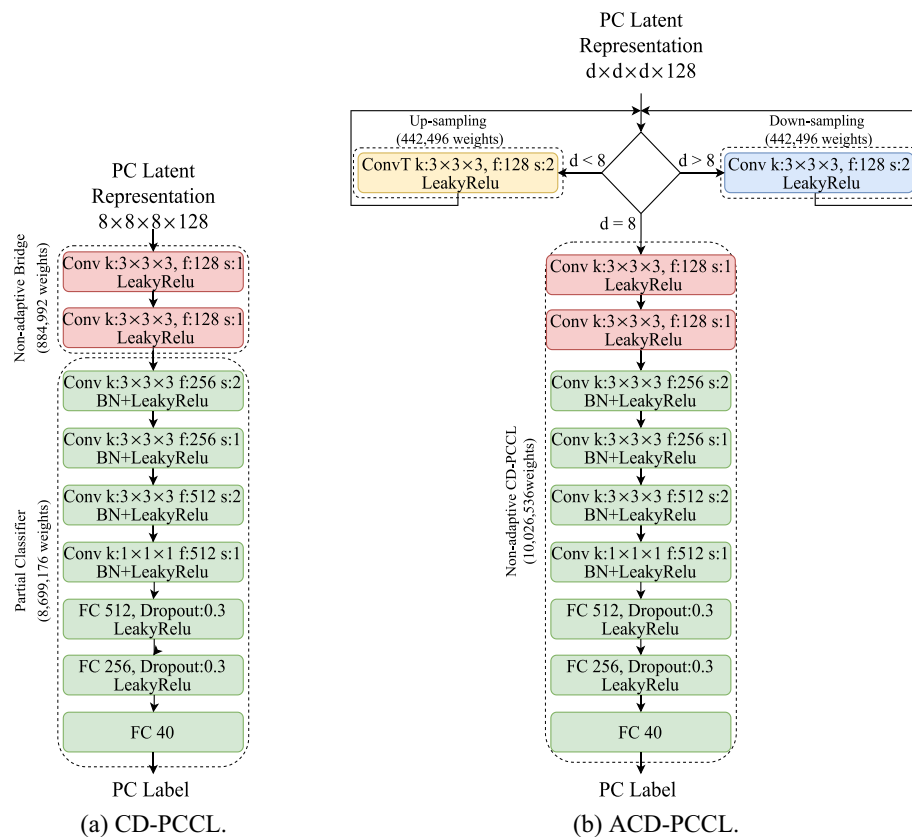
In [17], the CD-PCCL is proposed as a use case for the proposed taxonomy for the design of compressed domain CV processors, i.e., operating in the compressed domain, which are partly compatible with a spatial domain CV processor. This taxonomy is designed for a generic scenario corresponding to multiple signal modalities and DL-based CV processors which perform any CV task. In this context, several CD-PCCLs with different degrees of compatibility with the selected spatial domain PointGrid classifier were designed to demonstrate the guidance of the proposed taxonomy for the design of compressed domain CV processors. These compressed domain PC geometry classifiers adopt a two-stage model, represented in Fig. 3a, composed by:

- Partial classifier: Corresponding to a set of layers and associated weights from the reference spatial domain classifier, in the case PointGrid, thus ensuring some degree of compatibility between the compressed and decompressed domain PC classifiers.
- Bridge: Corresponding to a set of layers and associated weights which adapt/match the compressed domain representation, in this case the JPEG PCC latents, received as input to the partial classifier derived from the reference spatial domain classifier.

It is important to notice that the partial classifier and the bridge have been referred in the literature using alternative terminologies, such as “CV model back-end” and “latent space transform”, respectively [15]. The term “Bridge” has been chosen to express the fact that this model not only transforms latents (this it is a ‘latent space transform’), but it also has the specific function of enabling the ‘matching’ connection between the two latent representations, the one produced by the PC codec and the one received by the partial classifier. Moreover, the term ‘Partial Classifier’ has been used in this manuscript to make clear that this model is an unaltered part of the initial PC spatial domain classifier model, in this case PointGrid. Nevertheless, the terminology above used in this paper follows the one previously proposed for CD-PCCL [17].

Using the proposed taxonomy for design guidance, several architectures and design approaches for partly compatible CD-PCCL solutions based on the reference spatial





**Fig. 3** **a** CD-PCCL model architecture [17], which processes only fixed-size latent representations with size  $8 \times 8 \times 8 \times 128$ ; and **b** ACD-PCCL model architecture, which processes different size latent representations with size  $d \times d \times d \times 128$

domain PointGrid classifier are proposed in [17]. All the designed and assessed CD-PCCL solutions share two core design requirements that will also be adopted in this paper:

- **Compatibility:** The CD-PCCL model shall have a high degree of compatibility with the reference spatial domain PointGrid classifier, both in terms of architecture as well as weights to reduce the memory footprint. This compatibility requirement results from acknowledging that there are application scenarios where both compressed and decompressed domain processing may be simultaneously needed and thus the memory footprint should be reduced; moreover, is also makes sense to adapt very high-performing spatial domain CV processors to also offer this high performance in the compressed domain.
- **Model size:** The CD-PCCL model shall have a lower number of weights than the reference spatial domain PointGrid classifier again to limit the memory footprint.

These requirements are crucial to ensure that the final CD-PCCL solution has practical relevance, notably that it is possible to achieve memory savings by sharing model weights with the reference spatial domain PointGrid classifier and the new model has

a size, measured in terms of number of weights, that is smaller or equal to the number in the original model.

Moreover, a key design constraint, referred to as latent block size adaptation, ensures that the dimensions of the JPEG PCC latents block match the dimensions of the CD-PCCL partial classifier input latent block size. In practice, the bridge which is at the top of CD-PCCL is responsible for this adaptation. In [17], this is achieved by limiting the JPEG PCC codec to a specific coding configuration, assuring that JPEG PCC always produces an output stream corresponding to a latent block size of  $8 \times 8 \times 8 \times 128$ , which is the only block size that the CD-PCCL solution can process. It is important to notice that all the PC classifiers, i.e., PointGrid, CD-PCCL and ACD-PCCL, are geometry-only classifiers, trained using the geometry data of the geometry-only ModelNet40 dataset. As a consequence, in this paper, only the geometry coding models of the JPEG PCC codec are used.

Based on the previously presented constraint and requirements, the key steps in the design of the CD-PCCL solution [17] are: (i) defining the *partial classifier*; and (ii) designing and training the *non-adaptive bridge*. The next two subsections present these two key CD-PCCL classifier components as well as their design process.

### 3.1 Partial classifier

Given that the PointGrid classifier receives as input a block with the full PC geometry rather than a PC latent representation, it is essential to prune some initial PointGrid layers to acknowledge that feature/latents are now received at the classifier (and not a spatial domain PC). The PointGrid layers resulting from this pruning process are referred to as the partial classifier. In fact, the partial classifier corresponds to the part of CD-PCCL that is common with the reference spatial domain PointGrid classifier, thus the part that may offer some degree of compatibility between the spatial and compressed domains, see Fig. 3a. For CD-PCCL, the partial classifier was defined as corresponding to the bottom 7 layers from the spatial domain PointGrid classifier, i.e., the first top 5 layers are pruned. In summary, the partial classifier model key characteristics are:

- Input latent size:  $8 \times 8 \times 8 \times 128$ .
- Number of layers: 7, corresponding to the 7 bottom layers of the PointGrid classifier.
- Type of layers
  - 4 convolutional layers.
  - 3 FC layers.
- Activation function: LeakyReLU.
- Number of weights: 8 699 176.
- Training: No retraining, i.e., weights are initialized from the reference spatial domain PointGrid classifier; this maximizes the inter-compatibility between compressed and decompressed domains.

To ensure high compatibility between CD-PCCL and the reference spatial domain PointGrid classifier, the partial classifier not only uses the same architecture for the

bottom 7 layers of the PointGrid classifier, but also inherits the associated weights from the reference model, meaning that no retraining is performed. Although the defined partial classifier can accept a latent representation with the same size ( $8 \times 8 \times 8 \times 128$ ) of the JPEG PCC latent representation, when JPEG PCC uses a fixed configuration with  $SF = 4$  and  $BS = 64$ , using the partial classifier directly (without a bridge) and no retraining for compressed domain classification would offer terrible classification performance since the JPEG PCC latents ‘semantics’ would not match the partial classifier trained ‘semantics’. Therefore, fine-tuning the partial classifier becomes essential when no bridge is integrated. Furthermore, as demonstrated in [17], the use of a bridge in CD-PCCL to adapt the JPEG PCC latent representation to a non-retrained partial classifier (thus with more compatibility) offers better classification performance compared to CD-PCCL without the bridge and a retrained partial classifier (thus with less compatibility).

### 3.2 Non-adaptive bridge design

After the definition of the partial classifier, the next step was to design and train the bridge model to be part of CD-PCCL. The non-adaptive bridge, shown in Fig. 3a, is designed and trained to adapt the JPEG PCC fixed-size latent representation to the selected partial classifier. Note that here the bridge is referred to as *non-adaptive* to express the fact that this bridge can process only specific JPEG PCC fixed-size latent representation blocks, in this case  $8 \times 8 \times 8 \times 128$ . This fact implies major constraints on the JPEG PCC coding configurations that may be used to code all PCs, obliged to generate this latent representation size, rather than using PC specific coding configurations which would result in maximizing the compression efficiency (as will be seen later in Subsection 5.2). Since the JPEG PCC latent dimension which is passed to the non-adaptive bridge matches exactly the input dimension to the partial classifier (what does not always have to be the case), the non-adaptive bridge layers design uses stride 1 and 128 filters, meaning that, in this case, the bridge adaptation task is only in terms of latent ‘semantics’ and not in terms of the latents size. The non-adaptive bridge model key characteristics are:

- Input latent size:  $8 \times 8 \times 8 \times 128$ .
- Number of layers: 2, to minimize the compressed domain classifier model size.
- Type of layers: 2 convolutional layers.
- Activation function: LeakyReLU.
- Number of weights: 884 992.
- Output latent size:  $8 \times 8 \times 8 \times 128$ .
- Training: Training with random initialization.

The full CD-PCCL is trained by training only the two CNN layers in the non-adaptive bridge with frozen partial classifier layers (to maximize the mutual compatibility). Despite the apparent simplicity of the non-adaptive bridge architecture, which uses only 2 CNN layers, this CD-PCCL solution offers a classification performance that is better than the reference spatial domain PointGrid classification, notably for lower

coding rates in the decompressed domain. Moreover, since this bridge model has a lower number of weights than the layers pruned from the PointGrid classifier (884 992 versus 1,792,896 weights), the total number of CD-PCCL weights [17] is lower than for the original PointGrid classifier (9 584 168 versus 10 492 072 weights).

Despite the good PC classification results reported in [17], the non-adaptive CD-PCCL is limited to a specific JPEG PCC coding configuration (SF = 4 BS = 64), resulting into an  $8 \times 8 \times 8 \times 128$  latent size. Naturally, this coding configuration may not be the most suitable coding configuration for all PCs, thus leading to a compression penalty regarding the optimal coding configuration. However, simply changing the JPEG PCC coding configurations would result in generating latent representations with different latent sizes which could not be processed by the existing CD-PCCL. This CD-PCCL limitation led to the development of an adaptive ACD-PCCL classification solution, which is the key technical novelty of this paper.

#### 4 Adaptive compressed domain point cloud classification: ACD-PCCL

This section presents the proposed Adaptive Compressed Domain PC Classifier (ACD-PCCL) solution, which architecture is shown in Fig. 3b. The ACD-PCCL solution uses a novel adaptive bridge that is able to process different size latent representations generated by the JPEG PCC encoder. This is critical since it allows to use different JPEG PCC coding configurations, notably SF parameter values, thus offering: (i) an enlargement on the range of target bitrates/qualities allowed for encoding a PC by using multiple coding configurations; and (ii) better RD performance, especially when considering PCs with different levels of sparsity, by using multiple coding configurations. While higher values of SF have been shown to increase the codec RD performance for sparse PCs (such as those in the ModelNet40 PC dataset), especially at low rates, also denser PCs and higher coding rates benefit from the use of lower SF values, commonly SF = 1, i.e., no down-sampling at all [27].

##### 4.1 Adaptive bridge design

The design of the proposed ACD-PCCL solution will be based on the same constraints and requirements that guided the CD-PCCL design, notably: (i) a high degree of compatibility with the reference PointGrid PC classifier; and (ii) an overall number of weights lower or equal than that of PointGrid. To create synergies and compatibility also between the available (non-adaptive) CD-PCCL solution and the novel (adaptive) ACD-PCCL solution, the same partial classifier used in CD-PCCL is adopted for ACD-PCCL, i.e., the 7 bottom layers of the PointGrid classifier, represented in green in Fig. 3b. Using the same partial classifier architecture and weights guarantees a high degree of compatibility between CD-PPCL, ACD-PCCL and the reference PointGrid classifier.

Given the JPEG PCC codec architecture, the latent block size dimensions for any given coding configuration can be represented as a function of the latent spatial dimension  $d$ , resulting in block with size  $d \times d \times d \times 128$  where the '128' means that the JPEG PCC latent blocks consistently comprise 128 feature channels. Considering the possible JPEG PCC configurations, the possible values for  $d$  are the powers of 2 which are greater than or equal to 4. Since the CD-PCCL solution is designed for  $d = 8$ , the novel ACD-PCCL solution will be designed using  $d = 8$  as reference in the sense that the CD-PCCL bridge

is included in ACD-PCCL and latents with size  $8 \times 8 \times 8 \times 128$  (as in CD-PCCL) will go straight to the previous CD-PPCL model; on the contrary, latents with different sizes need further processing to be transformed into  $8 \times 8 \times 8 \times 128$  latents before they may again be fed to the previous CD-PPCL model.

In summary, to be able to process input latent blocks with varying sizes and produce the target output block size required as partial classifier input (which corresponds to  $d = 8$ ), the novel adaptive bridge architecture is designed in two-stages:

(A)  $d \times d \times d \times 128$  to  $8 \times 8 \times 8 \times 128$  block size adaptation: The first stage processes the JPEG PCC latent representation to change its  $d \times d \times d \times 128$  size into the previous CD-PCCL target block size ( $8 \times 8 \times 8 \times 128$ ) size. This first bridge stage, represented at the top of Fig. 3b, considers three different branches:

- Bypass branch: To be used if the input latents have already an  $8 \times 8 \times 8 \times 128$  block size which matches the target block size expected by the previous CD-PCCL bridge, i.e., the network simply bypasses this adaptation stage without altering the latents in any way;
- Up-sampling branch: To be used if  $d < 8$  (latents block size smaller than  $8 \times 8 \times 8 \times 128$ ), including iterative CNN-based layers with strided convolutions. In this branch, a transposed convolutional layer is iteratively used to up-sample the input latent block, until it matches the target block size expected by the previous CD-PCCL bridge, i.e.,  $8 \times 8 \times 8 \times 128$ . This layer uses stride 2, so that each iteration can double the value of the spatial dimension  $d$ . Iterations are repeated until  $d$  reaches 8.
- Down-sampling branch: To be used if  $d > 8$  (latents size bigger than  $8 \times 8 \times 8 \times 128$ ) including iterative CNN-based layers with strided convolutions. In this branch, a CNN-based layer with stride 2 is iteratively used to down-sample the input latent block, until it matches the target block size expected by the previous CD-PCCL bridge, i.e.,  $8 \times 8 \times 8 \times 128$ . This layer uses stride 2, so that each iteration can reduce the spatial dimension,  $d$ , by half. Iterations are repeated until  $d$  reaches 8.

(B)  $8 \times 8 \times 8 \times 128$  to partial classifier adaptation: The second stage adapts the  $8 \times 8 \times 8 \times 128$  size to the partial classifier as already done in CD-PCCL; for this reason, this second-stage bridge is composed by two CNN layers, which are common to the previous CD-PCCL bridge since the same type of adaptation has to be performed; these layers are represented in Fig. 3b in red.

The ACD-PCCL design above ensures that the final number of weights is lower than the reference spatial domain PointGrid classifier, i.e., 10 469 160 versus 10 492 072 weights.

## 4.2 Training process

The ACD-PCCL training process has a strong influence on the final classification performance and thus must be also carefully defined. In this paper, two training strategies are defined and will be assessed in Sect. 5, which can be summarized as follows:

- (A) Freezing the second-stage bridge layers, i.e., the legacy CD-PPCL bridge layers: In this training strategy, referred from now on as “Freezing”, the available pre-trained CD-PCCL bridge solution (comprising the two CNN layers inherited from the non-adaptive bridge) and the partial classifier are used without additional retraining. Consequently, the training process focuses exclusively on the first-stage bridge performing up-sampling and down-sampling adaptation in its branches. A large degree of compatibility between ACD-PCCL and CD-PCCL is offered.
- (B) Retraining the second-stage bridge layers, i.e., the legacy CD-PCCL bridge layers: In this training strategy, referred from now on as “Retraining”, all bridge layers, i.e., first and second stages, including those inherited from CD-PPCL, are trained from scratch using random initialization; the partial classifier is still frozen. A lower degree of compatibility between ACD-PCCL and CD-PCCL is offered, now reduced to the partial classifier.

The “Freezing” training strategy is naturally faster than the “Retraining” one, as it requires the training of only part of the adaptive bridge layers. However, the “Retraining” training strategy is more flexible since it is able to train all layers jointly. To maintain a consistent degree of compatibility with the reference spatial domain PointGrid classifier, the partial classifier layers always use the original PointGrid model weights, so no additional training is applied.

The training of the six CD-PCCL non-adaptive bridge models (for six different rates/qualities) followed a sequential training approach. The same sequential training approach is followed for training the new six ACD-PPCL adaptive bridge models based on the defined training strategies, i.e., freezing and retraining. In the sequential training approach, the bridge model for the highest rate (lowest  $\lambda$  value) is trained first, using random weight initialization. Then, for each subsequent rate (next  $\lambda$  value), the bridge model is initialized with the weights from the previously trained bridge model. This sequential training approach significantly reduces the training time and tends to yield better performance. It is worth noting that the existing pre-trained JPEG PCC models were also trained using sequential training.

## 5 Test conditions and experimental setup

This section describes the test conditions and experimental setup used for compression and classification performance assessment, notably:

- (A) PC classification pipelines: In practical scenarios, PC classification can occur at various domains, specifically before or after lossy coding. For each domain, an independent PC classification pipeline is adopted in this paper as follows:
- Original domain PC classification pipeline: In this pipeline, the original PC classification performance is assessed using the selected reference spatial domain PC classifier with the selected original PC dataset, i.e., no coding or additional processing is applied. In this paper, the original PC classification performance is assessed using the PointGrid PC classifier and the original ModelNet40 PC test dataset [28].

- Voxelized domain PC classification pipeline: In this pipeline, the voxelized domain PC classification performance is assessed using the selected reference spatial domain PC classifier after voxelizing the original PC dataset used in the original domain PC classification pipeline to a predefined precision, i.e., converting the original PC dataset from floating point representation to an integer representation. In this paper, the voxelized PC classification performance is assessed using the PointGrid PC classifier and the voxelized ModelNet40 PC test dataset with 8-bit precision. Since the selected PC codecs in this paper, G-PCC Octree and JPEG PCC, code voxelized PCs, it is important to assess the impact of the voxelization process on the PointGrid classification performance.
  - Decompressed domain PC classification pipeline: In this pipeline, the decompressed domain PC classification performance is assessed using the selected reference spatial domain PC classifier after coding the voxelized version of the selected original dataset. In this paper, the decompressed PC classification performance is assessed after coding the voxelized ModelNet40 PC test dataset with both the conventional G-PCC Octree and DL-based JPEG PCC codecs. For JPEG PCC, different coding models and configurations are used to code the ModelNet40 PC test dataset. Again, the reference PointGrid PC classifier is used, now with lossy decoded PCs.
  - Compressed domain PC classification pipeline: In this pipeline, the compressed domain PC classification performance is assessed directly using a compressed domain PC classifier with the latent representations of the voxelized version of the selected dataset for the selected PC codec. In this paper, the PC classification performance is assessed directly using the ModelNet40 latent test dataset from JPEG PCC using both CD-PCCL and ACD-PCCL. The voxelized ModelNet40 PC test dataset is coded with JPEG PCC, notably with specific and different coding configurations to generate fixed sizes and different block sizes latent representations for CD-PCCL and ACD-PCCL, respectively.
- (B) PC datasets: The dataset adopted in this paper is the most used dataset for PC classification, namely the ModelNet40 PC dataset [28]. This PC dataset contains a total of 12,308 PCs with only geometry, and is organized into 40 classes, further divided into training, validation, and test sub-datasets with 9840, 2048, and 420 PCs, respectively. All the PCs in the ModelNet40 dataset have a total of 2 048 points, represented using floating-point precision coordinates, with values ranging from  $-1$  to  $1$ . In the spatial domain (original, voxelized and decompressed domains) scenarios, PC resampling is performed using the Farthest Point Sampling (FPS) algorithm [29] to bring the ModelNet40 PCs to 1 024 points as needed to use the same size for which PointGrid classifier has been trained.
- (C) Training latent datasets: When training the DL-based compressed domain classifiers used in this paper, notably the existing CD-PCCL and the proposed ACD-PCCL, it is important to highlight the following:
- The CD-PCCL DL models (one model for each  $\lambda$  value) were trained using the fixed-size ( $8 \times 8 \times 8 \times 128$ ) latent representations from the ModelNet40 PC

training dataset encoded with JPEG PCC. In practice, this means that each PC was compressed using the JPEG PCC model trained for the corresponding RD trade-off, associated to a specific  $\lambda$  value. The JPEG PCC coding configuration was always set to  $SF = 4$  with  $BS = 64$  to generate the fixed-size  $8 \times 8 \times 8 \times 128$  latents, thus building the ModelNet40 latent training dataset, as described in [17].

- The ACD-PCCL DL models (one model for each  $\lambda$  value) were trained as described for CD-PCCL, but now using latent representations with different block sizes. These different size latents are associated with the coding of the ModelNet40 PC training dataset using JPEG PCC with different coding configurations, namely:
  - $SF = 8$  (corresponding to  $BS = 32$  for a full PC size of  $256 \times 256 \times 256$ ), which generates latent blocks with size  $4 \times 4 \times 4 \times 128$ ;
  - $SF = 4$  (corresponding to  $BS = 64$  for a full PC size of  $256 \times 256 \times 256$ ), which generates latent blocks with size  $8 \times 8 \times 8 \times 128$ , as for CD-PCCL;
  - $SF = 2$  (corresponding to  $BS = 128$  for a full PC size of  $256 \times 256 \times 256$ ), which generates latent blocks with size  $16 \times 16 \times 16 \times 128$ .

The selection of the ‘best’ coding configuration for each PC is based on a specific criterion presented in the next section.

- (D) Training hyperparameters: The Adam optimizer was used to train the ACD-PCCL solution, with a batch size of 32. The cross-entropy loss function between the predicted and ground truth classes was used. Early stopping was set using a patience value of 50 epochs, meaning that if there is no improvement in classification accuracy over the ModelNet40 latent validation dataset (comprising 2048 latents) after 50 epochs, the training process is halted, and the current model is considered final. The learning rate was set to  $10^{-4}$ , halving it whenever the classification accuracy over the ModelNet40 latent validation dataset stopped improving for 10 epochs. These training hyperparameters are the same as those used for CD-PCCL training.
- (E) PC geometry codecs: Two geometry-only static PC codecs were used in this paper to code the PCs from the selected ModelNet40 test dataset:

- Conventional MPEG G-PCC Octree codec: The G-PCC reference software, TMC13, version v14, in Octree mode, was used under the configurations defined in the MPEG Common Test Conditions (CTC) [30]. The G-PCC Octree decompressed PCs will only be used for the decompressed domain PC classification pipeline since there are no G-PCC compressed domain classifiers available.
- DL-based JPEG PCC codec: The JPEG PCC reference software, version v2 [21] was used in both the decompressed and compressed domains PC classification pipelines to generate decompressed PCs and the corresponding latent representations, respectively.

- (F) Compression performance metrics: The RD performance was evaluated using the PSNR D1 metric since this is the main PC geometry fidelity quality metric recom-



mended by both MPEG CTC [30] and JPEG CTTC [23]. The coding rate was measured in bits per input point (bpp).

- (G) Classification performance metrics: The evaluation of the classification performance for all tested PC classification pipelines was measured using the Top-1 and Top-5 metrics, which are the most widely adopted classification metrics in the literature. Top-1 corresponds to the percentage of test examples in which the class with the highest probability precisely matches the ground truth. Top-5 corresponds to the percentage of test examples for which the ground truth is included in the 5 classes with the highest probabilities as output by the classifier. Naturally, Top-5 is always equal to or higher than Top-1.

Table 1 provides a summary of the key characteristics associated to each PC classification pipeline. The next section will report and discuss the performance assessment for the defined PC classification pipelines.

### 6 Performance assessment

This section presents the performance assessment for the proposed ACD-PCCL solution. Subsection 6.1 reports and discusses the PC compression performance for two relevant PC geometry codecs, notably G-PCC Octree and JPEG PCC. After, Subsection 6.2 reports and analyzes the spatial domain PC classification performance for the original, voxelized and decompressed PC classification pipelines. Later, Subsection 6.3 reports and discusses the PC classification performance for compressed domain classifiers (CD-PCCL and ACD-PCCL), considering the two defined ACD-PCCL training strategies, in comparison with the reference spatial domain classification performance.

**Table 1** Summary of characteristics for each PC classification pipeline

PC classification pipeline	ModelNet40 test dataset	Data representation	PC coding		PC classification			Observations	
			G-PCC Octree	JPEG PCC	PointGrid	CD-PCCL	ACD-PCCL		
Spatial domain	Original	• Subset SF 2 • Subset SF 4	Floating point coordinates	✗	✗	✓	✗	✗	Values range is in [- 1, 1]
	Voxelized	• Subset SF 8	Integer coordinates	✗	✗	✓	✗	✗	Voxelized to 8-bit precision with scaling (values range is in [0, 255])
	Decompressed			✓	✓	✓	✗	✗	JPEG PCC with recommended and non-recommended coding configurations
Compressed		Latent representation		✗	✓	✗	✓	✓	• Specific coding configurations for CD-PCCL • Recommended coding configurations for ACD-PCCL

### 6.1 Compression performance

This subsection reports and analyzes the compression performance for the selected ModelNet40 PC test dataset with the selected codecs, notably G-PCC Octree and JPEG PCC.

Since the ModelNet40 PC test dataset is very sparse, it is very suitable for coding with G-PCC Octree. Conversely, the DL-based JPEG PCC codec has been shown to perform better than the MPEG standards, G-PCC and V-PCC Intra, when coding dense PCs [21]. To mitigate the negative impact of sparsity on the JPEG PCC compression performance, the JPEG PCC architecture includes a subsampling module which uses a tunable SF coding parameter to adjust the sparsity of the input PC blocks to the JPEG PCC DL coding model.

To determine a suitable SF to a given PC ( $X$ ), in terms of compression efficiency, the mean Euclidean distances ( $d_x$ ) between each point ( $x \in X$ ) and its five nearest neighbors are first computed, and then the median across all these mean distances is obtained to define the sparsity index (SI):

$$SI = \text{median} \left\{ d_x \mid d_x = \frac{1}{5} \sum_{k=1}^5 \|x - x_k\|^2, \forall x \in X \right\}. \tag{1}$$

Finally, SF is defined as the closest, previous power of 2 from SI:

$$SF = 2^{\lfloor \log_2 [SI] \rfloor}, \tag{2}$$

where  $\lfloor \cdot \rfloor$  denotes the mathematical floor operation. In practice, Eq. (2) defines the most suitable JPEG PCC SF coding parameter for a PC as the highest power of 2 below the median across the neighbor distance means. For example, if the median is 2.7 or 3.5, SF becomes 2; for a sparser PC with a median above 4, SF becomes 4; if the median is just above 1, SF becomes 1, meaning that no subsampling is needed since the PC is already dense enough for the trained DL coding model.

For the ModelNet40 PC test dataset, Eq. (2) only provides three different results for the SF value, notably 2, 4 or 8; this means that this dataset is so sparse that  $SF = 1$ , meaning no subsampling before coding, is never a result. The SF values resulting from Eq. (2) allow dividing the ModelNet40 PC test dataset into three distinct subsets, depending on which SF value each PC prefers to maximize its RD performance, as summarized in Table 2. This table shows that, according to Eq. (2), most PCs in the ModelNet40 PC test dataset should benefit from using for compression  $SF = 4$  or  $8$  and only very few PCs benefit from  $SF = 2$  (none for  $SF = 1$ ).

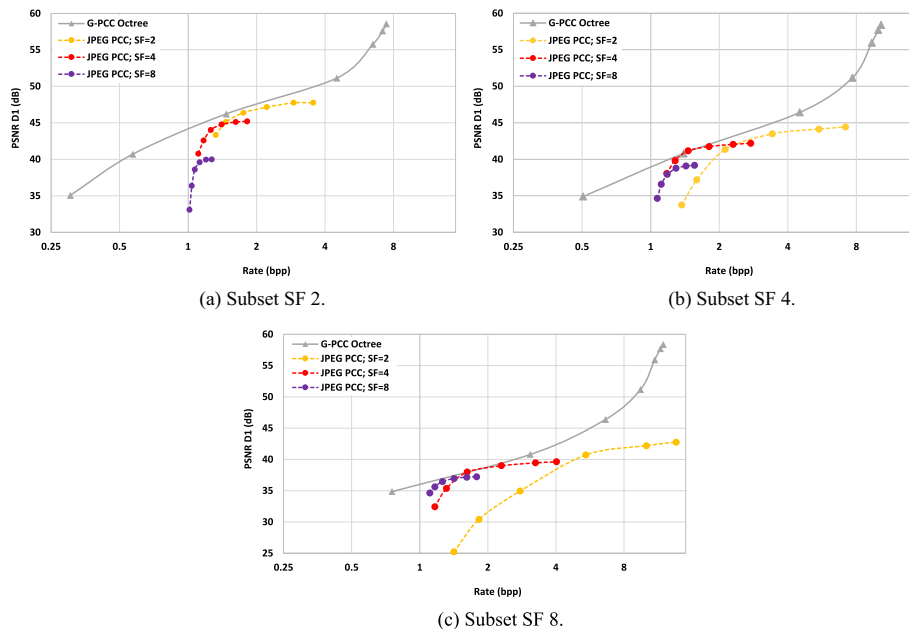
**Table 2** Splitting the ModelNet40 PC test dataset based on the SF values defined by Eq. (2)

Subset name	Number of PCs	JPEG PCC coding configurations		Latent representation size
		Recommended SF	BS	
Subset SF 2	25	2	128	$16 \times 16 \times 16 \times 128$
Subset SF 4	217	4	64	$8 \times 8 \times 8 \times 128$
Subset SF 8	178	8	32	$4 \times 4 \times 4 \times 128$

To assess the impact on JPEG PCC RD performance of using or not, for each PC, the SF value recommended by Eq. (2), the compression performance will be evaluated separately for the three subsets defined in Table 2 using SF = 2, 4 and 8; for example, for subset SF 2, which PCs should prefer being coded using SF = 2, it should be possible to see the compression penalty of using another SF value and not the recommended one.

Figure 4 shows the average RD performance for coding the three ModelNet40 PC test subsets (subset SF 2, subset SF 4 and subset SF 8) using the G-PCC Octree and JPEG PCC codecs. For JPEG PCC, three different configurations were tested, using different SF values, notably 2, 4, and 8. It is important to notice that these different JPEG PCC coding configurations produce latent representations with different sizes as shown in Table 2. The results in Fig. 4 allow concluding:

- On average, G-PCC Octree offers better RD performance than JPEG PCC, notably for the higher rates, which is due to the very high sparsity of the ModelNet40 PCs.
- Using different JPEG PCC coding configurations, notably in terms of SF parameter, has a clear impact on both the reconstruction quality and coding rate. Higher SF values allow to achieve better RD performance for very low rates, while lower SF values allow improving the RD performance for higher rates.
- On average, JPEG PCC with the recommended SF values offers better compression performance than using other SF values for coding each ModelNet40 PC test subset. As an example, one can observe that the JPEG PCC performance for subset SF 4 is much better using SF = 4 than using SF = 2 or SF = 8; a similar behavior happens for the other subsets, thus implying that each PC should be coded with its recommended SF value and not all with the same SF (notably SF = 4 as in CD-PCCL) since



**Fig. 4** RD performance for G-PCC Octree and JPEG PCC using three relevant coding configurations defined in terms of SF

this brings a compression penalty. This also validates Eq. (2) for defining the most suitable SF parameter value for each PC depending on its sparsity characteristics but also, more importantly, demonstrates the importance of being able to use different SF values when coding PCs with JPEG PCC.

By allowing the use of adaptive JPEG PCC coding configurations, particularly in terms of the SF parameter value, JPEG PCC can achieve its best RD performance for each of the sparse PCs in the ModelNet40 PC test dataset. This involves selecting the best SF parameter for each PC. However, as highlighted in Fig. 4, the JPEG PCC RD performance still falls below that of G-PCC Octree for the very sparse PCs in the ModelNet40 PC dataset. This contrasts with the results for denser, more natural PCs, where JPEG PCC outperforms G-PCC in terms of the RD performance for PC geometry [21].

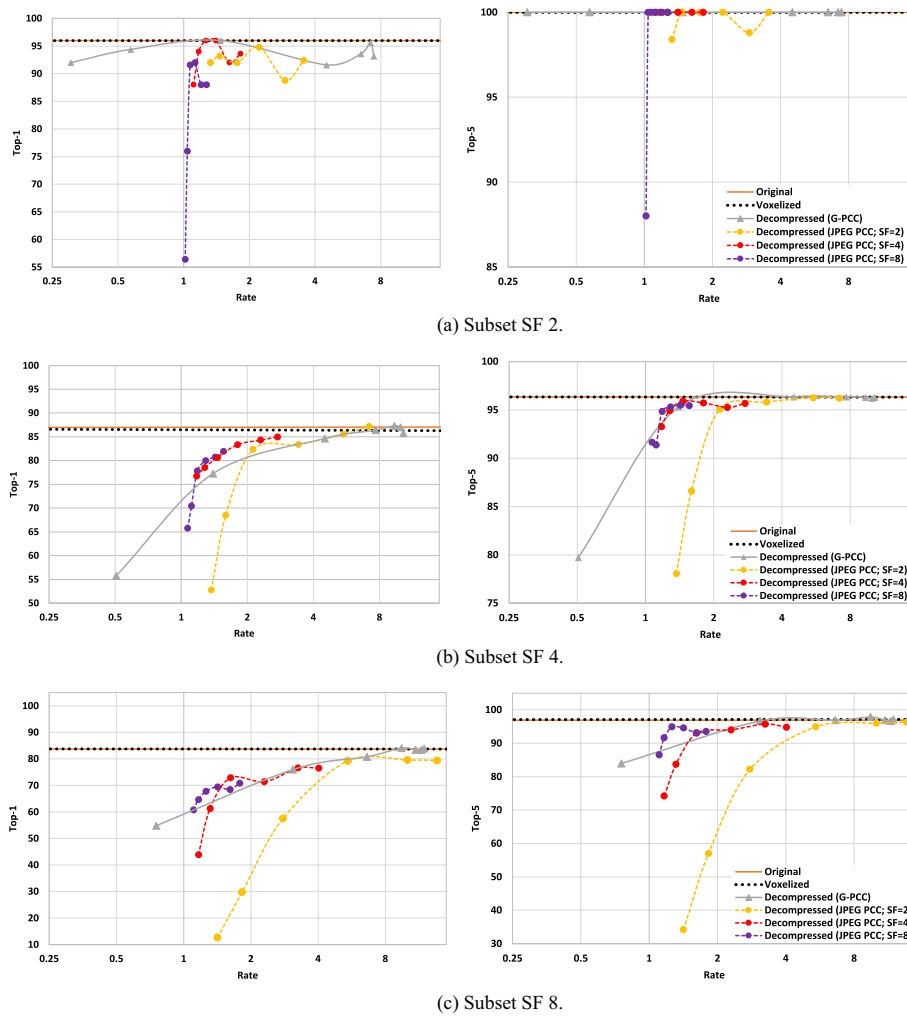
Furthermore, it is important to emphasize that JPEG PCC was not trained using the same dataset used for training the PointGrid classifier. The JPEG PCC coding model was trained according to the CTTC [23] defined by the JPEG committee, which define a training dataset and a loss function primarily intended for human visualization applications due to the importance of the performance for these applications for the deployment of the JPEG PCC standard. While it would be possible to train JPEG PCC using the same ModelNet40 PC dataset and a different loss function, e.g., considering the classification performance, with possible gains in RD performance for this dataset, and possibly also some compressed domain classification performance gains, this would likely have a negative impact on the RD performance for real-world PC content, as shown in previous studies in the literature [31]; thus, the original JPEG PCC model was used. However, this would imply losing compatibility with the JPEG PCC standard since the decoder is normative.

After defining the subsets from the selected ModelNet40 PC test dataset and analyzing the related RD performance with the recommended and non-recommended coding configurations in terms of SF value, the next subsections will present the classification performance for the three defined ModelNet40 test subsets, considering the ModelNet40 PC test subsets and the corresponding ModelNet40 latent test subsets for the spatial domain pipelines (original, voxelized, and decompressed) as well as the compressed domain pipeline, respectively.

## 6.2 Spatial domain classification performance

This subsection reports and analyzes the PC classification performance for the three spatial domain classification pipelines, i.e., original, voxelized and decompressed domains, as previously defined. Fig. 5 presents the average classification accuracy as a function of the rate (in bits per point, bpp), for the Top-1 and Top-5 classification metrics, for the three test subsets, i.e., subset SF 2, subset SF 4, and subset SF 8. The results in Fig. 5, organized by subsets, allow concluding:

- For the original and voxelized domains, PointGrid offers similar PC classification performance for both Top-1 and Top-5, for all three PC test subsets. This suggests that the voxelization of the input PCs (to 8-bit) has no major impact on the PointGrid classification performance.



**Fig. 5** Spatial domain (original, voxelized and decompressed domains) PC classification performance, Top-1 (left) and Top-5 (right), for the three ModelNet40 PC test subsets: **a** subset SF 2; **b** subset SF 4; and **c** subset SF 8

- For decompressed domain classification, when PCs are G-PCC Octree encoded, the PC classification performance follows closely the relative performance behavior observed for the G-PCC Octree RD performance, for the three test PC subsets. This indicates that, similar to the RD performance, the classification performance improves with an increasing rate. Additionally, at the highest rate points where G-PCC Octree approximates lossless quality, the classification performance for both Top-1 and Top-5 reaches the classification performance of the original and voxelized classification pipelines.
- For decompressed domain classification, when PCs are JPEG PCC encoded, the decompressed domain classification performance depends significantly on the used coding configurations, notably the SF parameter. Since coding each test subset with the recommended SF value offers better RD performance (fewer coding artifacts), see Fig. 4, than using non-recommended SF values, better classification performance is also obtained under these conditions.

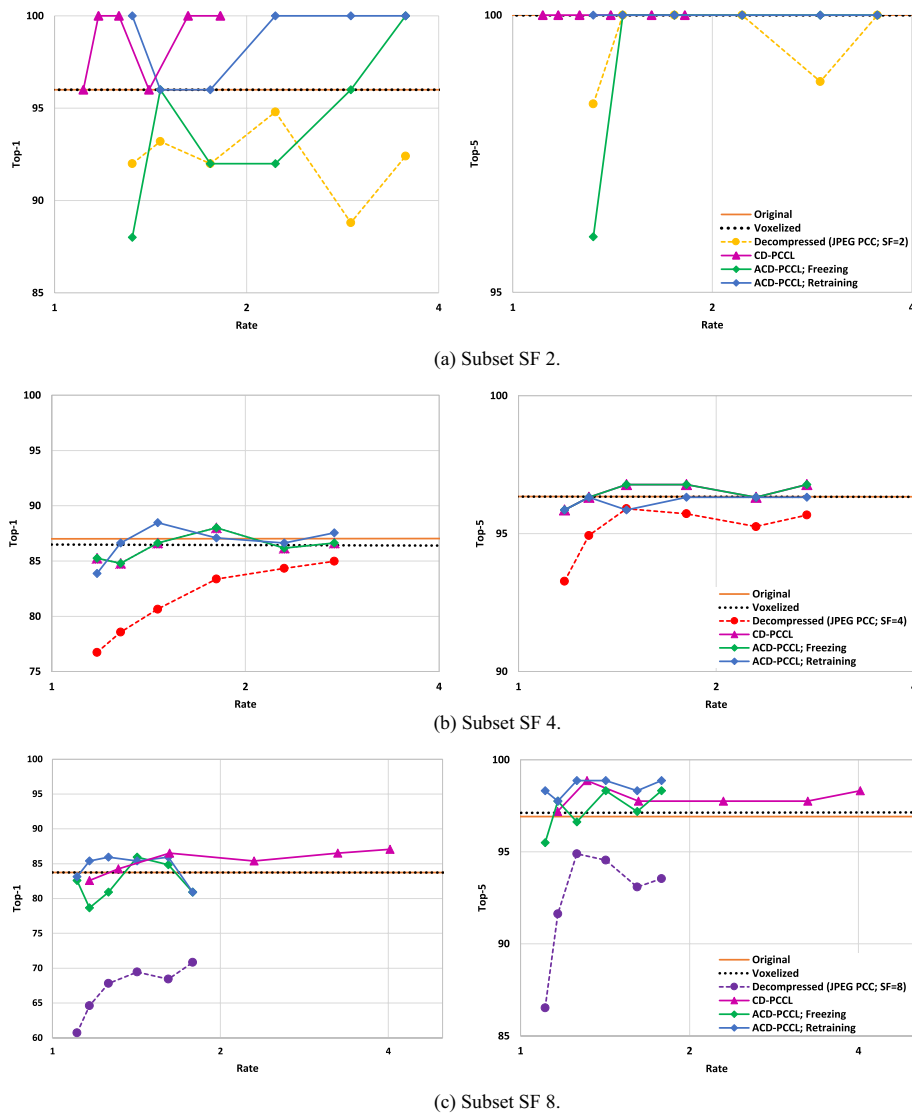
- For decompressed domain classification, using JPEG PCC coded PCs generally offers better classification performance than G-PCC Octree, notably when using the recommended SF, for both Top-1 and Top-5 classification metrics. This relative classification performance does not follow the same behavior as for the corresponding RD performance, presented in Fig. 4, where G-PCC Octree generally outperforms JPEG PCC. This suggests that the compression artifacts introduced by JPEG PCC do not seem to have an impact on PC classification performance as negative as those introduced by G-PCC Octree.
- Given the smaller size of subset SF 2, including only 25 PCs, a single incorrect prediction has a noticeable impact on the final classification performance across various rate models. This phenomenon, which may be observed for both G-PCC Octree and JPEG PCC, makes it harder to have clear conclusions in terms of the classification performance for this subset.

An important but expected conclusion of this analysis regards the impact of lossy coding on the decompressed domain PC classification. The coding artifacts present in the reconstructed PCs significantly reduce the classification performance, especially for the lower rates. As it will be shown in the next subsection, one way to reduce this impact is by performing PC classification directly on the latent representation which has been directly obtained from the original PCs.

### 6.3 Compressed domain classification performance

This subsection reports and analyzes the compressed domain PC classification performance for both the CD-PCCL and ACD-PCCL compressed domain solutions described in Sects. 3 and 4, respectively. For CD-PCCL, the JPEG PCC latent representations of all PC subsets use SF = 4 and BS = 64 since this coding configuration is imposed to generate specific fixed-size latent representations ( $8 \times 8 \times 8 \times 128$ ) for all three subsets. For ACD-PCCL, the classification performance will be presented for the two defined training strategies, i.e., freezing and retraining. Since the motivation for the development of ACD-PCCL is to allow JPEG PCC coding with a suitable SF parameter value for a given input PC, the recommended JPEG PCC coding configurations, as presented in Table 2, are used for creating the latent representations for each PC subset. As a result, the obtained JPEG PCC latent representations for each PC subset have different sizes, which means that the three ACD-PCCL model branches proposed in Sect. 4, have to be used for ACD-PCCL classification.

Figure 6 presents the Top-1 and Top-5 PC classification results for the CD-PCCL and ACD-PCCL compressed domain classifiers as a function of the bpp rate. It is important to remind that subset SF X, includes the PCs from the test dataset which offer the 'best' RD performance for SF=X according to Eq. (2) summarized in Table 2. The results in Fig. 6 allow concluding:



**Fig. 6** PC classification performance comparison, Top-1 (left) and Top-5 (right), between the proposed ACD-PCCL and the CD-PCCL solutions for the ModelNet40 PC test dataset: **a** subset SF 2; **b** subset SF 4; and **c** subset SF 8

(A) Compressed versus spatial domains

In general, for all PC subsets, the compressed domain classification performance, i.e., using latents, offers better classification performance than the spatial domain performance, notably:

- Both CD-PCCL and ACD-PCCL outperform the decompressed domain PC classification, for both Top-1 and Top-5 classification metrics, and for all test subsets. The compressed domain classification gains are clearly observable in the low-rate points for both Top-1 and Top-5 classification metrics.

- Although CD-PCCL imposes constraints on encoding subset SF 2 and subset SF 8 with JPEG PCC using SF = 4, it still offers better classification performance than decompressed domain classification performance with recommended SF parameter values.
- Occasionally, the compressed domain PC classification performance even surpasses the PC classification performance for the original and voxelized domains; this may be due to some noise filtering associated to compression.

#### (B) ACD-PCCL versus CD-PCCL

Regarding the ACD-PCCL to CD-PCCL comparison, it is important to remind that the CD-PCCL and ACD-PCCL results were obtained using JPEG PCC with a fixed SF=4 value and different SFs depending on the PC, respectively. The results in Fig. 6 allow concluding:

- For subset SF 2, the CD-PCCL (using always SF = 4) achieves similar or slightly better classification performance with lower rates than ACD-PCCL. Furthermore, the classification results shown in Fig. 6a are clearly more erratic and with larger variations than the ACD-PCCL results for SF = 4 and SF = 8 cases. This is associated to the fact that the number of PCs composing the subset SF 2 is much smaller than for the other subsets (25 PCs versus 217 and 178 PCs for subset SF 4 and subset SF 8, respectively). This implies that, as shown in Fig. 6a, the impact produced by wrongly classifying just one PC belonging to subset SF 2 (while it effectively belongs to another subset) leads to a significant variation in accuracy, with a drop from 100 to 96%. On top of this, it is also important to notice that the gains in RD performance for the CD-PCCL model (using always SF = 4), notably when compared with the ACD-PCCL model using the Retraining strategy for subset SF 2, are only observed for a very narrow low-rate range. For this subset, ACD-PCCL using the Retraining strategy offers better classification performance than the Freezing strategy.
- For subset SF 4, ACD-PCCL using the Freezing strategy is, as expected, equivalent to CD-PCCL solution since the non-adaptive bridge weights are re-used without retraining.
- For subset SF 8, once again there is a difference in coding rates between CD-PCCL and ACD-PCCL, this time with ACD-PCCL using lower rates due to the smaller block sizes. Here, ACD-PCCL tends to outperform CD-PCCL in terms of classification performance, for the same rates, though only when using the Retraining strategy.

#### (C) ACD-PCCL versus training strategies

For all latent subsets, the ACD-PCCL classification performance varies noticeably depending on the training strategy:

- For subsets SF 2 and SF 8, ACD-PCCL with the Retraining strategy outperforms ACD-PCCL with the Freezing strategy, for both the Top-1 and Top-5 classification metrics. This improvement is explained by the fact that with the Retraining



ing strategy all layers for all branches in the adaptive bridge are trained jointly and thus better optimized; on the contrary, with the Freezing strategy, the new bridge layers are conditioned to the non-adaptive bridge weights, reused from CD-PCCL without retraining.

- For subset SF 4, ACD-PCCL with the freezing and retraining strategies offer, naturally, similar performance: in fact, for this specific case, the Freezing strategy bridge only corresponds to the two CNN layers from the non-adaptive bridge, trained only with latent representations with the same size ( $8 \times 8 \times 8 \times 128$ ).

From the previous results, it may be concluded that the proposed ACD-PCCL solution is able to efficiently adapt to latents of different block sizes, while offering the same or better classification performance, when compared with CD-PCCL and decompressed domain PC classification. By using an adaptive bridge model, ACD-PCCL is able to process JPEG PCC latent representations from different coding configurations, which enables the elimination of the compression efficiency penalties associated with the imposition of a specific and sub-optimal coding configuration, as for CD-PCCL. The ACD-PPCL classification performance is better for the Retraining strategy, due to the joint optimization of all layers in the three adaptive bridge branches used in the compressed domain PC classification. The ACD-PCCL classification performance gains (notably versus decompressed domain) are particularly impressive when considering that the model complexity, measured in terms of the total number of weights, is below the reference spatial domain PointGrid classifier (10,469,160 versus 10,492,072 weights).

## 7 Conclusions and future work

This paper proposes the first coding stream adaptive compressed domain PC classification solution based on learning-based tools. This solution is targeted to coding streams compliant with the emerging JPEG PCC standard and the spatial domain PointGrid PC classifier and offers some degree of compatibility between spatial and compressed domains solutions. The adaptation capabilities regard the JPEG PCC latent representation block size which may vary depending on the coding configuration, notably the sampling factor (and consequent 3D block size). Since it is essential to vary the JPEG PCC coding configuration to reach the best compression performance for each specific PC, thus generating varying size latent representations, it is also essential to have an adaptive compressed domain classification solution that processes PC coding streams for the most relevant coding configurations. The codec-classifier adaptation is performed with a novel bridge DL-based model placed between the codec output and the so-called partial classifier. The partial classifier is designed to be largely compatible with the reference spatial domain classifier, in this case PointGrid. Experimental results show a clear compression performance advantage for the proposed ACD-PCCL solution over existing non-adaptive compressed domain classifiers, which only accept a specific latent size, meaning that some PC would have to be coded with a 'non-ideal' JPEG PCC configuration. Moreover, by improving the RD performance using the best recommended sampling factor for each PC, i.e., reaching better quality for a target rate, also the classification performance may slightly improve, increasing even more the advantage in terms of PC classification performance over traditional decompressed domain PC classifiers.

Regarding PointGrid, the ACD-PCCL classification performance gains are substantial while using a model with less weights.

While the presented results showcase positive outcomes for PC coding and classification, it is important to acknowledge that the selected JPEG PCC codec, PointGrid classifier, CD-PCCL and ACD-PCCL do have certain limitations, notably related to the adoption of a voxel-based representation which is computationally expensive, and the use of a latent representation that is optimized solely for human visualization purposes. For JPEG PCC, adopting a sparse tensor representation can overcome the complexity limitation, since only the occupied voxels and their coordinates are represented; this makes it significantly lighter in terms of computational complexity and allows the use of point-based classifiers as well as voxel-based classifiers. Future work will consider the use of alternative PC latent representations, and the use of the JPEG PCC latent representation for other compressed domain PC CV tasks, which will require adequate training datasets and CV task-dependent design strategies for the bridge.

#### Abbreviations

ACD-PCCL	Adaptive compressed domain PC classifier
BS	Block size
CD-PCCL	Compressed domain PC classifier
CNN	Convolutional neural network
CTTC	Common training and test conditions
CV	Computer vision
CfP	Call for proposals
DL	Deep learning
FC	Fully connected
FPS	Farthest point sampling
G-PCC	Geometry-based PCC
JPEG	Joint Photographic Experts Group
MPEG	Moving Picture Experts Group
PC	Point cloud
PCC	Point cloud coding
RD	Rate-distortion
SF	Sampling factor
V-PCC	Video-based PCC
VM	Verification model
VVC	Versatile video coding

#### Acknowledgements

No additional acknowledgements.

#### Author contributions

All authors designed the overall idea and proposed an experimental plan for verification. AS collected the used dataset, conducted the experiment, collecting the results and completed the writing of the first draft. AFRG, NMMR and FP conducted overall supervision of the work and determined the final version. All authors take part in the discussion of the work described in this paper. All authors read and approved the final manuscript.

#### Funding

This research was funded in whole or in part by the Fundação para a Ciência e a Tecnologia, I.P. (FCT, Funder ID=50110000187) under Grant PTDC/EEI-COM/1125/2021. For the purpose of Open Access, the authors have applied a CC-BY public copyright license to any Author's Accepted Manuscript (AAM) version arising from this submission.

#### Availability of data and materials

The mesh sampled ModelNet40 PC dataset used for the experiments is available at the link: [https://shapenet.cs.stanford.edu/media/modelnet40\\_ply\\_hdf5\\_2048.zip](https://shapenet.cs.stanford.edu/media/modelnet40_ply_hdf5_2048.zip).

#### Declarations

##### Competing interests

The authors declare that they have no competing interests.

Received: 8 December 2023 Accepted: 29 May 2024

Published online: 08 June 2024

## References

1. D. Graziosi et al., An overview of ongoing point cloud compression standardization activities: video-based (V-PCC) and geometry-based (G-PCC). *APSIPA Trans. Signal Inf. Process.* **9**, 1–17 (2020). <https://doi.org/10.1017/ATSIP.2020.12>
2. ISO/IEC JTC 1/SC29/WG1 N100097, Final call for proposals on JPEG Pleno point cloud coding. 94th JPEG Meeting, Online, Jan. 2022
3. A.F.R. Guarda, N.M.M. Rodrigues, F. Pereira, Point cloud geometry and color coding in a learning-based ecosystem for JPEG coding standards. *IEEE Int. Conf. on Image Process. (ICIP)*, Kuala Lumpur, Malaysia, Oct. 2023
4. ISO/IEC JTC 1/SC29/WG1 N100332, JPEG AI verification model 1.0 description. 97th JPEG Meeting, Online, Oct. 2022
5. ISO/IEC JTC 1/SC29/WG1 N100250, Report on the JPEG AI call for proposals results, 96th JPEG Meeting, Online, July 2022
6. J. Ballé, D. Minnen, S. Singh, S. J. Hwang, N. Johnston, Variational image compression with a scale hyperprior, in *Int. Conf. Learning Representations*, Vancouver, Canada, Apr. 2018
7. D. Minnen, J. Ballé, G. Toderici, Joint autoregressive and hierarchical priors for learned image compression, *Advances in Neural Inf. Process. Syst.*, Montreal, Canada, Dec. 2018
8. A. Seleem, A. F. R. Guarda, N. M. M. Rodrigues, F. Pereira, Impact of conventional and deep learning-based point cloud geometry coding on deep learning-based classification performance. *IEEE Int. Symp. Multimedia*, Naples, Italy, Dec. 2022
9. Y. Deng, L.J. Karam, Learning-based compression for material and texture recognition. [arXiv:2104.10065](https://arxiv.org/abs/2104.10065) [cs.CV] (2021)
10. J. Liu, H. Sun, J. Katto, Learning in compressed domain for faster machine vision tasks. *Int. Conf. Vision Commun. Image Process*, Munich, Germany, Dec. 2021
11. Z. Wang, M. Qin, Y.-K. Chen, Learning from the CNN-based compressed domain. *IEEE/CVF Winter Conf. Appl. Comput. Vision*, Waikoloa, HI, USA, Jan. 2022
12. Liu, H. Sun, J. Katto, Improving multiple machine vision tasks in the compressed domain. *Int. Conf. Pattern Recognition*, Montreal, QC, Canada, Aug. 2022
13. L. Liu et al., 2C-Net: integrate image compression and classification via deep neural network. *Multimed. Syst.* **29**, 945–959 (2022). <https://doi.org/10.1007/s00530-022-01026-1>
14. Z. Duan, Z. Ma, F. Zhu, Unified architecture adaptation for compressed domain semantic inference. *IEEE Trans. Circuits Syst. Video Technol.* **33**(8), 4108–4121 (2023). <https://doi.org/10.1109/TCSVT.2023.3240391>
15. H. Choi, I.V. Bajić, Latent-space scalability for multi-task collaborative intelligence, *IEEE Int. Conf. on Image Process. (ICIP)*, Anchorage, AK, USA, Sep. 2021
16. M. Ulhaq, I.V. Bajić, Learned point cloud compression for classification, *IEEE 25th Int. Workshop on Multimedia Signal Process. (MMSP)*, Poitiers, France, Sep. 2023
17. A. Seleem, A.F.R. Guarda, N.M.M. Rodrigues, F. Pereira, Deep learning-based compressed domain multimedia for man and machine: a taxonomy and application to point cloud classification. *IEEE Access* **11**, 128979–128997 (2023). <https://doi.org/10.1109/ACCESS.2023.3332599>
18. T. Le, Y. Duan, PointGrid: a deep network for 3D shape understanding, *IEEE Conf. Comput. Vision Pattern Recognition*, Salt Lake City, UT, USA, June 2018
19. D. Maturana, S. Scherer, VoxNet: a 3D convolutional neural network for real-time object recognition. *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, Hamburg, Germany, Sep. 2015
20. G. Riegler, A.O. Ulusoy, A. Geiger, OctNet: learning deep 3D representations at high resolutions, *IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, July 2017
21. ISO/IEC JTC 1/SC29/WG1 N100367, Verification model description for JPEG Pleno learning-based point cloud coding v1. 97th JPEG Meeting, Online, Oct. 2022
22. T.-Y. Lin et al., Focal loss for dense object detection. *IEEE Int. Conf. Comput. Vision*, Venice, Italy, Oct. 2017
23. ISO/IEC JTC 1/SC29/WG1 N100112, JPEG Pleno point cloud coding common training and test conditions v1.1. 94th JPEG Meeting, Online, Jan. 2022
24. A.F.R. Guarda et al., Deep learning-based point cloud coding and super-resolution: a joint geometry and color approach. *IEEE Trans. Multimedia* (2023). <https://doi.org/10.1109/TMM.2023.3338081>
25. H. Su, S. Maji, E. Kalogerakis, E. Learned-Miller, Multi-view convolutional neural networks for 3D shape recognition, *IEEE Int. Conf. Comput. Vision*, Santiago, Chile (2015)
26. C. R. Qi, H. Su, K. Mo, L. J. Guibas, PointNet: deep learning on point sets for 3d classification and segmentation, *IEEE Conf. Comput. Vision Pattern Recognition*, Honolulu, HI, USA (2017)
27. M. Ruivo, A.F.R. Guarda, F. Pereira, Double-deep learning-based point cloud geometry coding with adaptive super-resolution. *Eur. Workshop Vis. Inf. Process.*, Lisbon, Portugal, Sep. 2022
28. Z. Wu et al., 3D shapenets: a deep representation for volumetric shapes. *IEEE Conf. Comput. Vis. Pattern Recognition*, Boston, MA, USA, June 2015
29. Y. Zhang et al., Not all points are equal: learning highly efficient point-based detectors for 3D lidar point clouds IEEE/CVF Conf. Comput. Vis. Pattern Recognition, New Orleans, LA, USA, Jun. 2022.
30. ISO/IEC JTC1/SC29/WG11 N19084, Common test conditions for point cloud compression. Brussels, Belgium, Jan. 2020
31. E. Alexiou, K. Tung, T. Ebrahimi, Towards neural network approaches for point cloud compression. *Appl. Digit. Image Process.* XLIII,11510, Bellingham, WA, USA, Aug. 2020

## Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.