UNIVERSIDADE D
COIMBRA

Miguel Bruno dos Santos Marques

Advances in 3D Point Cloud Compression using Deep Learning

Dissertação no âmbito do Ramo de Computadores do Mestrado de Engenharia Eletrotécnica e de Computadores orientada pelo Professor Luís Alberto da Silva Cruz e apresentada ao Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Setembro de 2022

FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Advances in 3D Point Cloud Compression using Deep Learning

Miguel Marques

Coimbra, September 2022

Advances in 3D Point Cloud Compression using Deep Learning

Supervisor:

Luís Alberto da Silva Cruz

Jury:

Vítor Manuel Mendes da Silva

Luís Alberto da Silva Cruz

Nuno Miguel Mendonça da Silva Gonçalves

Dissertation submitted in partial fulfillment for the degree of Master of Science in Electrical and Computer Engineering.

Coimbra, September 2022

Acknowledgements

I want to express my sincere gratitude to my supervisor, Dr. Luís Cruz, for his guidance and support, and without whom this work would not be possible. Thank you for your patience and for pushing me to be better.

I would also like to thank all my friends and colleagues from DEEC, especially Diogo Santos, Marco Domingues, André Teixeira, Luiz Cancellier, Óscar Martins, Marta Nunes, Nuno Mendes, Tânia Gonçalves (and so many more!), for all the fun memories, advice and support. You guys rock.

To Instituto de Telecomunicações for providing me with excellent work conditions and resources.

And to my family, for their unconditional support. My mother, who always wished me the best. My father, who always listened to what i had to say. My brother, who always had my back (and i've got yours!).

Abstract

As 3D point clouds become more common as a representation of 3D visual content, the need to efficiently compress this data grows ever stronger. Research has shown that deep learning based approaches to point cloud coding see an increase in performance when compared with more traditional methods like the G-PCC and V-PCC encoders developed by MPEG. This Dissertation examines and evaluates the use of the deep learning Transformer architecture and patch-based inputs combined with well developed deep learning point cloud compression solutions described in the literature. To that end, we propose four new deep learning encoders. The obtained results show an improvement over the G-PCC Octree encoder in terms of the D1 PSNR metric, as well as an improvement over the baseline PCC Geo v2 codec. The Dissertation also presents an ablation study conducted to analyze the impact of several encoder related parameters and structures that can guide future research in deep learning point cloud compression. Finally, a study is conducted to extend current state-of-the-art deep learning point cloud compression solutions to also compress the color information of the point cloud. A detailed study is performed over the RGB, YCbCr, LAB and HSV color spaces to determine the best suited color space to compress the point clouds, while also comparing the reconstructed point clouds to the MPEG V-PCC codec baseline.

Keywords: Point cloud, compression, deep learning

Resumo

À medida que nuvens de pontos 3D se tornam mais comuns como uma representação de conteúdo visual 3D, a necessidade de comprimir eficientemente estes dados torna-se cada vez maior. Investigações evidenciam que soluções baseadas em aprendizagem profunda para codificação de nuvens de pontos resultam num aumento no desempenho comparado com métodos mais tradicionais como os utilizados nos codificadores G-PCC e V-PCC desenvolvidos pela MPEG. No contexto de compressão de nuvens de pontos baseada em aprendizagem profunda, esta Dissertação examina e avalia o uso da arquitetura de aprendizagem profunda denominada Transformer, bem como entradas do modelo profundo baseadas em *patches*. Combinando estas técnicas com soluções estado da arte na literatura de compressão de nuvens de pontos usando aprendizagem profunda, são propostos e avaliados quatro novos codificadores. Os resultados obtidos demonstram, não só um aumento de desempenho comparado com o codificador base MPEG G-PCC Octree em termos da métrica D1 PSNR, mas também um aumento no desempenho comparado com o codificador base baseado em aprendizagem profunda PCC Geo v2. Esta Dissertação também apresenta um estudo que analisa o impacto no desempenho dos codificadores propostos de vários parâmetros, com o intuito de guiar investigações futuras no tópico de compressão de nuvens de pontos baseada em aprendizagem profunda. Finalmente, é realizado um estudo com o objetivo de extender a funcionalidade de soluções estado da arte em compressão de nuvens de pontos baseada em aprendizagem profunda para também comprimir informação da cor de cada ponto da nuvem de pontos. Em termos de espaço de cor na codificação, são realizados estudos usando os espaços de cor RGB, YCbCr, LAB e HSV para determinar qual deles é o mais adequado para comprimir as nuvens de pontos. Todas as soluções exploradas são também comparadas com o desempenho do codificador base V-PCC.

Palavras chave: Nuvem de pontos, compressão, aprendizagem profunda

"Any fool can know. The point is to understand."

— Albert Einstein

Contents

List of Acronyms

AE Auto Encoder

bpp Bits-per-point

CD Chamfer Distance

CMEL ConvMixer Encoder Layer

CNN Convolutional Neural Network

DL Deep Learning

FCME Full ConvMixer Encoder

FL Focal Loss

FTE Full Transformer Encoder

G-PCC Geometry-based Point Cloud Compression

JPEG Joint Photographic Experts Group

KL Kullback-Leibler

MLP Multilayer Perceptron

MPEG Moving Picture Experts Group

MSE Mean Squared Error

PC Point Cloud

PCC Point Cloud Compression

PCME Progressive ConvMixer Encoder

PSNR Peak Signal-to-Noise Ratio

PTE Progressive Transformer Encoder

RD Rate Distortion

ReLU Rectified Linear Unit

TEL Transformer Encoder Layer

V-PCC Video-based Point Cloud Compression

VAE Variational Auto Encoder

wBCE weighted Binary Cross Entropy

List of Figures

List of Tables

1 Introduction

1.1 Context and Motivation

Point Clouds (PCs) are a data structure recognized as being of great importance in the representation of 3D visual content. They're used in several diverse applications, such as the representation of LIDAR signals, the preservation of cultural artifacts and augmented and virtual reality. In a way, PCs are the 3D extension of 2D images. A PC can be represented by a group of unordered 3D points, each one represented by their cartesian coordinates (x, y, z). Moreover, each one of these points can have multiple different attributes such as RGB color, transparency, normal vectors and/or other application specific values. However, the push to achieve realistic and immersive scenes leads to PCs having millions of points. As such, their data size can become very large very quickly, leading to the necessity to efficiently compress these data structures. There are two types of compression: lossy, when the reconstructed data is an approximation of the original data, and lossless, when the reconstructed data is equal to the original data. Both types can be achieved using machine learning, however, the majority of the scientific community is focused on lossy compression as its compression factors are much higher than those of lossless compression. The work described in this Dissertation involves only lossy compression and will be referred to as compression, unless otherwise stated.

To better understand the need to compress PCs, an example will be given. Consider a LIDAR system capturing a PC every 0.1 seconds (10 fps). For example, the PC in Figure 1.1a has geometry and color information with a total of around 6 million points. This captured PC requires 84.8 MB of storage space. However, a more typical LIDAR PC has around 100 thousand points, which requires 3×64 bits per point, resulting in about 2.3 MB per PC. Considering every PC has the same size, after only 1 minute of operation, around 1.34 GB of space are necessary to store all the captured PCs. For reference, 1 minute of 8K 24 frames-per-second video weighs about 5.76 GB on disk. It is clear that a compression of the captured PCs has to be employed to allow sustainable continuous use of the LIDAR generated data.

(a) Example of a LIDAR point cloud captured by a drone. (b) Example of a static point cloud.

Figure 1.1: Point cloud examples.

So far, we have been comparing PCs with 2D images, however, that comparison only holds true for static PCs. Akin to 2D video, PCs can also be dynamic. More specifically, a dynamic PC can be described as a sequence of static PCs. That being said, the focus of this Dissertation will be on the former, i. e., static PCs, like those represented in Figure 1.1. Unless otherwise stated, in this Dissertation the term PC will mean static PC.

The Moving Picture Experts Group (MPEG) developed a coding standard known as Geometry-based Point Cloud Compression (G-PCC) [26] designed to encode static and dynamic PCs. G-PCC rate-distortion (RD) performance leaves much to be desired compared with other solutions in the literature, like those based on deep learning (DL) approaches. Currently, state-of-the-art compression of PCs uses machine learning-based solutions.

1.2 Objectives

DL-based coding is a common research topic in current times and its application to PC compression is no exception. Various novel research works on this topic are being published with weekly new contributions. The objective of this Dissertation is to also contribute to that research effort. In this context, the main objectives of this Dissertation are as follows:

- Study the state-of-the-art in PC geometry compression and the latest advances in DL-based

PC coding.

- Study the latest advances in novel DL-based architectures in computer vision tasks.
- Develop novel DL-based static PC geometry (and attribute) compression with the aim to achieve the best possible RD performance.

To evaluate all the developed static PC geometry compression solutions, their performance will be compared with the MPEG G-PCC and Video-based Point Cloud Compression (V-PCC) standards and with other relevant compression solutions found in the literature.

1.3 Scientific Contributions

Part of the work developed in this Dissertation has been presented as a conference communication:

- **M. Marques** and L. Cruz, "Explorations on 3D point clouds coding using transformers and patches," European Workshop on Visual Information Processing (EUVIP), Lisbon, Portugal, September 2022.

1.4 Dissertation Outline

Considering the main objectives described in Section 1.2, this Dissertation is organized as follows:

- **Chapter 1** (this chapter) presents the context and motivates the objectives for this Dissertation.
- **Chapter 2** presents several core topics necessary for the understanding of the structure and performance of the solutions proposed in the following chapter.
- **Chapter 3** reviews the DL point cloud compression (PCC) works in the literature, describing the state-of-the-art solutions.
- **Chapter 4** proposes a geometry only PCC solution based on an end-to-end DL coding model and studies its performance compared with relevant baselines.
- **Chapter 5** proposes an extension to published DL-based geometry only PCC to also compress PC color information and studies its performance compared with relevant baselines.
- **Chapter 6** presents conclusions from the results of the developed solutions and proposes ideas for future work.
- **Annex A** presents results from the proposed solutions in Chapter 4.

- **Annex B** presents color conversion transformations used in Chapter 5.
- **Annex C** presents the article indicated in Section 1.3.

2 Background Information

2.1 Point Cloud Acquisition

PC acquisition is generally classified into direct and indirect methodologies.

Direct acquisition refers to systems specifically designed to capture 3D information like sparse points in 3D space or 2D images with pixel-per-pixel depth values. This definition includes the previously mentioned LIDAR system as well as time-of-flight cameras, kinect cameras and RealSense cameras. As an example, the LIDAR system obtains 3D information by sending several laser pulses and measuring the structure and time profiles of the returning signals.

Indirect acquisition is defined by methodologies that do not directly measure 3D information. The most common example are algorithms that extract depth information from a set of several standard 2D images using triangulation techniques. Once depth information is generated, a PC representation of the scene can be achieved given knowledge of the camera used in the capture process.

2.1.1 Point Cloud Voxelization

Usually, either in direct or indirect acquisition, the PC data structure is represented as a set of cartesian coordinates represented as real numbers, i.e., with $x, y, z \in \mathbb{R}$. However, common PCC applications require that the PC point coordinates are normalized to function correctly. In these applications, the coordinates of the PC are represented as integers. To achieve this normalization, a voxelization operation can be performed. A voxelization with a depth of d bits transforms the interval of variation of the (x, y, z) coordinates of an input PC into the interval $[0, 2^d - 1]$, with usual voxelization depths $d \in \{9, 10, 11\}$. For example, with a voxelization of depth of 10, it is possible to achieve a precision of approximately 1 millimeter in a 1 meter cube.

$$x'_i = \frac{x_i - N_{min_x}}{N_{max} - N_{min_x}}; \quad y'_i = \frac{y_i - N_{min_y}}{N_{max} - N_{min_y}}; \quad z'_i = \frac{z_i - N_{min_z}}{N_{max} - N_{min_z}} \tag{2.1}$$

Eq. 2.1 shows the three component normalization expressions where N_{min_i} is the smallest value of the i coordinate and N_{max} is the largest value of all the coordinates. To finalize the voxelization, the resulting coordinates of the min max operation are multiplied by $2^d - 1$ (based on the bit depth

d) and rounded to the nearest integer. Lastly, duplicated points are removed. When also handling color or other similar attributes, for each voxel with duplicated points, for example, the average of the respective values can be used as the final attribute value for the point. An example of 1D voxelization can be seen in Figure 2.1.

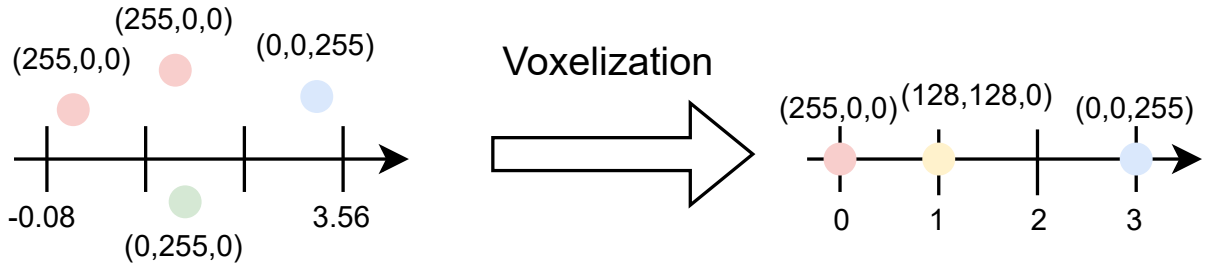


Figure 2.1: Example of 1D voxelization with a bit depth of 4. The three values next to each point are the RGB color components.

Voxelization a spatial quantization operation, where the coordinates are transformed from a continuous space to a discrete set ($[0, 2^d - 1]$) of 2^d values. That is also why the removal of duplicated points is necessary since they can be created by the quantization operation applied to the point coordinates.

2.2 Point Cloud Quality Metrics

To compare two PCs, several metrics can be used. In terms of lossy PCC, metrics relative to its geometry and color distortion are of utmost importance. Since the reconstructed (after compression) PCs have differences/distortions relative to the original PC, it is important to be able to quantify this distortion. In this Dissertation, several standard objective metrics will be presented, in particular, those based on point-to-point and point-to-plane distances, also known as the D1 and D2 metrics

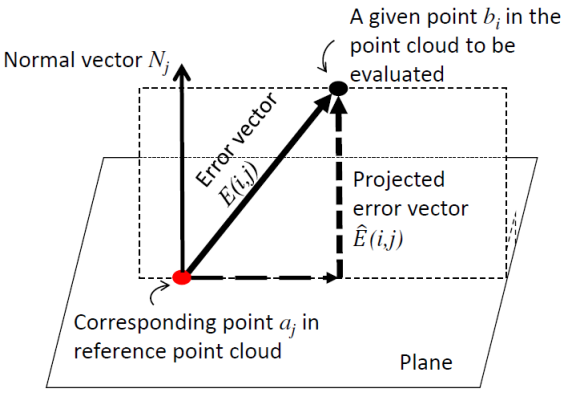


Figure 2.2: Point-to-point distance vs. point-to-plane distance. *Source:* [27]

[27], respectively. Figure 2.2 shows how these distances are measured, for a specific PC point.

These were the metrics chosen to assess the geometry distortion of two PCs in this work, but other metrics could also be used, such as the plane-to-plane and haussdorf metrics. In terms of color distortion, the standard point-to-point C1, C2 and C3 [25] will be presented. Finally, a metric that combines both geometry and color distortion, denominated PCQM [18], will be presented. Ideally, these objective metrics should have a strong correlation to subjective PC quality metrics to better reflect the distortion perceived by human observers. More specifically, an increase in a subjective metric should result in an increase in the objective metric. An example of this correlation can be seen in Figure 2.3. In this case, objective metric 1 exhibits a strong correlation to the subjective metric, while objective metric 2 displays a weak correlation to the subjective metric. Therefore, the objective metric 1 would be more indicative of the distortion perceived by the general public.

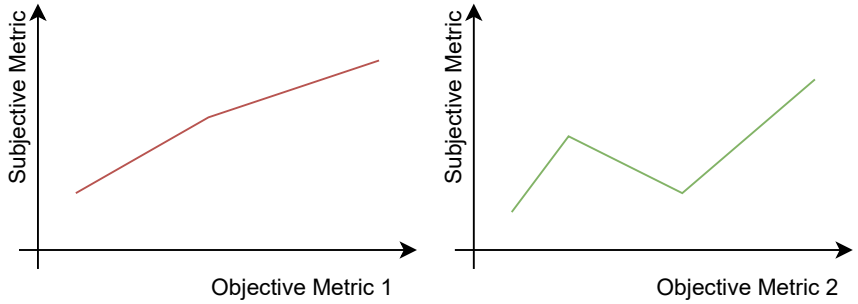


Figure 2.3: Example of a strong and weak correlation between a subjective metric and a objective metric.

As for subjective geometry and color distortion metrics, like the mean opinion score metric, no study will be conducted. Usually, to obtain this type of metrics, a pool of several original versus reconstructed PCs is created and a sample is presented to a general public audience. Each person would then grade the reconstructed PC on a quality scale, like 1 to 5, of how much they think the reconstructed PC resembles the original PC. Finally, a mean opinion score for a certain PC pair

(a) Original.

(b) Reconstructed.

Figure 2.4: Example of a pair of original vs. reconstructed point clouds.

can be calculated. In Figure 2.4, an example of a subjective PCC test is shown. If given a scale of 1 (very annoying) to 5 (imperceptible) to grade the reconstructed PC [3], i would give this PC a score of 1. The subjective metric for this pair would then be the average of all the observers scores.

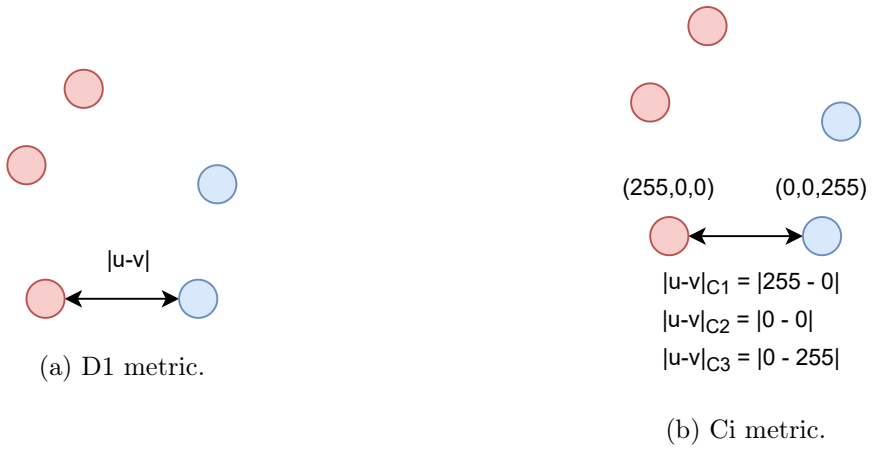
Finally, graphs of the rate distortion performance for a certain codec will be compared by calculating their Bjontegaard Deltas (BDs) [2] for the rate and distortion metric.

2.2.1 Point-to-Point Metrics

Considering a PC A and a PC B, the calculation of the D1 metric can be expressed as a mean squared error (MSE) or a peak signal-to-noise ratio (PSNR). The D1 MSE metric, from which the D1 PSNR metric is derived, measures the mean squared error of distances between the points in PC A and the closest points in PC B. The D1 MSE metric is defined in eq. 2.2

$$D1_{MSE} = \frac{1}{N_A} \sum_{u \in PC_A} \min_{v \in PC_B} \|u - v\|^2 \tag{2.2}$$

where u and v are PC points and N_A is the number of points in PC A. In Figure 2.5a, an example of the application of the D1 metric can be observed, where red points are points of the original PC and blue points are points in the reconstructed PC.



(a) D1 metric.

(b) Ci metric.

Figure 2.5: Point-to-point metrics.

In terms of the D1 PSNR metric, which is the version of the D1 metric used in this Dissertation, it can be defined by eq. 2.3

$$D1_{PSNR} = 10 \log_{10} \left(\frac{p^2}{D1_{MSE}} \right) \tag{2.3}$$

where $p = \max(\min(x_{max} - x_{min}), \min(y_{max} - y_{min}), \min(z_{max} - z_{min}))$ is the peak value of the geometry coordinates (for a voxelized PC, $p = 2^d - 1$). Specifically, the D1 PSNR metric is the logarithm of the peak value of the geometry over the mean squared error of distances between the points in PC A and the closest points in PC B. This metric is asymmetric, that is, calculating the

D1 PSNR with PC A in relation to PC B is different to calculating the D1 PSNR with PC B in relation to PC A. To obtain a symmetric metric, the worst value of the two is chosen as the final metric.

For the C1, C2 and C3 color distortion metrics, the methodology is similar to the D1 metric. The Ci MSE metric measures the mean squared error of distances between the i color component of the points in PC A and the i color component of the closest points in PC B. The Ci MSE metric is defined in eq. 2.4

$$Ci_{MSE} = \frac{1}{N_A} \sum_{u \in PC_A} \min_{v \in PC_B} \|u - v\|_{Ci}^2 \tag{2.4}$$

where u and v are PC points, N_A is the number of points in PC A and $\|u - v\|_{Ci}$ is the distance between the i color component of the u and v points. In Figure 2.5b, an example of the application of the Ci metric can be observed, where red points are the original and blue points are reconstructed with the three values over the points being the RGB components.

The Ci PSNR metric, which is the version of the Ci metric used in this Dissertation, is defined by eq. 2.5

$$Ci_{PSNR} = 10 \log_{10} \left(\frac{p_{Ci}^2}{Ci_{MSE}} \right) \tag{2.5}$$

where $p = Ci_{max} - Ci_{min}$ is the peak value of the i color component (for the RGB color space, $p_{Ci} = 255$). Specifically, the Ci PSNR metric is the logarithm of the peak value of the i color component over the mean squared error of distances between the i color component of the points in PC A and the i color component of the closest points in PC B. To obtain a symmetric metric, the worst value of the Ci PSNR of PC A in relation to PC B and the Ci PSNR of PC B in relation to PC A is considered as the final metric.

2.2.2 Point-to-Plane Metric

The previously mentioned point-to-plane D2 metric is calculated similarly to its point-to-point counterpart. However, instead of measuring the distance between two points, this metric measures the distance between a point and a plane defined by a point of the other PC and tangent to the implicit surface of the PC. The D2 MSE metric is defined by eq. 2.6

$$\begin{aligned} D2_{MSE} &= \frac{1}{N_A} \sum_{u \in PC_A} \min_{v \in PC_B} \|u - v_{plane}\|^2 \\ \|u - v_{plane}\| &= \vec{n}_v \cdot (v - u) \end{aligned} \tag{2.6}$$

where u and v are PC points, N_A is the number of points in PC A and $\|u - v_{plane}\|$ is the distance between the point u and the plane tangent to point v. This point-to-plane distance can be expressed

by the dot product between the associated normal vector $\vec{n}_v$ and the vector $v - u$. If point normal information is not present in the PC, it can be calculated by using information of the point in question and its neighbors [23]. In Figure 2.6, an example of the application of the D2 metric can be observed, where red points are the original and blue points are reconstructed.

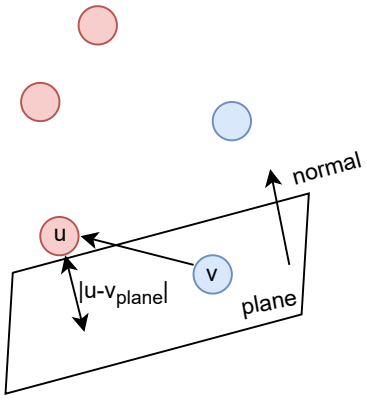


Figure 2.6: Point-to-plane D2 metric.

In terms of the D2 PSNR metric, which is the version of the D2 metric used in this Dissertation, it can be defined by eq. 2.7

$$D2_{PSNR} = 10 \log_{10} \left(\frac{p^2}{D2_{MSE}} \right) \tag{2.7}$$

where $p = \max(\min(x_{max} - x_{min}), \min(y_{max} - y_{min}), \min(z_{max} - z_{min}))$ is the peak signal of the geometry (for a voxelized PC, $p = 2^d - 1$). Specifically, the D2 PSNR metric is the logarithm of the peak signal of the geometry over the mean squared error of distances between the points in PC A and the planes of the closest points in PC B. To obtain a symmetric metric, the worst value of the D2 PSNR of PC A in relation to PC B and the D2 PSNR of PC B in relation to PC A is considered as the final metric.

2.2.3 PCQM Metric

The PCQM metric is a metric that evaluates the combined effects of both geometry and color distortion. The metric is an optimally-weighted linear combination of PC geometry and color features that attempts to mimic subjective PC metrics like the previously mentioned mean opinion score.

In the D1 and D2 metrics, considering a reference PC A and a distorted PC B, points in PC A were compared with the closest neighbor point in PC B. However, in PC B, the PCQM metric considers a projected point result of a quadratic surface fitting computed on a set of points. The points considered for the quadratic surface fitting belong to a spherical neighbourhood $N(\hat{p}, h)$ of center $\hat{p}$ and radius h. In this context, $\hat{p}$ is the nearest neighbor in PC B of the point p in PC A. An example of this projected point can be viewed in Figure 2.7.

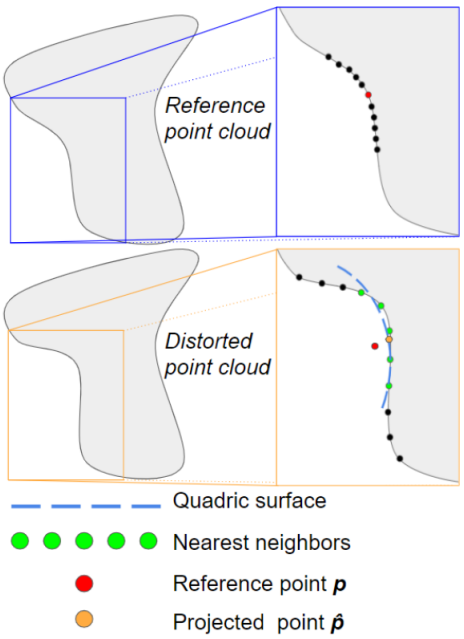


Figure 2.7: Ilustration of the point-to-surface correspondence computation. *Source:* [18]

The quadratic fitting is achieved by a least squares fitting of the quadratic surface $Q(x, y) = ax^2 + by^2 + cxy + dx + ey + f$ minimising

$$\sum_i \|z_i - Q(x_i, y_i)\|^2 \tag{2.8}$$

where (x_i, y_i, z_i) are points in the neighborhood $N(\hat{p}, \frac{h}{2})$.

Three geometry-based features, curvature comparison (f_1), curvature contrast (f_2) and curvature structure (f_3) are used. These features are based on the mean curvature information ρ that can be directly computed from the coefficients of the fitted quadratic surfaces:

$$\rho = \frac{a(1+d^2) + b(1+e^2) - 4abc}{(1+e^2+d^2)^{\frac{3}{2}}} \tag{2.9}$$

Two curvature values are calculated for each point p belonging to PC A: ρ_p, computed using a quadratic fitted over the local neighborhood $N(p, \frac{h}{2}) \in PC_A$, and $\rho_{\hat{p}}$, computed using a quadratic fitted over the local neighborhood $N(\hat{p}, \frac{h}{2}) \in PC_B$. Finally, the geometry features can be calculated:

$$\begin{aligned} f_1^p &= \frac{\|\mu_p^\rho - \mu_{\hat{p}}^\rho\|}{\max(\mu_p^\rho, \mu_{\hat{p}}^\rho) + k_1} \\ f_2^p &= \frac{\|\sigma_p^\rho - \sigma_{\hat{p}}^\rho\|}{\max(\sigma_p^\rho, \sigma_{\hat{p}}^\rho) + k_2} \\ f_3^p &= \frac{\|\sigma_p^\rho \sigma_{\hat{p}}^\rho - \sigma_{p\hat{p}}^\rho\|}{\sigma_p^\rho \sigma_{\hat{p}}^\rho + k_3} \end{aligned} \tag{2.10}$$

where k_i are small constants used to avoid instability when denominators are close to zero, μ_p^ρ and $\mu_{\hat{p}}^\rho$ are gaussian weighted averages of curvature over the 3D points belonging to neighborhoods

$N(p,h) \in PC_A$ and $N(\hat{p},h) \in PC_B$, respectively, and σ_p^ρ, $\sigma_{\hat{p}}^\rho$ and $\sigma_{p\hat{p}}^\rho$ are standard deviations and covariance of curvature over these neighborhoods.

Color quality is estimated from five color-based features, lightness comparison (f_4), lightness contrast (f_5), lightness structure (f_6), chroma comparison (f_7) and hue comparison (f_8). The color space of the PC is first converted to the LAB color space, associating each point p with a lightness L_p and two chromatic values a_p and b_p. A chroma value $c_p = \sqrt{a_p^2 + b_p^2}$ is also defined. Finally, the color features can be calculated:

$$\begin{aligned} f_4^p &= 1 - \frac{1}{k_4(\mu_p^L - \mu_{\hat{p}}^L)^2 + 1} \\ f_5^p &= 1 - \frac{2\sigma_p^L \sigma_{\hat{p}}^L + k_5}{\sigma_p^{L^2} + \sigma_{\hat{p}}^{L^2} + k_5} \\ f_6^p &= 1 - \frac{\sigma_{p\hat{p}}^L + k_6}{\sigma_p^L \sigma_{\hat{p}}^L + k_6} \\ f_7^p &= 1 - \frac{1}{k_7(\mu_p^c - \mu_{\hat{p}}^c)^2 + 1} \\ f_8^p &= 1 - \frac{1}{k_8 \overline{\Delta H_{p\hat{p}}}^2 + 1} \end{aligned} \tag{2.11}$$

where k_i are constants, $\Delta H_{p\hat{p}} = \sqrt{(a_p - a_{\hat{p}})^2 + (b_p - b_{\hat{p}})^2 - (c_p - c_{\hat{p}})^2}$ and $\overline{\Delta H_{p\hat{p}}}$ is the gaussian weighted average over $N(p,h)$.

For each of these metrics, the mean over all the points in PC A is calculated according to eq. 2.12, where N_A is the number of points in the PC A and f_i^p is the value of the feature i of the p point in the PC A.

$$f_i = \frac{1}{N_A} \sum_{p \in PC_A} f_i^p \tag{2.12}$$

Finally, the PCQM metric can be obtained as a weighted average of the individual features, following eq. 2.13

$$PCQM = \sum_{i \in S} w_i f_i \tag{2.13}$$

where S is the set of indices of the pertinent PC features, w_i is the weight given to feature i determined through cross-validation based on results from a subjective PCC metric, and f_i is the feature i.

2.2.4 Bjontegaard Deltas Metric

When comparing two encoders at multiple RD operation points, the BDs are important metrics to consider. More specifically, in the context of compression, this metric will be used to compare the performance of two encoders.

Both the BD-PSNR and BD-rate will be calculated. The BD-PSNR measures, on average, how much higher in terms of the PSNR metric one RD curve is compared to the other. Considering a comparison between an encoder 1 and an encoder 2, the BD-PSNR will be

$$BD_{PSNR} = \frac{A_2 - A_1}{r_2 - r_1} \tag{2.14}$$

where A_1 is the area below the RD curve of encoder 1, A_2 is the area below the RD curve of encoder 2 and $[r_1, r_2]$ is the interval where the two RD curves overlap. In Figure 2.8, an example of the concept of the BD-PSNR application and computation is presented. The BD-PSNR is measured in dBs.

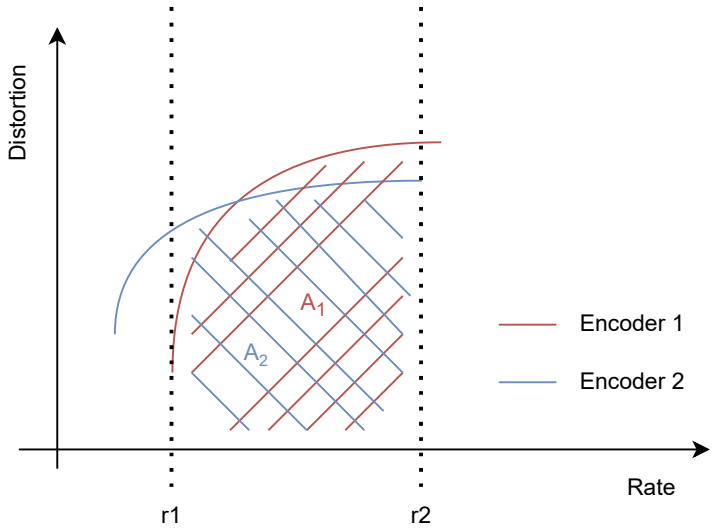


Figure 2.8: Example of the computation of the BD-PSNR metric. In this case, $A_2 - A_1$ will be negative.

In contrast, the BD-rate measures, on average, how much higher in terms of the rate one RD curve is compared to the other. The calculation of the BD-rate is similar to that of the BD-PSNR, however, the areas and interval considered are measured with reference to the distortion axis (area to the left of the RD curve).

$$BD_{rate} = \frac{A_2' - A_1'}{d_2 - d_1} \tag{2.15}$$

In Figure 2.9, an example of BD-rate computation is presented.

Usually, in compression performance studies, the BD-rate is expressed in percentage, so that will also be the methodology followed in this Dissertation.

Considering that the BDs are calculated based on RD curves, the RD points obtained from a certain codec need to be curve fitted. Any curve fitting can be applied as long as it gives reasonable results, however, since the RD curves usually follow a logarithmic curve, a third degree logarithmic

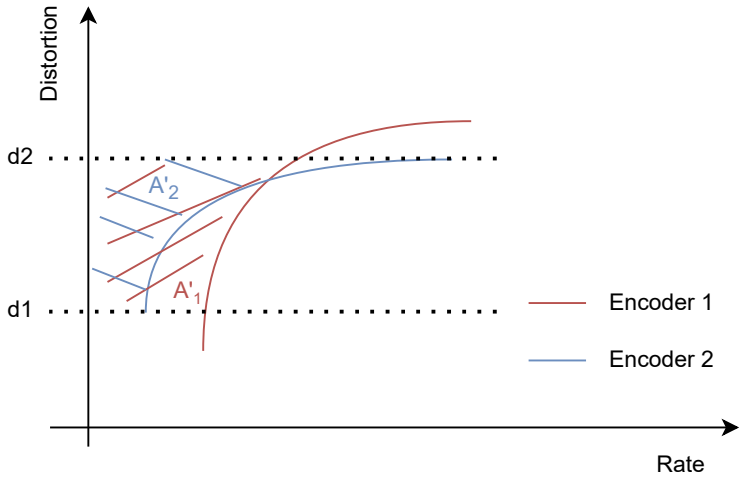


Figure 2.9: Example of the application of the BD-rate metric. In this case, $A'_2 - A'_1$ will be negative, because, in the considered interval, the area to the left of the RD curve 2 is smaller than the area to the left of the RD curve 1.

fit is often performed. Given a rate R and a distortion D, a least square polynomial fitting is applied according to eq. 2.16.

$$D = a\log^3(R) + b\log^2(R) + c\log(R) + d \tag{2.16}$$

The BDs can then be calculated based on the fitted curves. Note that, since it is a third degree fitting, at least four RD points are necessary to not incur overfitting. Finally, in terms of BD-rate, to avoid calculating areas to the left of the RD curves, a least squares polynomial fitting is applied following eq. 2.17.

$$\log(R) = aD^3 + bD^2 + cD + d \tag{2.17}$$

Consequently, the areas pertaining to the calculation of the BD-rate are now under the fitted polynomial. Therefore, the BD-PSNR is calculated using the curves from eq. 2.16 and the BD-rate is calculated using the curves from eq. 2.17.

2.3 Deep Learning Fundamentals

DL is a branch of machine learning that leverages multiple transformation layers, like convolutional layers, with backpropagation [22] and gradient descent techniques to optimize a set of weights that process an input to obtain a desired output. Usually, for 2D or 3D inputs, convolutional neural networks (CNNs) are used. These networks have the objective of extracting features from the given inputs by convolving them with arbitrary kernels. The weights of the kernels are learned when training the network. In turn, the resulting features can be used, for example, to classify the input

into a set of different categories. The arrays of features from the different layers of the feature extraction step are often referred to as *latent representations*. In Figure 2.10, a well known CNN architecture to process 2D images called AlexNet [16] is presented.

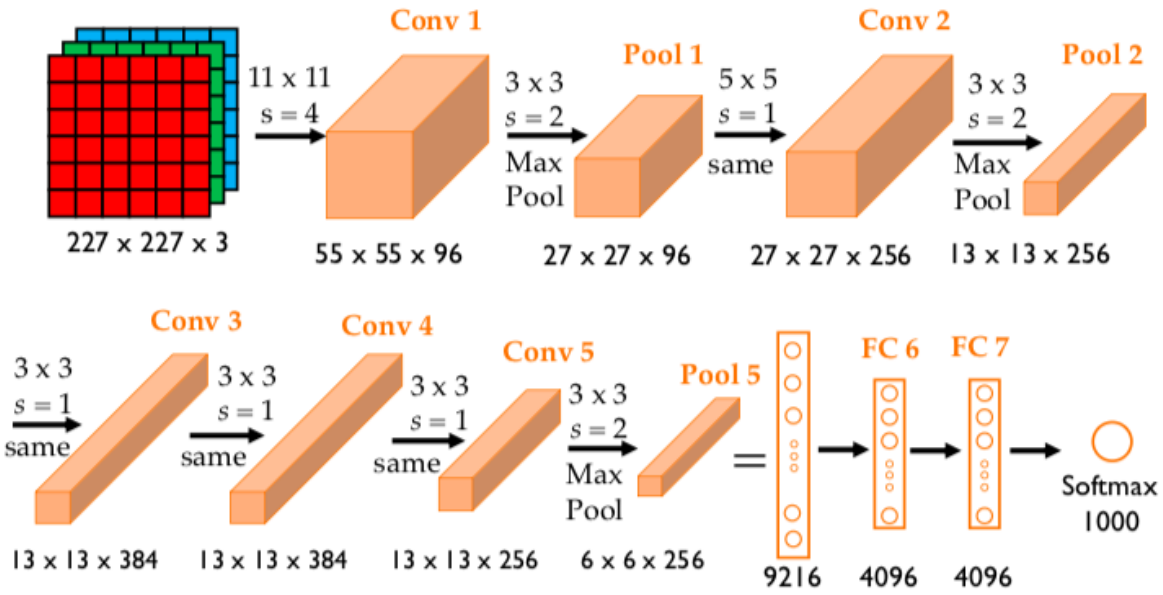


Figure 2.10: Architecture of the AlexNet deep learning network. Its objective is to classify an input image into one of 1000 classes. In this case, the final latent representation is the 1D tensor resultant of the *Pool 5* operation. *Source:* [21]

In the subsequent sections, a brief study of typical CNNs layers will be presented.

2.3.1 Convolutional Layer

The most important layer of CNNs is the convolutional layer. It applies convolutions operations using a set of kernels that travel across the data, extracting feature maps. In computer vision tasks, convolutional layers are used as a substitute for human vision, with the idea being that scanning an input with a kernel is an approximation of the processing done by the human visual system. Typically, several convolutional layers are stacked one after the other, forming a sequential processing pipeline. Early convolutional layers are responsible for extracting low level features, while later convolutional layers are responsible for extracting high level features. An example of this kind of structure is presented in Figure 2.11. Each convolutional layer has a set of kernels/filters with different weights, whose values determine what features the convolutional layer can detect. These weights are learned during model training using the backpropagation algorithm.

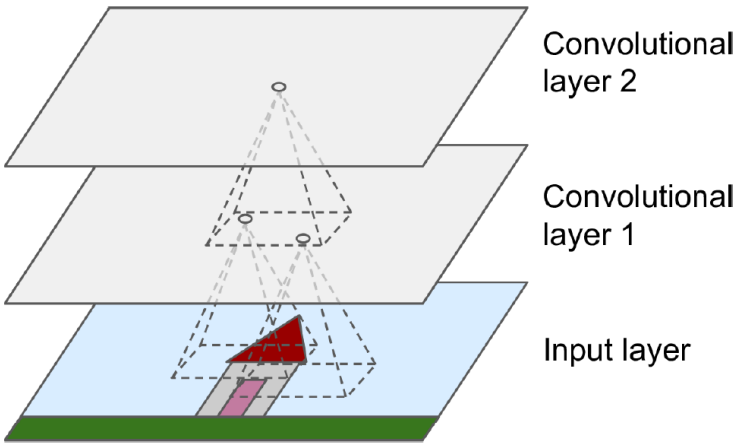
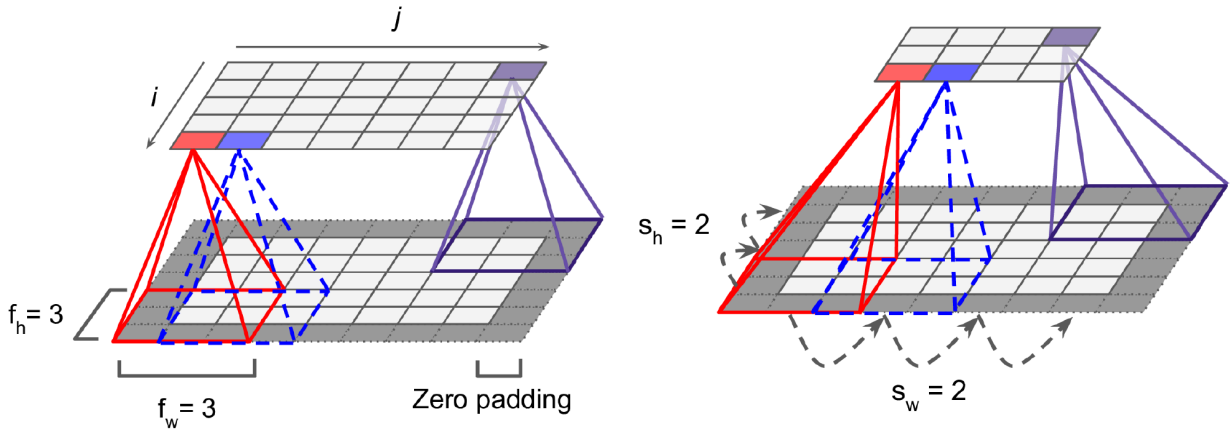


Figure 2.11: 2D example of stacked convolutional layers. *Source:* [14]

Convolutional layers also have parameters that modify their behaviour, such as the dimensions and stride of the kernel, type of padding, number of filters, etc. Considering a 2D case as an example, a convolutional layer with a kernel of size 3×3, zero padding of 1, stride 1 and 1 filter will give an output of the same dimensions as the input. This convolution operation is presented in Figure 2.12a. Changing the stride of the kernel to 2, as illustrated in Figure 2.12b, will reduce the height and width of the input by a factor of 2. This is particularly useful when compressing data.



(a) Stride 1.

(b) Stride 2.

Figure 2.12: 2D example of a convolution operation with a 3×3 kernel, zero padding of 1, 1 filter and different strides. *Source:* [14]

While the presented examples only have one feature map as an output, typical convolutional layers are composed of several stacked feature maps of the same dimensions (Figure 2.13), each produced by different kernels. Thus, the number of features that a convolutional layer can detect increases, increasing the capacity of the model to perform its designated task.

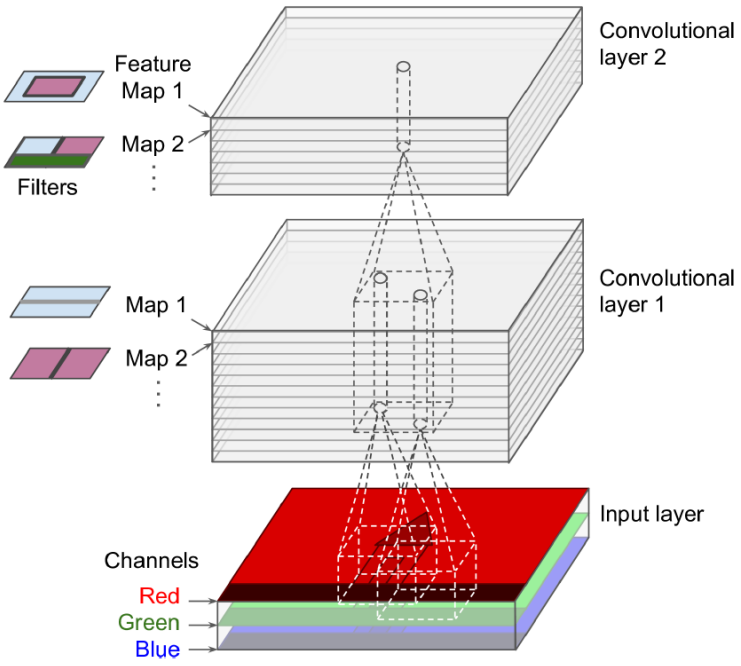


Figure 2.13: 2D example of a convolutional layer with multiple feature maps. *Source:* [14]

2.3.2 Pooling Layer

In conjunction with convolutional layers, pooling layers are frequently used as subsampling layers. Three of their main characteristics are the kernel size, stride and pooling operation. In Figure 2.14, a 2D pooling operation with a 2×2 kernel and stride 2 is presented. In this case, the output value of the pooling operation is the maximum value in the kernel window (max pooling).

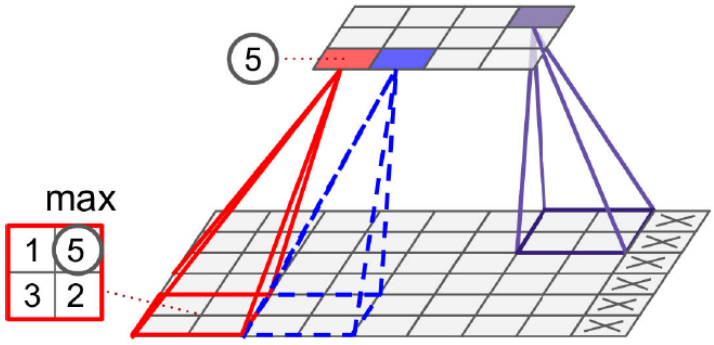


Figure 2.14: 2D example of a max pooling operation with a 2×2 kernel and stride 2. *Source:* [14]

This is an information destructive operation, i. e., three of the four values are discarded (in the given example), so pooling layers need to be placed with care when constructing a CNN. Furthermore, pooling can often be replaced by a larger stride step in the convolutional layers with no decrease in performance.

2.3.3 Activation Layer

Generally, after a convolution layer, the resulting tensor is passed through an activation layer. The purpose of this layer is to introduce non linearity into the model since most real world problems are non linear in nature. To achieve this, every value of the input is passed through a non linear function. The typical choice of function, presented in Figure 2.15, is the Rectified Linear Unit (ReLU). Other functions can also be used, such as the sigmoid or hyperbolic tangent, however, the best performing activation function is always extremely case sensitive.

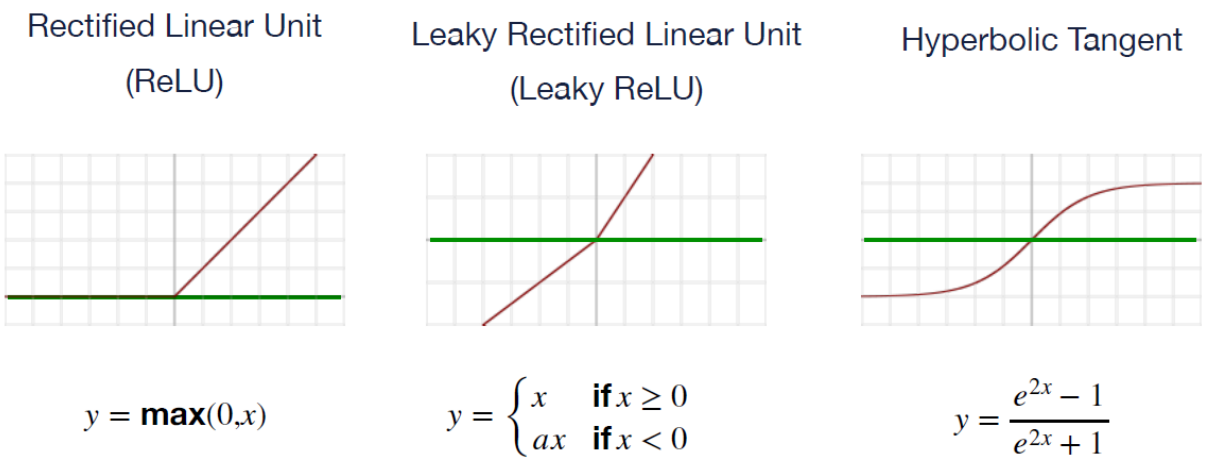


Figure 2.15: Graphs for the ReLU, leaky ReLU and hyperbolic tangent activation functions.

2.3.4 Normalization Layer

Considering the ReLU activation function, if most of the input values are negative, then the output tensor will be mostly comprised of zero values, losing almost all relevant feature information. To minimize this effect, a normalization layer can be used. This layer will try to normalize the input tensor to have approximately zero mean and standard deviation of one, as defined in eq. 2.18.

$$x' = \frac{x - \mu}{\sigma} \tag{2.18}$$

Since this operation will technically hide the original distribution of the data, which is undesirable, μ and σ are learned parameters. Consequently, the network will be able to decide the degree of the applied normalization. In Figure 2.16, an example of a normalization layer operation is presented.

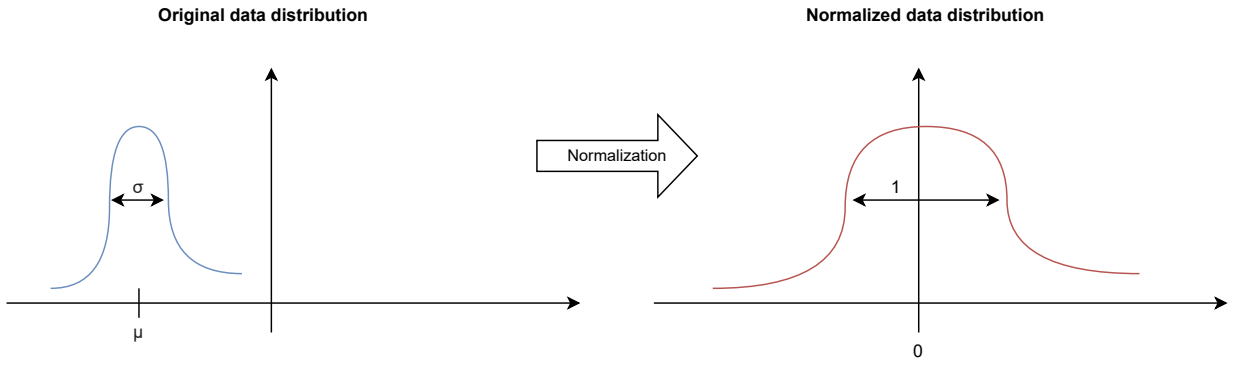


Figure 2.16: Normalization layer operation.

2.3.5 Fully Connected Layer

The fully connected layer is a type of layer often used as a classifier. Each neuron in a layer is connected to every other neuron in the previous layer through learned weights. In this context, a neuron can be an input, hidden or output neuron. An input neuron is a value of the 1D input, a hidden neuron is a value in an intermediate layer and an output neuron is a value in the output layer. Both the hidden and output neurons result from the operation

$$y = b + \sum_{i=1}^{N} w_i x_i \tag{2.19}$$

where N is the number of inputs, w_i is the weight attributed to input i, x_i is the input i and b is a bias. An example of a fully connected network is presented in Figure 2.17.

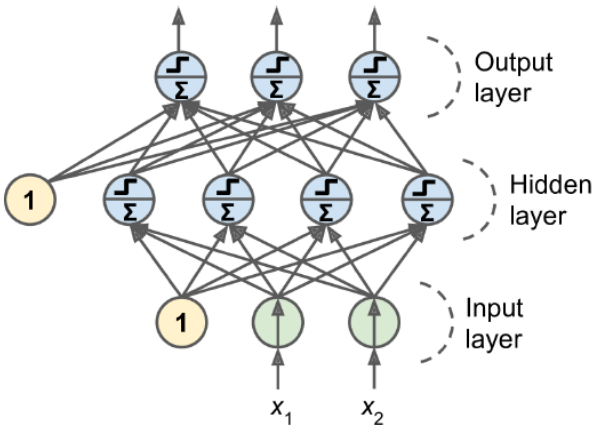


Figure 2.17: Example of a fully connected network. In this case, the bias is considered as a unit value concatenated to the input and a step activation function is used after every layer. *Source:* [14]

2.3.6 Dropout Layer

The final layer to be addressed is the dropout layer. Its main objective is to act as a regularization layer that prevents the overfitting of the DL model. It achieves this by randomly dropping values from its input. More specifically, each value of the input has a probability p, defined as a parameter, of being set to zero. An example of the dropout operation is presented in Figure 2.18.

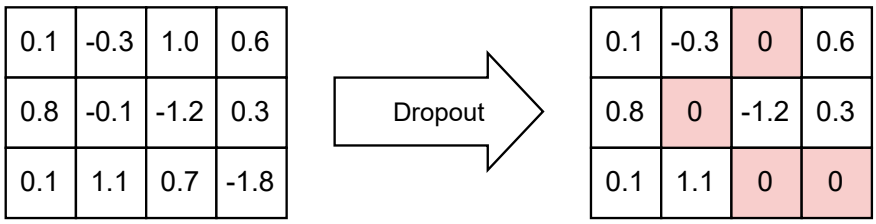


Figure 2.18: Example of a dropout layer on a 3×4 input. The red cells were dropped out.

2.4 Deep Learning Auto Encoder

In the context of DL-based compression, the auto encoder (AE) is the base DL architecture used in many applications. This type of DL architecture was designed to try to map the identity function given an input while simultaneously reducing its dimensionality. In Figure 2.19, the architecture of the AE is presented. Its Analysis Transform is the block that encodes the input into a smaller dimensional tensor, while the Synthesis Transform is the block that tries to decode the resulting latent representation of the encoding step. Each one of these transforms can be implemented using any desired architecture, for example, CNNs.

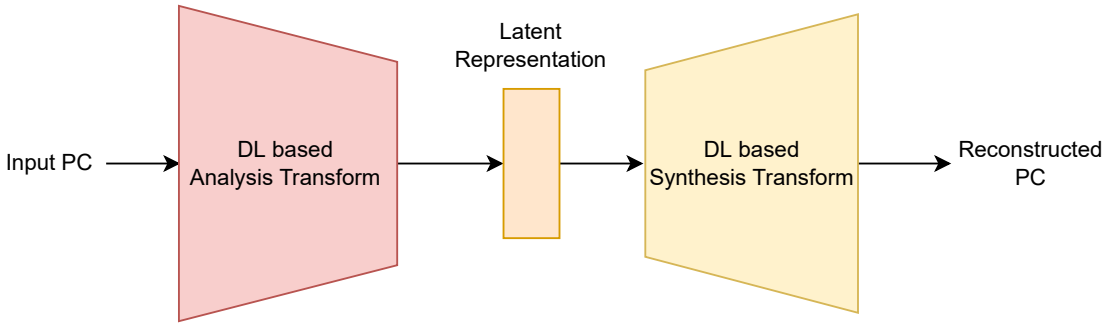


Figure 2.19: Standard DL-based AE architecture in PCC.

2.4.1 Deep Learning Variational Auto Encoder

While the AE provides a good way of reducing the dimensionality of an input, its reconstruction performance can vary widely. This is a commom phenomenon in AE architectures, as they tend to overfit to a certain type of data very easily. To combat this, the Variational Auto Encoder (VAE) was proposed [15]. The VAE is a probabilistic AE, more specifically, its output is partly generated

by probabilistic models. In Figure 2.20, a standard architecture for the VAE is presented.

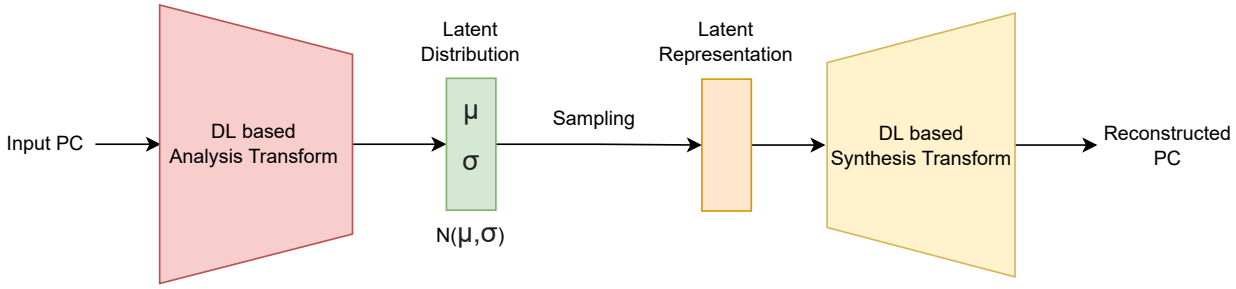


Figure 2.20: Standard DL-based VAE architecture in PCC.

Instead of the Analysis Transform encoding directly the latent representation, it encodes a latent distribution with mean μ and standard deviation σ. The latent representation is then randomly sampled from that latent distribution.

Ideally, the latent distribution should be definable as a known distribution, for example, as a gaussian distribution with mean μ and standard deviation σ. To achieve this, the loss used to train the VAE has two terms. The first is the same loss term as in the AE that measures the distortion between the input and output of the network. The second is a regularization term that pushes the latent distribution into the shape of a known target distribution. To that end, the Kullback-Leibler (KL) divergence [17] between the latent distribution and the target distribution is used. The KL divergence is a relative entropy measure, where the lower its output is, the more similar the distributions are to each other. In eq. 2.20, the standard loss of the VAE is presented.

$$loss = \|PC_{input} - PC_{reconstructed}\| + D_{KL}(N(\mu, \sigma), N(0, 1)) \tag{2.20}$$

2.5 Deep Learning Hyperparameters

When training a DL model, there are several variables that need to be defined before the training begins. These variables are denominated hyperparameters and their function is to control the training process.

2.5.1 Learning Rate

The learning rate is an extremely important hyperparameter. It controls the step size when updating the model's weights based on the error between the generated results and the target results. Defining a small learning rate could result in a long training process, with the network taking a long time to reach an optimal solution. On the contrary, defining a large learning rate could lead to drastic changes in the model's weights, causing the network to become unstable and unable to reach an optimal solution.

2.5.2 Batch Size

The batch size is the hyperparameter that defines the number of samples to be fed to the model in between each weight update. Usually, this hyperparameter is constrained by the amount of memory available in the system, as higher batch sizes tend to generate better results. However, the overall performance of the DL model does not hang heavily on the batch size, so the trade off between increasing the batch size and increasing the model's performance is not a key point in machine learning studies.

2.5.3 Number of Epochs

The final hyperparameter to explore is the number of epochs to execute during training. An epoch, in the context of DL, is defined as the number of steps that batch size of samples are fed to the model. For example, an epoch can be 20 batch sizes, i. e., 20 updates to the weights of the DL model. The DL training considers a maximum number of epochs to use during training. Typically, this number is very large, like one million epochs, however, trainings tend to not last that long because of early stopping mechanisms. Early stopping is the action of stopping training because the loss function stops improving after a certain number of epochs, indicating that the model has reached an optimal solution. This mechanism is used to avoid training unnecessarily because the model has already reached an convergence point in its loss value and to avoid overfitting.

3 Overview of DL-based Point Cloud Compression

3.1 Auto Encoder-based Point Cloud Compression

In the first DL-based PCC solutions, and considering the work [9] as an example, the AE was used with simple 3D CNNs as its analysis and synthesis transforms. The input PC was first divided into non-overlapping blocks, with each block encoded independently. Every block would produce a latent representation, similar to transform coefficients in traditional coding, that would subsequently go through a quantization step and end with entropy coding, a lossless data compression scheme. In this work, the entropy coding step uses arithmetic coding [31]. Finally, energy filtering was also applied to the latent representation to eliminate values with little energy. The compressed bitstream was comprised of all the latent representations as well as side information necessary to reverse the division of blocks of the input PC. The full pipeline can be observed in Figure 3.1.

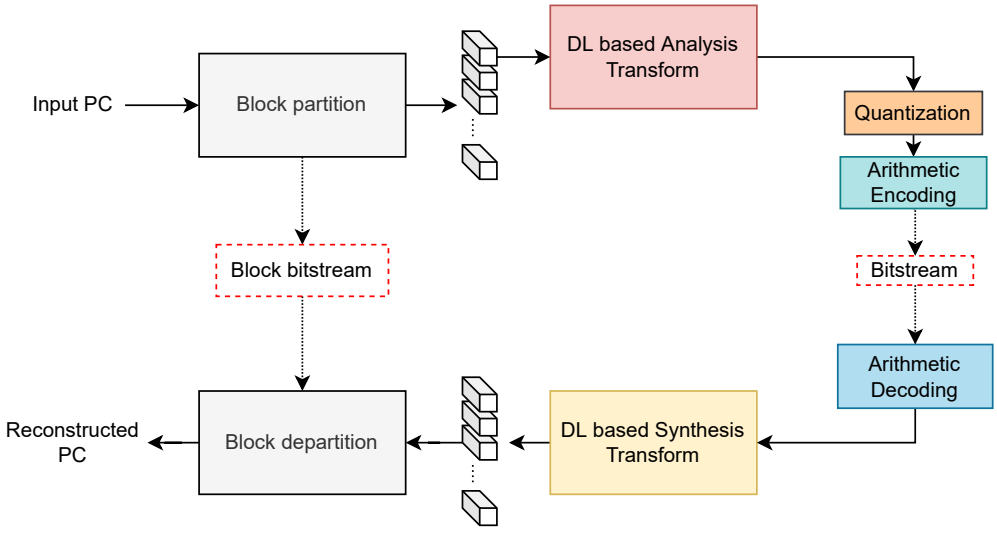


Figure 3.1: Basic pipeline of AE-based PCC.

This basic approach already gave good results in terms of RD compared with its G-PCC counterpart. However, the method showed there was still a lot of room for improvement, because the quantization and entropy coding steps were never considered during training. This resulted in the

decoder having to decode latent representations that it was not trained to decode, which produced a non-negligible increase in the reconstructed PC distortion. Furthermore, the only way for this approach to control the RD operation point was to choose a different quantization value during coding time by changing the quantization parameter of the quantization step. Lastly, the model itself had performance issues when it came to coding dense or sparse PCs, indicating that its generalization was not very good. For example, in [9], the model had worse performance when compressing sparse PCs for low bitrates compared to the Point Cloud Library [23] baseline.

3.2 Adaptive End-to-End Auto Encoder-based PCC

To solve the problems identified in the previous section, the previous approach was extended to an adaptive end-to-end DL-based model [20, 10]. This time, both the quantization and entropy coding steps were considered during training using a VAE to learn the context model of the entropy coding. This context model is called an hyper prior and is, essentially, the prior distributions for the hyper parameters of the entropy coding. Furthermore, the model considered a RD loss (eq. 3.1) using a lagrangian multiplier to choose a specific RD trade-off at training time.

$$loss = \lambda \cdot distortion + rate \tag{3.1}$$

Now, the compressed final bitstream is made up of three different bitstreams:

- The main bitstream comprised of the information of the encoded blocks.
- The side bitstream comprised of the information of the context model for the entropy coding.
- The block bitstream comprised of the information to reconstruct the PC based on the individualy encoded blocks.

The full pipeline for the adaptive AE-based PCC can viewed in Figure 3.2.

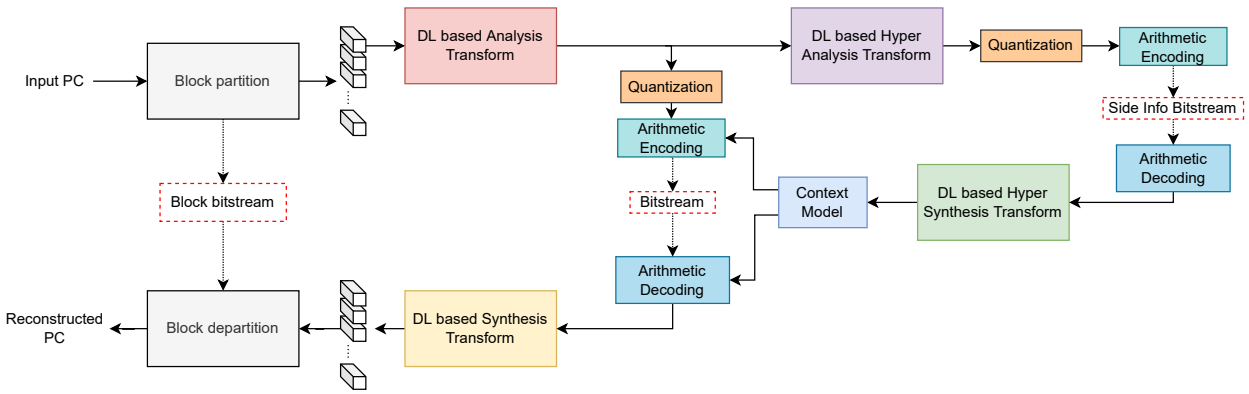


Figure 3.2: Basic pipeline of Adaptive AE-based PCC.

Finally, the adaptability of the model to different PCs was ensured by training multiple DL models, each one optimized for a different type of PC. In the inference phase, considering a specific PC, the best performing model was chosen (Figure 3.3).

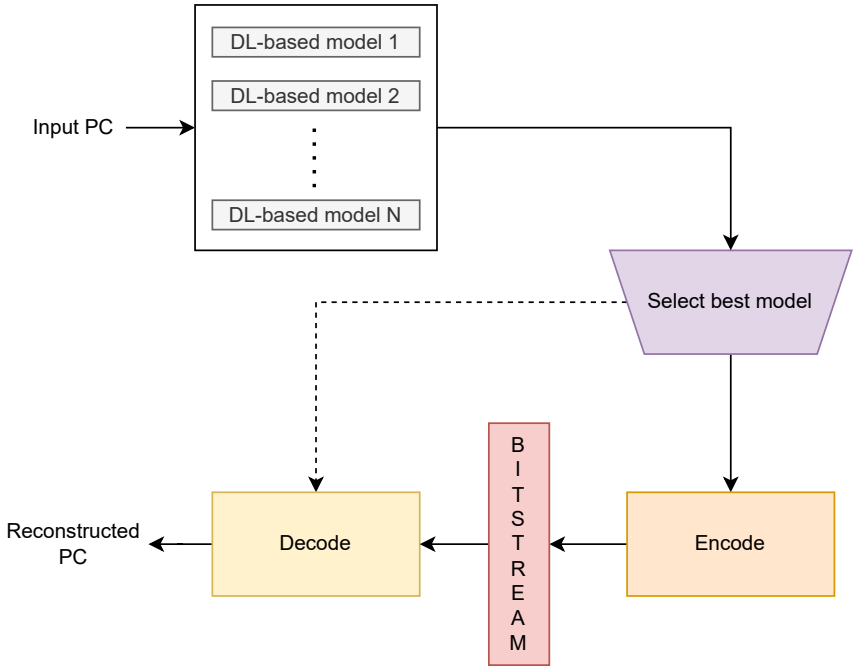


Figure 3.3: Adaptive DL-based PCC.

3.3 Neighborhood Adaptive Distortion Loss

One of the challenges in training a DL model is the choice of an appropriate loss function that can capture the nature of the current problem. With a DL PCC model that uses 3D blocks as its PC representation, the most popular choice of loss function for the distortion metric is the Focal Loss (FL) (eq. 3.2).

$$FL(v,u) = \begin{cases} -\alpha(1-v)^{\gamma}\log v, & u=1 \\ -(1-\alpha)v^{\gamma}\log(1-v), & u=0 \end{cases} \tag{3.2}$$

More specifically, each block is a sparse 3D binary occupancy matrix. Therefore, the task ends up being a binary classification between the classes zero and one. Because only $5-10\%$ of the 3D block voxels are filled, the FL is preferred to the weighted Binary Cross Entropy (wBCE) loss as it is better suited to situations where the data is very imbalanced. However, the FL fails to consider the context surrounding each voxel, unlike the typical distance-based PC metrics used for the final RD performance evaluation. To cope with this problem, the Neighborhood Adaptive Distortion Loss was proposed [12]. This new distortion loss takes into consideration the neighborhood of each voxel which more closely approaches the behavior of typical distance-based PC metrics. This solution

achieved a have more consistent RD performance across a range of different PCs for a single DL model.

3.4 Further RD Control via Implicit and Explicit Quantization

If there's a need to encode a PC for different RD trade-offs, then different models have to be trained and stored, controlling their RD point by choosing a different lagrangian multiplier at training time. In this case, implicit quantization is performed. More specifically, the quantization step at training time is achieved by applying an integer rounding operation. On the other hand, a different degree of quantization can be applied at coding time, allowing for a single DL model to have multiple RD points. This is called explicit quantization. By utilizing these two techniques, fewer DL models have to be trained to produce a wider range of RD trade-offs.

3.5 Scalable DL-based PCC

In compression, scalability refers to the capability to generate a bitstream that may be partially and progressively decoded to obtain a set of PC representations with increasing quality or resolution. In practice, the bitstream is typically organized into successively decodable layers, i.e. sub-streams, which cumulatively add quality or more points to the PC reconstructions (Figure 3.4).

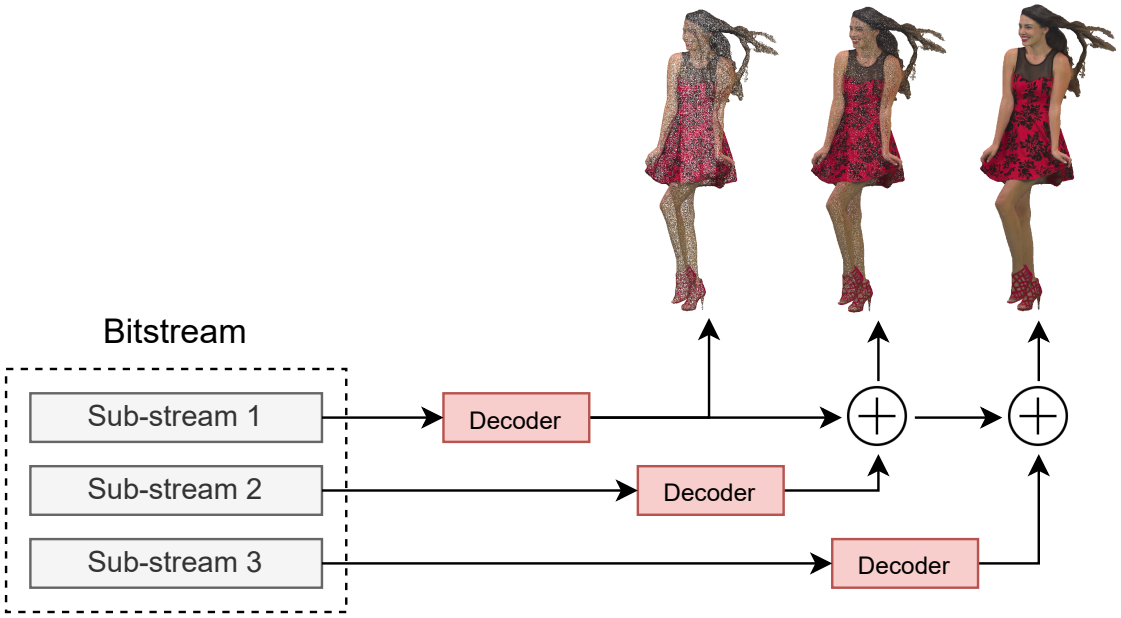


Figure 3.4: Example of progressively decoding a point cloud.

3.5.1 Resolution scalability

With a DL-based approach, the actual implementation of scalability in the model can vary. For example, two ways of achieving resolution scalability are interlaced sampling of the input PC [11]

or multiscale reconstruction [30].

With interlaced sampling, the input PC is divided into interlaced blocks (Figure 3.5), similar to interlaced image. Then, each group is individually encoded and added as a sub-stream. When decoding, each group is progressively added to the overall reconstruction of the PC.

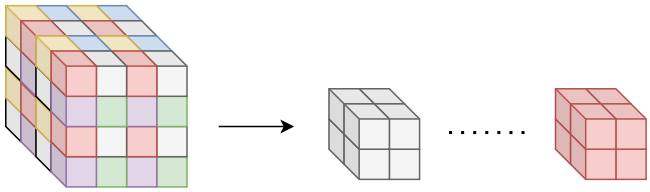


Figure 3.5: Example of interlaced sampling with a sampling factor of 2 in each 3D direction.

With a multiscale reconstruction approach, the model itself implicitly progressively reconstructs the PC. By utilizing sparse convolutions instead of the regular dense convolutions, each convolutional block of the encoder can downsample the PC into a desirable smaller resolution, with the decoder doing the opposite operation (Figure 3.6).

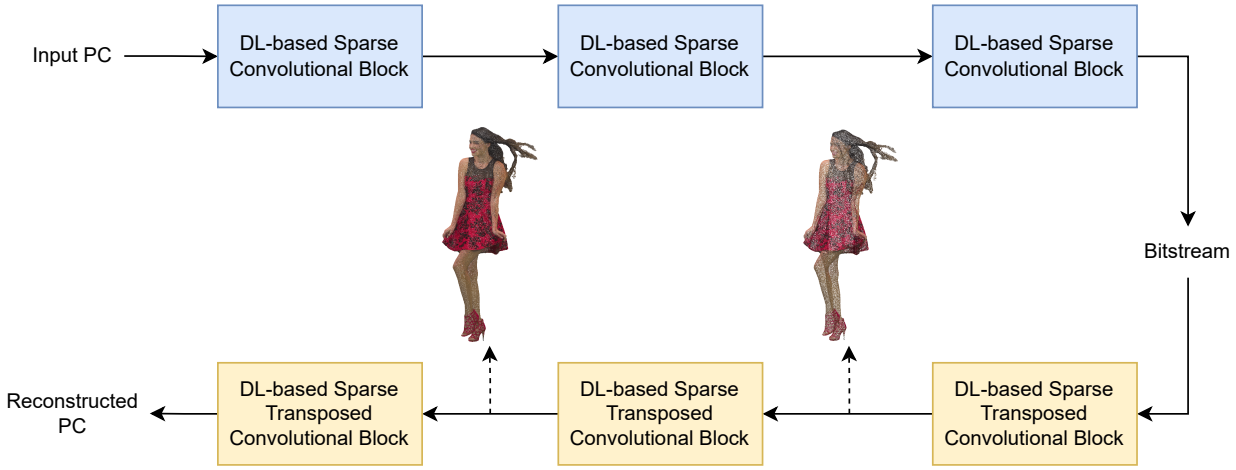


Figure 3.6: Example of a multiscale DL-based model. Each encoder block downsamples the input by a factor of N and each decoder block upsamples the input by a factor of N.

3.5.2 Quality scalability

To achieve quality scalability, a different approach to resolution scalability has to be taken. One such approach is the splitting of the latent feature map [13]. The main idea behind this quality scalable solution is the division of the latent representation into multiple non-overlapping latent sets, where each set is attributed to a scalable layer. Each layer is then independently quantized and entropy coded. In contrast to the resolution scalable solutions mentioned previously, this approach has the drawback of not being able to decode each sub-stream in an arbitrary order. Finally, on the decoder side, the latent representation is progressively completed with each latent set, zero padding missing sets (Figure 3.7).

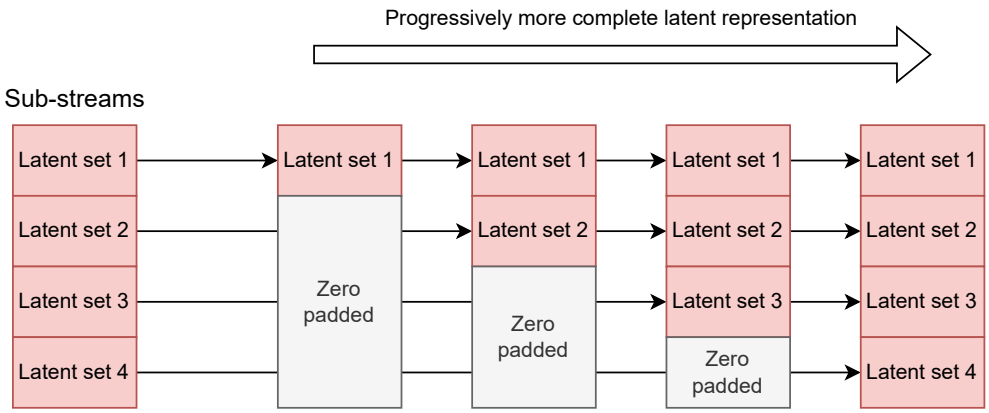


Figure 3.7: Example of latent set grouping on the decoder side. As more latent sets are available, the more complete the latent representation becomes and the higher quality the reconstructed PC is.

3.6 Point-based End-to-End Auto Encoder-based PCC

In all of the DL architectures so far, the input PC has been represented as a big 3D binary occupancy map. That is to say, its native representation of cartesian coordinates was transformed into a 3D counterpart. In turn, this resulted in a voxel-based PCC, however, approaches to PCC were also made maintaining the original representation of the PC, being mostly referred to as point-based PCC [33]. The overall idea of the compression model is the same, but some changes had to be made to accommodate the new representation:

1. Instead of a block partition, a subset of points uniformly distributed is found by utilizing an iterative farthest point sampling method. This kind of approach is based on a well-known PC architecture called PointNet [19].

2. The transform's operations were changed to use 2D CNNs or multi-layer perceptrons (MLPs). Additionally, for example, in the case of the PointNet architecture, a grouping operation of features is also performed to better encompass spatial information.

3. In the case of [33], the distortion metric of the loss function was changed to the Chamfer Distance (CD). For each input point, the closest point in the reconstructed PC is found and its squared euclidean distance is calculated. Then, all the squared distances are summed. Since this operation is not the same calculated with the original PC in relation to the reconstructed PC versus the reconstructed PC in relation to the original PC, the CD is the sum of both (eq. 3.3).

$$CD\left(PC_1, PC_2\right) = \sum_{u \in PC_1} \min_{v \in PC_2} \|u - v\|^2 + \sum_{v \in PC_2} \min_{u \in PC_1} \|u - v\|^2 \tag{3.3}$$

In subsequent sections, only voxel-based models and methods will be studied and presented.

4 Proposed End-to-End AE-based Geometry Point Cloud Coding

4.1 The Transformer

Convolutional networks have been the dominant architecture in computer vision tasks for several years, however, the Transformer [29] has shown to be a powerful alternative, most notably, in the Vision Transformer [4] work. At its core, the Transformer is an architecture designed to exceed in sequence transduction models, like natural language processing, that previously used recurrent, long short-term memory or gated recurrent neural networks. The reason for the study of Transformers in PCC is that the PC can be viewed as a sequence of patches (smaller PC blocks) if partitioned correctly. Therefore, a sequence can be extrapolated from the PCs and the Transformer architecture can be applied. The base architecture for the Transformer Encoder Layer (TEL) can be seen in Figure 4.1.

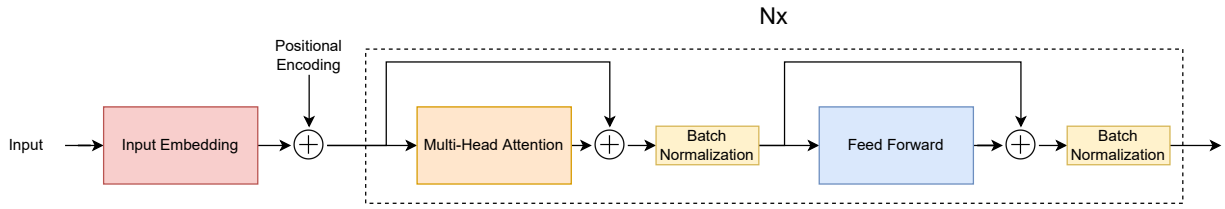


Figure 4.1: Base architecture of the Transformer Encoder Layer.

The Transformer has several steps in its pipeline. Starting with the input embedding step, its purpose is to transform the raw input into a 2D tensor of features where each set of features is one element of a sequence. After this step, we are left with an $S \times F$ tensor where S is the sequence length and F is the number of features of each element in the sequence. Next, positional encoding is added to the tensor so that the model has information about the relative position of the data points in their respective sequences. Any type of positional encoding can be applied in this step, for example, by adding values that will also be learned by the model. The next step considers a block of transformations that can be repeated N times. Besides additive residual and batch normalization operations, this block applies multi-head attention and feed forward operations. The multi-head

attention pipeline can be observed in Figure 4.2. The pipeline considers three sequences: query (Q), key (K) and value (V). In the case of self-attention, which is the only type of attention that will be used in this Dissertation, the query, key and value tensors will all be the same and equal to the input of the multi-head attention block. The transformations performed over the query, key and value tensors are performed h times in parallel and the results are concatenated into a long sequence. Multi-head attention also considers a scaled dot-product attention with the output matrix defined by eq. 4.1. An optional masking operation before the softmax can also be performed.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{S}})V \tag{4.1}$$

Finally, the feed forward block is just a generic DL block that contains any architecture to process the input tensor, such as CNNs or MLPs.

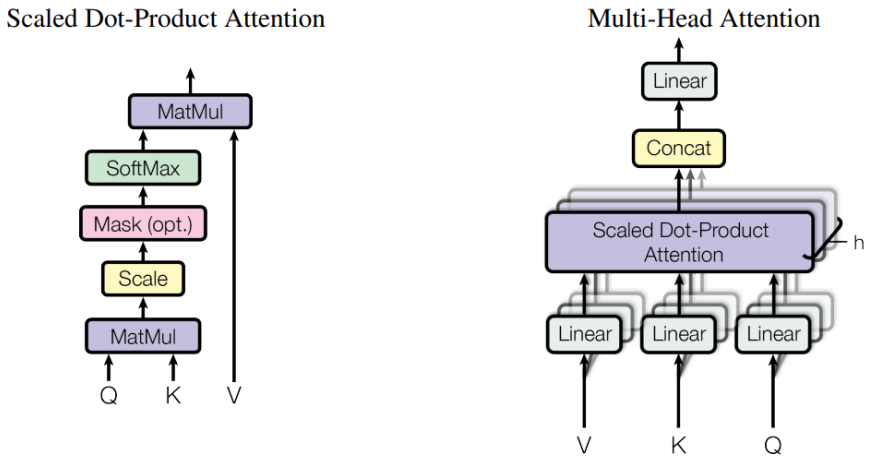


Figure 4.2: Multi-Head Attention pipeline. In this context, *linear* refers to a fully connected layer and *matmul* refers to matrix multiplication. *Source:* [29]

Even in the Vision Transformer, a DL approach to image classification, only the TEL is used, with the decoding (classification task) being performed by a MLP. This makes sense, because the images themselves do not have a proper sequence embedded in them. Therefore, it would not be appropriate to decode them as sequences. Likewise, for PCC, only the TEL is used in the DL-based encoder of the compression step [7].

One last important comment is the way the input embedding step is done. The most obvious way, in images or PCs, is to consider patches of a certain size, having a dedicated layer of the model learn the weight of every input value in a patch. However, like proposed in [28], at that point, the reason for the good performance of our model might not be the use of the Transformer architecture, but the use of patch-based representation of the inputs. Therefore, ConvMixers [28], a patch-based but purely convolutional neural layer, will also be considered for the encoder step of the compression.

4.2 DL-based Encoder for Point Cloud Compression

Adapting the TEL into a DL-based encoder for PCC can be done in many different ways. Following the architecture in Figure 3.2, what we're concerned about is defining the DL-based Analysis Transform. Therefore, four different encoders are proposed:

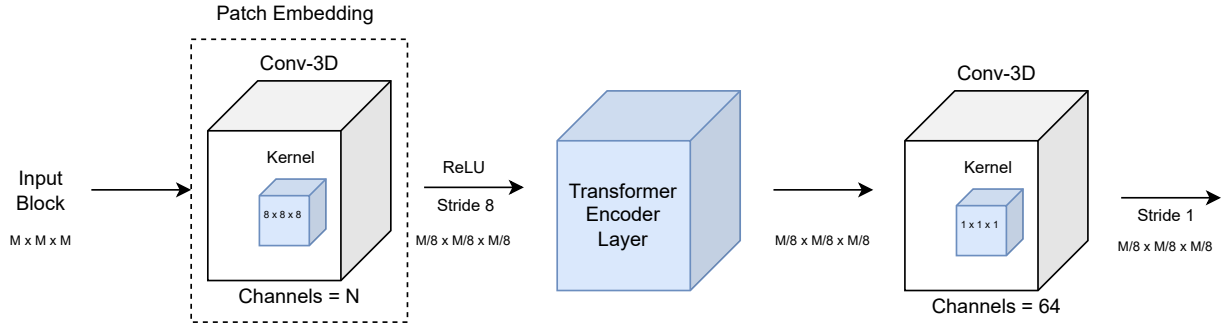
- Full Transformer Encoder (FTE), which can be seen in Figure 4.3a.
- Progressive Transformer Encoder (PTE), which can be seen in Figure 4.3b.
- Full ConvMixer Encoder (FCME), which can be seen in Figure 4.3c.
- Progressive ConvMixer Encoder (PCME), which can be seen in Figure 4.3d.

The objective of all the architectures in Figure 4.3 is to process an incoming $M \times M \times M$ tensor into an $M/8 \times M/8 \times M/8$ with H channels tensor. In the case of the *full* variant of the encoders, the patch embedding layer, that is to say, the first convolutional layer, generates an N channel tensor from the input block's patches. In this case, a patch is considered to be an $8 \times 8 \times 8$ block of the input block, however, the weights of each input voxel to that patch are learned by the model during training. The resulting tensor is then processed by the TEL or ConvMixer Encoder Layer (CMEL) and the final convolutional layer is only there to bring the number of channels back to the desired H. In this variant, the encoders combine the Transformer/ConvMixer architecture with patch-based inputs.

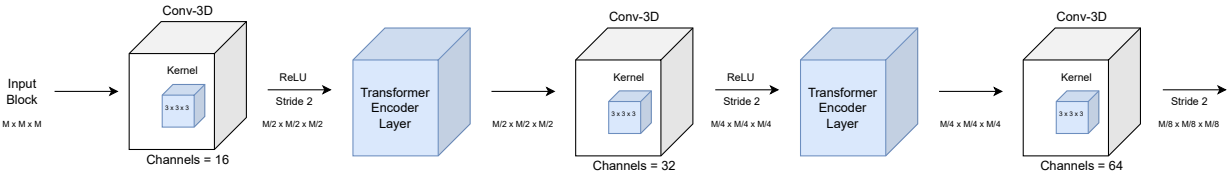
In the *progressive* variant of the encoders, the $M \times M \times M$ input tensor is progressively compressed by the convolutinal layers with stride 2, while also gaining more and more channels. Before a new convolutional layer, the tensor is processed in a TEL or CMEL.

One other detail is that there is no positional encoding taking place in the Transformer-based encoders. The reason is that, in the experiments conducted, adding positional encoding did not have a notable impact on final performance. In fact, usually, the simpler the encoder model, the better the model performed.

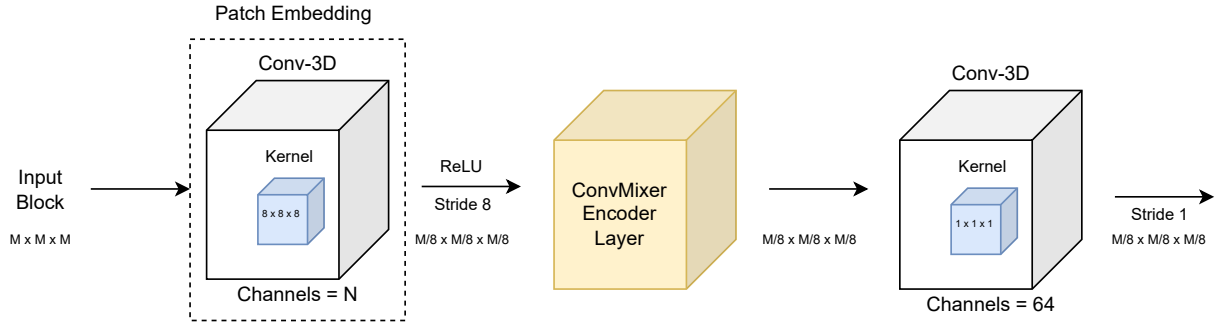
Lastly, both the TEL and the CMEL need to be defined (Figures 4.4 and 4.5, respectively). The final implementation is similar to their original works, but adapted to 3D input data. Of note, in the TEL, the multi-head attention block of the Transformer architecture was replaced with a 3D multi-head attention block which is just multi-head attention performed over 3 axis of data instead of 1 axis of data. In the experiments conducted, the use of batch normalization layers or dropout layers significantly reduced performance. Therefore, none of them were used. Given the complex nature of PCC, in some architectures, altering the original distribution of the data by means of normalization/dropout layers may just confuse the DL model. However, some works do use batch normalization/dropout, like [7].



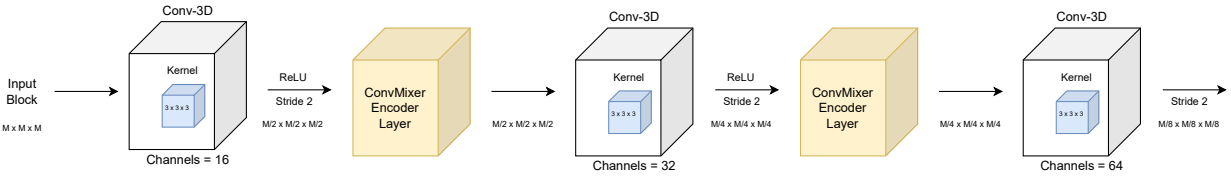
(a) FTE architecture.



(b) PTE architecture.



(c) FCME architecture.



(d) PCME architecture.

Figure 4.3: Proposed Transformer-based and ConvMixer-based PCC encoders.

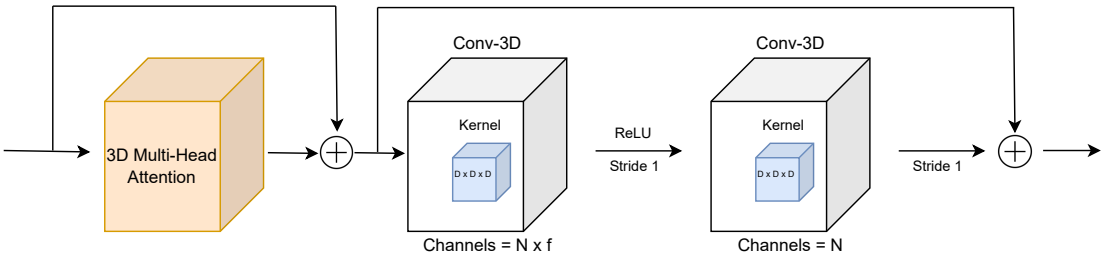


Figure 4.4: Final implementation of the Transformer Encoder Layer. In testing, $D = 3$ for the kernel size and $f = 4/3$ for the number of channels of the first convolutional layer were set empirically. In this iteration, the feed forward network is implemented as two convolutional layers.

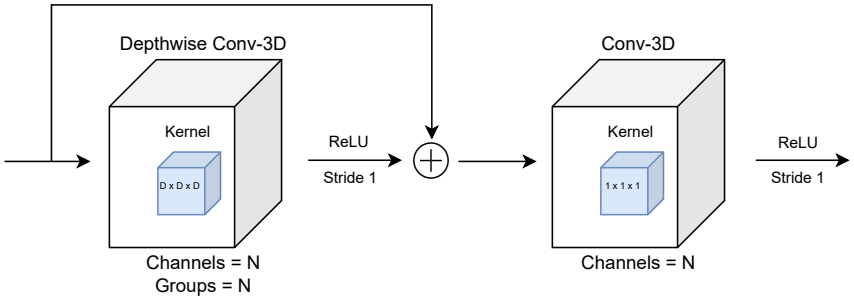


Figure 4.5: Final implementation of the ConvMixer Encoder Layer. In testing, $D = 3$ for the kernel size was set empirically.

4.3 DL-based Decoder for Point Cloud Compression

To complement the encoders previously presented, a decoder has to follow to allow the model to recover the original block. In this case, the decoder would be the DL-based Synthesis Transform. While other works use MLPs to decode the information generated by the Transformer, the computational complexity and the 3D nature of the problem make them unsuitable for that task. Thus, the 3D Synthesis Transform CNN from [20] was used (Figure 4.6) since it already proved to be a good DL decoder architecture for PCC in its work. With more time, other DL architectures could be studied for the use in PCC.

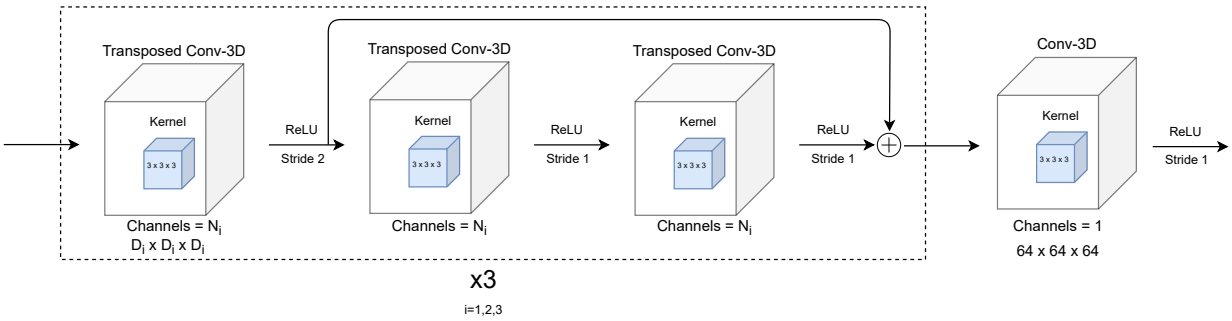


Figure 4.6: Synthesis Transform of the DL-based PCC model. Both the number of channels and dimensions of the blocks progressively increase with $D_i = \{8, 16, 32\}$ and $N_i = \{16, 32, 64\}$.

4.4 DL-based Hyper Transform for Point Cloud Compression

To improve the flexibility and performance of the entropy coding step in the PCC pipeline, a context model is learned by the DL model using a VAE. This context model is called an hyper prior and is, essentially, the prior distributions for the hyper parameters of the entropy coding. Since the context model will also need to be saved alongside the information from the 3D block, it will be compressed by the model by a hyper analysis transform and decompressed by a hyper synthesis transform. The architectures used for these transforms are presented below.

4.4.1 Hyper Analysis Transform

In Figure 3.2, the hyper analysis transform is responsible for compressing the context model to be used in the entropy coding step of the block compression. To achieve this, the 3D Hyper Analysis Transform CNN from [20] was used (Figure 4.7) since it already proved to be a good DL hyper encoder architecture for PCC in its work. This architecture is considerably simpler than the analysis and synthesis transforms because the considered context model is a much less complex structure than the input PC block.

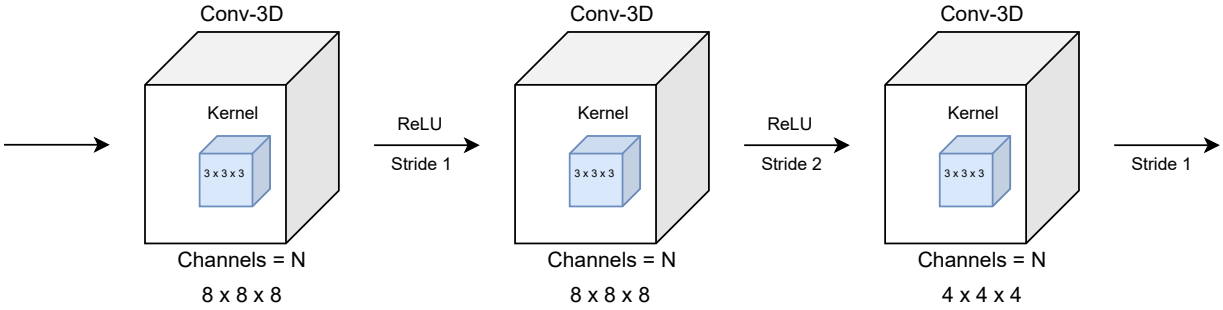


Figure 4.7: Hyper Analysis Transform of the DL-based PCC model.

4.4.2 Hyper Synthesis Transform

In Figure 3.2, the hyper synthesis transform is responsible for decompressing the context model to be used in the entropy coding step of the block compression. To achieve this, the 3D Hyper Synthesis Transform CNN from [20] was used (Figure 4.8) since it already proved to be a good DL hyper decoder architecture for PCC in its work. This architecture is considerably simpler than the analysis and synthesis transforms because the considered context model is a much less complex structure than the input PC block.

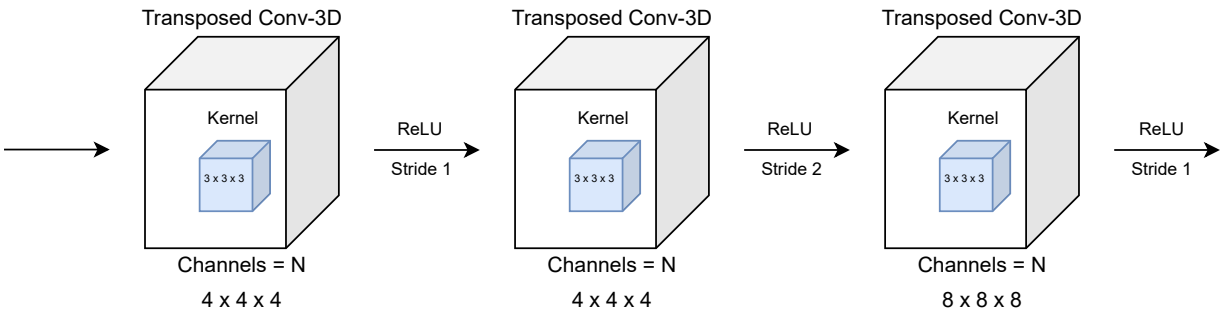


Figure 4.8: Hyper Synthesis Transform of the DL-based PCC model.

4.5 DL Model Training and Testing

Now that the general model pipeline has been designed, it's necessary to indicate, specifically, the hyper parameters and methods used. The subsequent sections will present the:

- Block partition method;
- Training dataset;
- Loss function and its parameters;
- Experimental conditions (testing dataset, performance metrics and benchmarking);
- Experimental results.

4.5.1 Block partition method

Block partitioning can be done in a variety of ways. However, like the standard G-PCC, the octree partition method [24] was adopted. This method involves a recursive algorithm that subpartitions 3D space. To achieve this partition, all the input point clouds dimensions have to be normalized, more specifically, they have to be voxelized. A voxelization with a depth of d bits transforms the

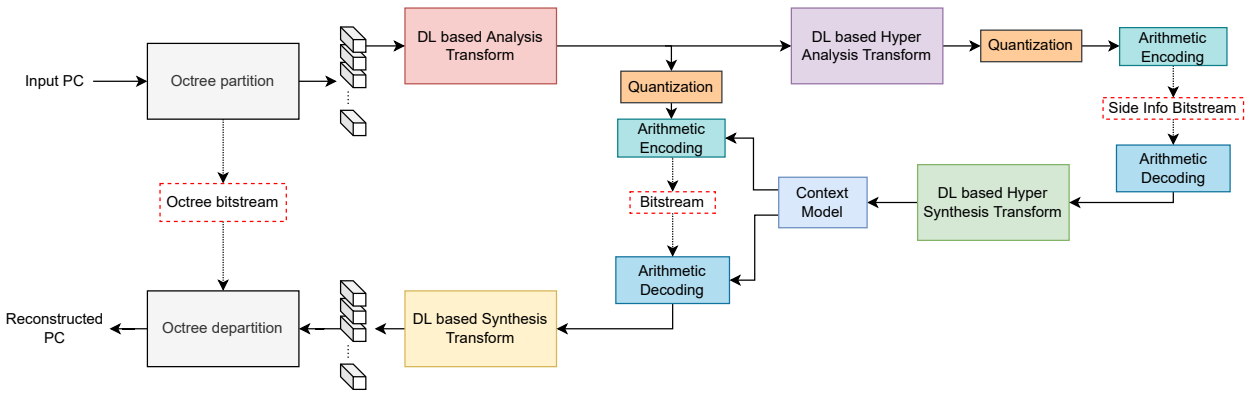


Figure 4.9: Full final pipeline of the experimental PCC model.

interval of the (x, y, z) coordinates of an input PC into the interval $[0, 2^d - 1]$. The actual precision of the voxelization will be mentioned later with the presentation of the PCs used. Lastly, the size of the blocks was set empirically to 64 in all the experiments. The full final pipeline of the model can be seen in Figure 4.9.

4.5.2 Training dataset

For the training dataset, the ModelNet40 dataset [32] was used, a mesh dataset of 40 different classes. To transform the dataset into usable 3D blocks, first, a random sampling was done of the meshes, followed by a voxelization with a depth of 9 bits. Then, the 200 largest PCs in terms of point counts were partitioned using an octree of depth 3, leading to $64 \times 64 \times 64$ 3D blocks. Since, by nature, most of those blocks have only $5-10\%$ filled voxels, only the 4000 blocks with the largest number of points were used for training.

4.5.3 Loss function

As stated previously, the loss function in eq. 3.1 is a combination of a distortion metric and a coding rate with a lagrangian multiplier to control the RD trade-off. The implementation is the same as in work [20], and is explained below.

Since at the heart of the compression problem there is a heavily imbalanced binary classification problem, the FL was used

$$FL(v, u) = \begin{cases} -\alpha(1-v)^\gamma \log v, & u = 1 \\ -(1-\alpha)v^\gamma \log(1-v), & u = 0 \end{cases} \tag{4.2}$$

where u is the original voxel value and v is the reconstructed voxel value. The FL has two hyper parameters that need to be specified before training. The γ parameter controls the relevance of voxels that are hard to classify by downplaying the loss impact of the easy ones. The higher the parameter is, the more relevant the hard voxels are. The $\alpha \in [0, 1]$ parameter controls the importance of filled voxels. The higher the parameter is, the more important the filled voxels are. In the experiments conducted, the two parameters were set empirically to $\alpha = 0.75$ and $\gamma = 2$.

As for the coding rate, more specifically, the expected number of bits per input point, an approximation was used. This is necessary because the actual entropy coding is not done at training time due to differentiability issues. Therefore, the training coding rate is calculated using the quantized latent representation, defined as qlr, and the quantized hyper latent representation, defined as $qhlr$.

$$coding\ rate = \frac{\sum \ln(p_{qlr}(qlr)) + \sum \ln(p_{qhlr}(qhlr))}{\ln\left(\frac{1}{2}\right) \times N} \tag{4.3}$$

In eq. 4.3, $p_{qlr}(qlr)$ and $p_{qhlr}(qhlr)$ represent the probability distributions of qlr and $qhlr$, respectively, and N corresponds to the number of the block's input points.

Table 4.1: Model hyper parameters: learning rate (LR), batchsize (B), kernel size (KS), f, N and λ.

LR	**B**	**KS**	f	N	λ
10^{-4}	32	$3 \times 3 \times 3$	$\frac{4}{3}$	$\{64, 128, 256, 512\}$	$\{5e^{-6}, 2e^{-5}, 5e^{-5}, 1e^{-4}, 3e^{-4}\}$

In terms of model-related hyper parameters (Table 4.1), a kernel size of $3 \times 3 \times 3$ for the TELs and CMELs was used and, in the case of the TELs, a factor f of 4/3 was used. In this context, f is a factor that multiplies the number of channels in the first convolutional layer of the TELs (see the TEL in Figure 4.4). These values were chosen empirically. In the case of the *full* variant encoders, when it came to studying the impact of N (number of channels in the first convolutional layer of its encoder) on the encoding performance, four values were used: 64, 128, 256 and 512.

Finally, as for the lagrangian multiplier λ in the loss function that controls the RD point, five values were used: $3e^{-4}$, $1e^{-4}$, $5e^{-5}$, $2e^{-5}$ and $5e^{-6}$. In the testing phase, the five values selected lead to five different values of RD performance, one for each DL model.

All the models were implemented and trained in Tensorflow [1], version 2.5.2. The Adam optimizer was used to minimize the training loss with a learning rate of 10^{-4} and a batch size of 32 blocks. Regarding batch size, experiments were conducted to infer its best value. To that end, trainings were conducted with batch size $B \in \{8, 16, 32, 64, 128\}$. As batch size increased, so did the quality performance of the model, however, the size of the compressed bitstream would also increase. A batch size of 16 or 32 offered good model performance without a notable increase in bitstream size (4% lower D1 PSNR at 54% decrease in bitstream size compared with a batchsize of 128). In contrast, a batch size of 8 had worse performance (6% lower D1 PSNR at 6% increase in bitstream size compared with a batchsize of 32). Since the 32 batch size model was marginally better than the 16 batch size one, a batch size of 32 was chosen.

4.6 Experimental conditions

4.6.1 Testing dataset

For testing, some PCs from the 8iVFB V2 [5] dataset was used, namely, the *Long Dress 1300*, *Loot 1200*, *Red and Black 1550* and *Soldier 690* PCs. In addition, three more popular PCs were used, namely, *Queen*, *Statue Klimt* and *House without Roof*. All the PCs were voxelized to have a depth of 10 bits, with an octree division of depth 4 to obtain $64 \times 64 \times 64$ 3D blocks. Figure 4.10 shows 2D views of these PCs and Table 4.2 shows the PCs characteristics.

(a) Long Dress. (b) Loot. (c) Red and Black. (d) Soldier.

(e) Queen. (f) Statue Klimt. (g) House without Roof.

Figure 4.10: Test point clouds.

Table 4.2: Test PCs characteristics. The sparsity of the PCs was measured by calculating, for all points in a PC, the average of the euclidean distance between the point and their 20 closest neighbors and averaging these distances over all points of the PC.

Point Cloud	**Number of points**	**Attributes**	**Normals**	**Sparsity**
Long Dress	857966	RGB color	yes	1.73
Loot	805285	RGB color	yes	1.73
Red and Black	757691	RGB color	yes	1.73
Soldier	1089091	RGB color	yes	1.73
Queen	1000993	RGB color	yes	1.63
Statue Klimt	482955	RGB color	yes	2.38
House without Roof	1722782	RGB color	yes	1.78

4.6.2 Performance Metrics

To compare the RD performance of the various DL PCC models and the relevant baselines, both the rate and the geometry distortion were measured and represented in a RD graph. The rate was calculated as the number of bits on disk needed to store all the necessary bitstreams to reconstruct the PC divided by the total number of points in the original PC. This metric is called bits-per-point (bpp), similar to bits-per-pixel in images.

As for the geometry distortion, the point-to-point and point-to-plane distances described in sections 2.2.1 and 2.2.2 were used, namely the D1 PSNR and D2 PSNR metrics. Both of them are asymmetric metrics, more specifically, calculating the metric with PC A in relation to PC B is different to calculating the metric with PC B in relation to PC A. Usually, the worst value is chosen as the final value, therefore, that methodology was also adopted for this study. To calculate these metrics, the software *mpeg-pcc-dmetric* [8], version 0.13.4, was used.

Finally, for all the encodings done, the BDs for the rate and PSNR were calculated.

4.6.3 Benchmarking

For benchmarking purposes, the G-PCC (in octree mode) and V-PCC with the VVC video encoder (in intra mode with only one frame) were used as a baseline, compressing the relevant PC with various RD trade-offs. Furthermore, the PCC GEO V2 work [20], another DL-based PCC model, was also included in the comparisons, since it was the encoder on which this study was based.

One last thing to note is that, like it was proposed in PCC GEO V2, an optimal threshold value is used in the compression of the PCs. In the inference phase, when classifying voxels as occupied or not, a fixed 0.5 threshold can be used for values in the interval $[0, 1]$. However, the PCC GEO V2 work showed that, by testing different thresholds, an optimal threshold can be found which

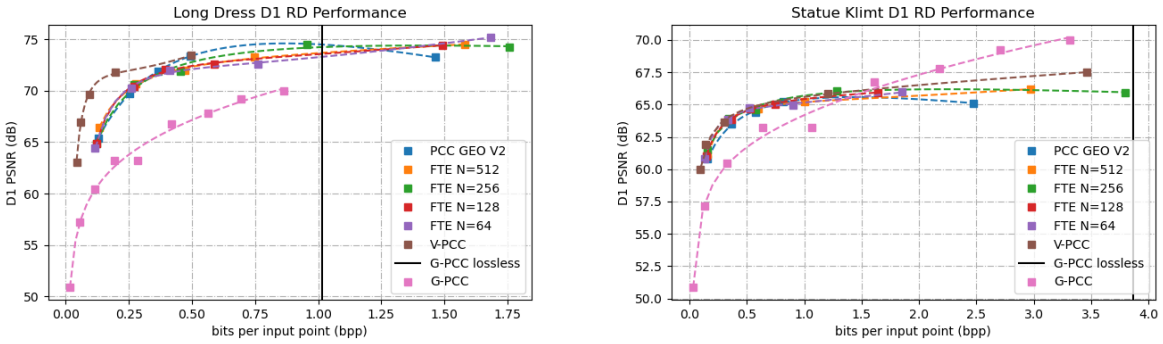
significantly improves the final quality of the reconstruction. This comes at the cost of saving all the necessary thresholds for all the PC's blocks in a separate bitstream. However, the rate increase introduced by storing these thresholds is negligible compared to the quality increase. In practice, the interval $[0, 1]$ is separated into 255 thresholds from 0 to 1. The PC block is then compressed and decompressed using all 256 thresholds and the D1 PSNR quality metric is recorded for each threshold, comparing reconstructed blocks with their original counterparts. The threshold that obtains the higher D1 PSNR metric is the one that is saved in the bitstream to later reconstruct the block.

4.7 Experimental results

RD performance is the most important criterion when measuring compression efficiency. Therefore, for each encoder proposed, the RD graphs for the D1 and D2 metrics and a table with the BD will be presented, comparing their performance with the G-PCC, V-PCC and PCC GEO V2 models. At the end, a table listing all the BDs will be presented to facilitate results analysis. Because of space constraints, only results of two PCs from the testing dataset will be presented in the following sections, but the full results can be viewed in Annex A. These PCs were chosen based on their sparsity score to represent the dense PCs and the sparse PCs. To represent the dense PCs, the Long Dress was chosen because it had a sparsity of 1.73, indicating a dense PC (sparsity less than 2). To represent the sparse PCs, the Statue Klimt was chosen because it had a sparsity of 2.38, indicating a sparse PC (sparsity greater than 2). The threshold of 2 to distinguish dense and sparse PCs was set arbitrarily as a threshold that made intuitive sense.

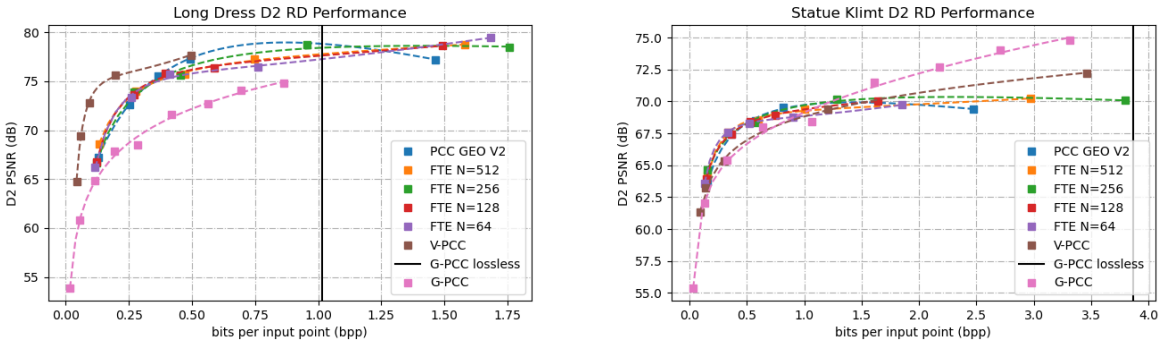
4.7.1 Full Transformer Encoder

In Figures 4.11 and 4.12 and Tables 4.3, 4.4 and 4.5, the RD performance for the FTE can be observed. In the *full* variant of the encoders, experiments were conducted with $N = \{64, 128, 256, 512\}$ feature channel models to study the effect of N on performance.



(a) Long Dress. (b) Statue Klimt.

Figure 4.11: D1 PSNR RD performance graphs for the FTE.



(a) Long Dress.

(b) Statue Klimt.

Figure 4.12: D2 PSNR RD performance graphs for the FTE.

Table 4.3: BD metrics for the FTE with N feature map channels compared to the G-PCC encoder. The BD-rate is in percentage and the BD-PSNR is in dB. The best value is in bold.

			N			
Point Cloud	**Metric**	**BD**	***512***	***256***	***128***	***64***
Long Dress	D1	rate	-250.36	-181.14	-229.17	**-267.91**
		PSNR	**5.48**	5.43	5.43	5.27
	D2	rate	**-129.21**	-87.66	-109.64	-120.64
		PSNR	**3.96**	3.83	3.85	3.63
Statue Klimt	D1	rate	-106.11	-110.38	-127.03	**-139.80**
		PSNR	1.39	1.43	**2.38**	2.33
	D2	rate	-45.06	-44.31	-50.37	**-59.61**
		PSNR	0.31	0.31	**1.08**	0.96

As is the case with other DL-based PCC solutions, the FTE significantly outperforms the G-PCC encoder. More so in the case of dense PCs (*Long Dress*, *Loot*, *Red and Black*, *Soldier* and *Queen*) than with the sparse and noisy PCs (*Statue Klimt* and *House without Roof*). In fact, as one can observe in the Figures 4.11b and 4.12b, the DL-based FTE and PCC GEO V2 have clearly inferior performance compared to the G-PCC encoder at high bitrates for sparse PCs. This can be caused by a multitude of reasons, but two of the strongest ones are the feature extraction of the encoder being more suited for dense PCs and the fact that the training dataset ModelNet40 is comprised of almost exclusively dense PCs. Therefore, it's no surprise that the final model has a bias towards dense PCs which are then better compressed. One way to mitigate this problem might be to consider a bigger patch size when doing the patch embedding, thus increasing the spatial awareness for each patch.

In terms of the impact of the capacity of the model with N channels, once more, the results are mixed between dense PCs and sparse PCs. For dense PCs, more channels have a marginal increase

in performance. For sparse PCs, fewer channels see an increase in performance. An intuitive reason for this change is that, with fewer channels, the model has less capacity to overfit to dense PCs, leading to a more general DL-based encoder.

Table 4.4: BD metrics for the FTE with N feature map channels compared to the V-PCC encoder. The BD-rate is in percentage and the BD-PSNR is in dB. The best value is in bold.

			N			
Point Cloud	**Metric**	**BD**	***512***	***256***	***128***	***64***
Long Dress	D1	rate	**54.28**	57.22	55.56	54.52
		PSNR	**-2.14**	-2.57	-2.36	-2.50
	D2	rate	**56.13**	58.57	57.18	56.45
		PSNR	**-2.93**	-3.49	-3.22	-3.39
Statue Klimt	D1	rate	21.98	17.87	19.07	**15.64**
		PSNR	-0.46	-0.30	-0.29	**-0.27**
	D2	rate	-42.49	-44.28	-44.96	**-50.54**
		PSNR	0.60	0.64	**0.98**	0.90

In Table 4.4, contrasting with the G-PCC encoder, the V-PCC encoder drastically outperforms the DL-based solutions. However, this increase in performance is only felt when compressing denser PCs. As one can observe in the Figures 4.11b and 4.12b, the V-PCC encoder has similar performance to the DL-based encoders. Furthermore, as the rate increases, the lesser the gain in performance compared to the other encoders. In fact, the V-PCC encoder is, by far, the most complex encoder tested. Fine-tuning its multitude of parameters to a certain PC can yield noticeable gains in performance, however, that fine-tuning is non-intuitive, requiring a lot of trial and error. Therefore, similar parameters were used in all the test PCs. The V-PCC parameters used in all the encodings were the default common test conditions provided by the V-PCC github repository[1].

Finally, in terms of the impact of the capacity of the model with N channels, the conclusions are similar to those of the G-PCC encoder. More channels benefit denser PCs and fewer channels usually benefit sparser PCs.

[1] `https://github.com/MPEGGroup/mpeg-pcc-tmc2.git`

Table 4.5: BD metrics for the FTE with N feature map channels compared to the PCC GEO V2 encoder. The BD-rate is in percentage and the BD-PSNR is in dB. The best value is in bold.

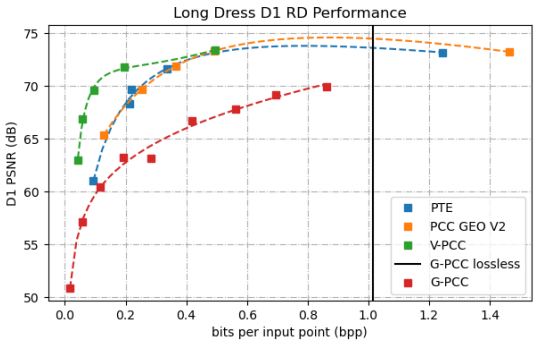
			N			
Point Cloud	**Metric**	**BD**	***512***	***256***	***128***	***64***
Long Dress	D1	rate	-1.02	-0.84	-1.95	**-3.57**
		PSNR	-0.22	**-0.14**	-0.24	-0.40
	D2	rate	-2.43	-1.35	-2.82	**-3.24**
		PSNR	-0.24	**-0.17**	-0.28	-0.48
Statue Klimt	D1	rate	-27.88	-29.13	-25.97	**-34.70**
		PSNR	0.27	**0.49**	0.35	0.38
	D2	rate	-10.92	-11.23	-9.24	**-12.83**
		PSNR	0.11	**0.27**	0.07	0.04

Observing Table 4.5, the PCC GEO V2 encoder marginally outperforms the FTE in the case of dense PCs, most of the time, with a reduction in rate. However, with sparse PCs, the FTE comes out on top, even with a reduction in rate. In fact, the PCC GEO V2 encoder has a bias towards better compressing dense PCs, as will be shown in subsequent analysis. It does not mean that it's overall better or worse than, for example, the FTE. It just means that, for general purpose use PCC, the FTE, for example, might be more appealing in more applications.

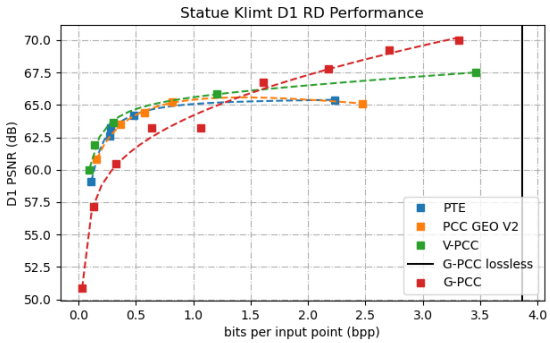
Concentrating now on the influence of N in the RD performance, the results in Table 4.5 do not follow the same trend as when comparing with the G-PCC and V-PCC encoders. In terms of the BD-PSNR gain, the 256 channel model dominates the results in all PCs (see Annex A Table A.3 for more results). In terms of BD-rate, the 64 channel model outperforms the other models. Given the *black box* nature of DL, the explanation for these results can be hard to grasp. One likely explanation might just be that the number of channels in question were accidental optimal parameters considering all the other test conditions.

4.7.2 Progressive Transformer Encoder

In Figures 4.13 and 4.14 and Table 4.6, the RD performance for the PTE can be observed.

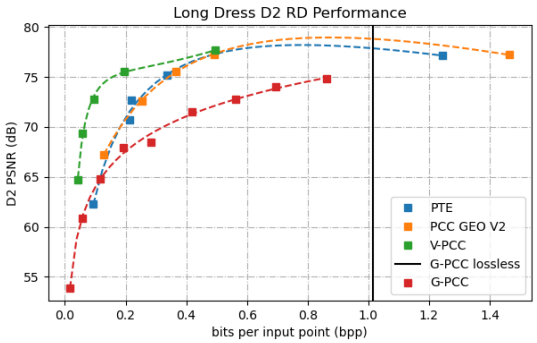


(a) Long Dress.

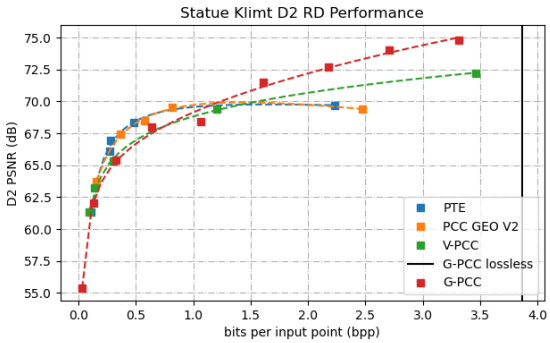


(b) Statue Klimt.

Figure 4.13: D1 PSNR RD performance graphs for the PTE.



(a) Long Dress.



(b) Statue Klimt.

Figure 4.14: D2 PSNR RD performance graphs for the PTE.

Table 4.6: BD metrics for the PTE compared to the G-PCC, V-PCC and PCC GEO V2 encoders. The BD-rate is in percentage and the BD-PSNR is in dB.

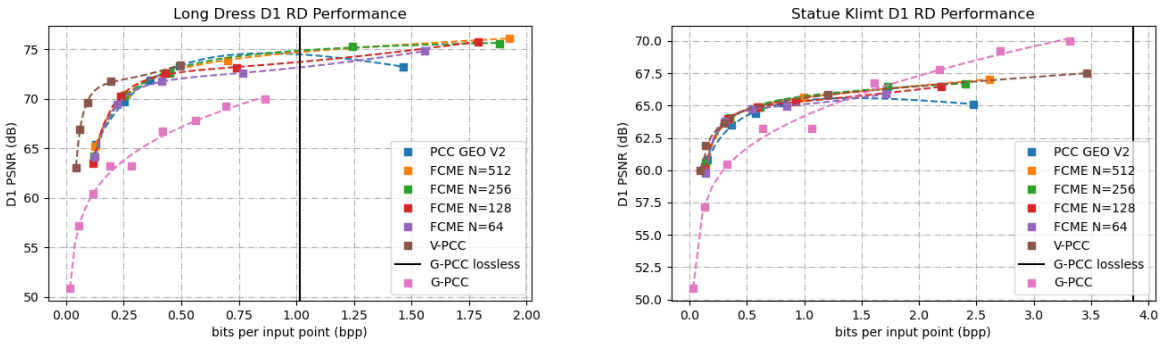
			Encoder		
Point Cloud	**Metric**	**BD**	***G-PCC***	***V-PCC***	***PCC GEO V2***
Long Dress	D1	rate	-141.91	58.39	-3.40
		PSNR	5.13	-3.40	-0.25
	D2	rate	-51.17	59.61	-2.32
		PSNR	3.43	-4.51	-0.22
Statue Klimt	D1	rate	-105.06	33.75	-7.95
		PSNR	1.70	-0.75	0.00
	D2	rate	-32.33	-21.76	-9.30
		PSNR	0.65	0.66	0.06

As was the case with the FTE, the PTE outperforms the G-PCC encoder in a similar fashion, being more biased towards dense PCs. Also, analogous to the FTE, the V-PCC encoder outperforms the PTE in a similar manner, being more biased towards dense PCs. On the other hand, the PCC

GEO V2 encoder outperforms the PTE on almost all test PCs (see Annex A Table A.4 for more results) in terms of D1 PSNR and D2 PSNR. However, this comes at the cost of a slight increase in rate, so the two encoders come out as similar in performance. It seems that the patch-based inputs may have had more of an impact on performance than the Transformer's attention-based architecture since the PTE, which uses the Transformer architecture but not patch-based inputs, had worse overall performance compared with the FTE, which uses the Transformer architecture and patch-based inputs. This loss of performance for the PTE compared to the FTE can be observed comparing Tables 4.5 and 4.6. While there is similar performance for dense PCs, the PTE has around equal performance to the PCC GEO V2 encoder for sparse PCs as opposed to the FTE which surpasses the PCC GEO V2 encoder for sparse PCs.

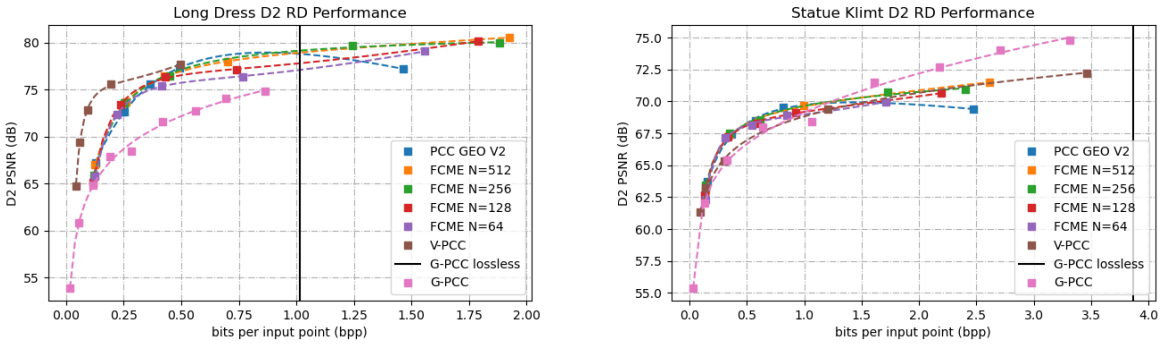
4.7.3 Full ConvMixer Encoder

In Figures 4.15 and 4.16 and Tables 4.7, 4.8 and 4.9, the RD performance for the FCME can be observed.



(a) Long Dress. (b) Statue Klimt.

Figure 4.15: D1 PSNR RD performance graphs for the FCME.



(a) Long Dress. (b) Statue Klimt.

Figure 4.16: D2 PSNR RD performance graphs for the FCME.

Table 4.7: BD metrics for the FCME with N feature map channels compared to the G-PCC encoder. The BD-rate is in percentage and the BD-PSNR is in dB. The best value is in bold.

			N			
Point Cloud	**Metric**	**BD**	***512***	***256***	***128***	***64***
Long Dress	D1	rate	-225.81	-178.78	**-251.17**	-232.35
		PSNR	**5.77**	5.67	5.54	5.07
	D2	rate	-110.53	-82.91	**-112.26**	-101.92
		PSNR	**4.27**	4.13	3.97	3.42
Statue Klimt	D1	rate	-107.56	-115.87	-123.65	**-126.42**
		PSNR	1.93	2.09	2.09	**2.26**
	D2	rate	-34.27	**-38.63**	-33.40	-35.18
		PSNR	0.67	**0.77**	0.61	0.76

As can be seen in Table 4.7, the FCME clearly outperforms the G-PCC encoder by, at least, 0.5 dB in terms of BD-PSNR while also experiencing a reduction of 33% in terms of BD-rate. Similar to previous encoders, it also has a bias towards dense PCs and underperforms the G-PCC encoder for high bitrates on sparse PCs.

In terms of the impact of N, the 512 and 256 channel models have better BD-PSNR gains in almost all the PCs, with some sparse PCs also seeing the best performance in the 64 channel model. In contrast, the 128 channel model has better BD-rate reductions on the dense PCs, with the sparse PCs being more split between the 512, 256 and 64 channel models for the best BD-rate performance (see Annex A Table A.5 for more results).

Table 4.8: BD metrics for the FCME with N feature map channels compared to the V-PCC encoder. The BD-rate is in percentage and the BD-PSNR is in dB. The best value is in bold.

			N			
Point Cloud	**Metric**	**BD**	***512***	***256***	***128***	***64***
Long Dress	D1	rate	53.06	56.16	**51.83**	57.45
		PSNR	**-2.17**	-2.48	-2.35	-2.62
	D2	rate	54.60	57.10	**53.79**	58.97
		PSNR	**-2.98**	-3.34	-3.20	-3.53
Statue Klimt	D1	rate	13.67	**13.50**	16.42	22.32
		PSNR	-0.19	**-0.13**	-0.28	-0.39
	D2	rate	-35.77	**-38.43**	-28.55	-25.90
		PSNR	0.83	**0.88**	0.65	0.67

As can be seen in Table 4.8, in the case of the V-PCC encoder, once again, it outperforms the DL-based solutions for denser PCs, struggling to keep up on sparser PCs.

In terms of the impact of N, the 512 and 256 channel models have better BD-PSNR gains in almost all the PCs. In contrast, the 128 channel model has better BD-rates on the dense PCs. In the sparse PCs, the 512 and 256 channel models have better BD-rates.

Table 4.9: BD metrics for the FCME with N feature map channels compared to the PCC GEO V2 encoder. The BD-rate is in percentage and the BD-PSNR is in dB. The best value is in bold.

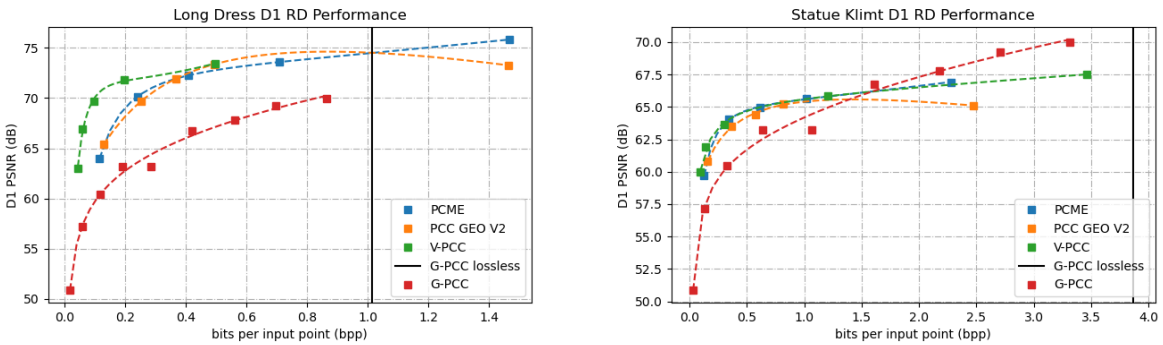
			N			
Point Cloud	**Metric**	**BD**	***512***	***256***	***128***	***64***
Long Dress	D1	rate	-12.60	-9.25	**-13.79**	2.49
		PSNR	**0.26**	0.24	-0.06	-0.59
	D2	rate	-12.75	-9.62	**-13.30**	2.27
		PSNR	**0.31**	0.29	-0.05	-0.71
Statue Klimt	D1	rate	-32.41	-31.01	**-35.32**	-26.27
		PSNR	0.55	**0.59**	0.44	0.29
	D2	rate	**-10.63**	-9.88	-4.61	-1.23
		PSNR	**0.25**	0.24	-0.05	-0.17

Observing the values in Table 4.9, the FCME outperforms the PCC GEO V2 encoder in every test PC. In terms of BD-PSNR gain, either the 512 or 256 channel models see an improvement in performance. The same can be said for the 512 or 128 channel models for an improvement in BD-rate. In this case, increasing the capacity of the model usually increases its overall performance, albeit marginally and with an impact on model complexity.

So, is the increase in performance mainly contributed by the patch-based input representation after all? To answer this question, we have to conduct an analysis of the final encoder, the PCME.

4.7.4 Progressive ConvMixer Encoder

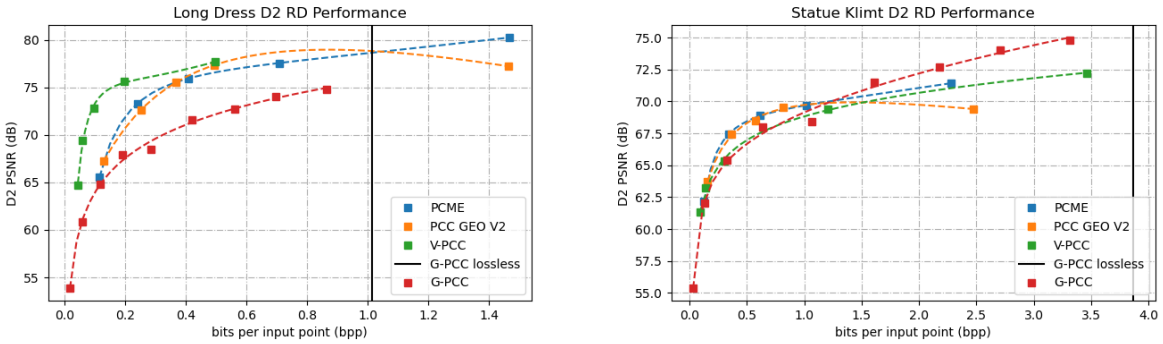
In Figures 4.17 and 4.18 and Table 4.10, the RD performance for the PCME can be observed.



(a) Long Dress.

(b) Statue Klimt.

Figure 4.17: D1 PSNR RD performance graphs for the PCME.



(a) Long Dress.

(b) Statue Klimt.

Figure 4.18: D2 PSNR RD performance graphs for the PCME.

Table 4.10: BD metrics for the PCME compared to the G-PCC, V-PCC and PCC GEO V2 encoders. The BD-rate is in percentage and the BD-PSNR is in dB.

			Encoder		
Point Cloud	**Metric**	**BD**	***G-PCC***	***V-PCC***	***PCC GEO V2***
Long Dress	D1	rate	-227.08	53.58	-12.59
		PSNR	5.60	-2.42	0.16
	D2	rate	-103.96	55.01	-12.60
		PSNR	4.01	-3.29	0.19
Statue Klimt	D1	rate	-118.40	11.65	-36.07
		PSNR	2.16	-0.20	0.56
	D2	rate	-39.08	-37.58	-15.40
		PSNR	0.89	0.95	0.33

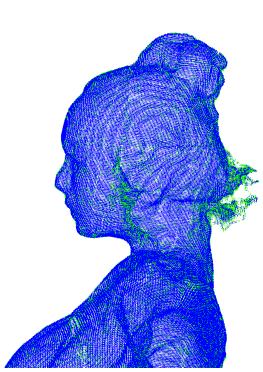
As was the case with the FCME, the PCME outperforms the G-PCC encoder in a similar fashion, being more biased towards dense PCs. Meanwhile, the V-PCC encoder outperforms the PCME at lower rates, being more biased towards dense PCs. On the other hand, the PCME outperforms the PCC GEO V2 encoder on all test PCs. This improvement in performance is translated into an increase in PSNR with a significant reduction in rate. On the other hand, looking at Figures 4.17 and 4.18, this gain in performance is not general. It mostly occurs at lower and higher bitrates. With medium bitrates, the PCC GEO V2 encoder is slightly better.

On another note, the PSNR gains of the PCME are, usually, not as high as its full variant counterpart. This fact makes it clear that patch-based inputs have a positive impact on performance.

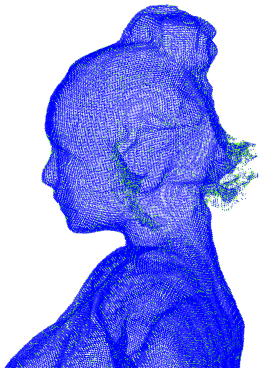
4.7.5 Qualitative study

While it is very important to use objective metrics when evaluating codec's RD performance, subjective quality is equally, if not more important. While no full subjective quality comparisons were

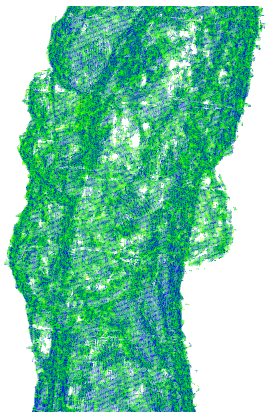
done involving the results of this study, we present in Figure 4.19 some images showing the difference between the original test PCs and the reconstructed PCs. An edge case for each PC is presented, more specifically, the reconstructed PC with the highest bitrate model for the PCME and PCC GEO V2 encoder were chosen for this part of the study. Both models have similar bitrates for the encoded PCs (less than 10% difference), so a qualitative study is possible. This was also the bitrate model where the most obvious visual difference between the original and reconstructed PC could be perceived.



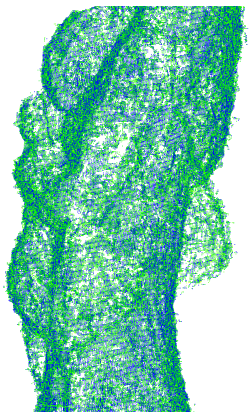
(a) PCC GEO V2 vs original.



(b) PCME vs original.



(c) PCC GEO V2 vs original.



(d) PCME vs original.

Figure 4.19: Difference between the original test PCs and the reconstructed PCs. Points in blue mean that the reconstructed point is close (closer than, on average, four voxels) to the original, while points in green mean that the reconstructed point is far (farther than, on average, four voxels) from the original.

In the case of the Long Dress test PC, it is clear from Figure 4.19a that the PCC GEO V2 encoder is more inaccurate when reconstructing the woman's hair than the PCME. On the other hand, the sparse Statue Klimt PC is not very accurately reconstructed by any of the codecs (Figures 4.19c and 4.19d). However, there are a lot more green points (inaccurate predictions) in the reconstructed PC from the PCC GEO V2 encoder than the PCME. Overall, visually, the highest bitrate model of the PCME has better performance than the highest bitrate model of the PCC GEO V2 encoder.

4.7.6 Final experiment remarks

The previous sections explained the overall training and testing process of the DL-based PCC models, comparing the performance of all the encoders presented with the G-PCC, V-PCC and PCC GEO V2 encoders. In conclusion, attention-based TELs did not seem to contribute much to the overall performance of the model. What made the biggest difference was the depthwise and pointwise convolutions of the CMELs accompanied by patch-based inputs. These operations enabled more efficient feature extraction, which ultimately resulted in more efficient PC compression. In the end, the better encoders were the ConvMixer-based ones, with the Transformer-based ones lacking behind. A summary of the best metrics for each test PC is displayed in Table 4.11.

Table 4.11: Summary of BD metrics for the proposed encoders compared to the PCC GEO V2 encoder. The BD-rate is in percentage and the BD-PSNR is in dB with the best values in bold. For the FTE, the 256 channel model was chosen as its overall best. For the FCME, the 512 channel model was chosen as its overall best.

			Encoder			
Point Cloud	**Metric**	**BD**	***FTE***	***PTE***	***FCME***	***PCME***
Long Dress	D1	rate	-0.84	-3.40	**-12.60**	-12.59
		PSNR	-0.14	-0.25	**0.26**	0.16
	D2	rate	-1.35	-2.32	**-12.75**	-12.60
		PSNR	-0.17	-0.22	**0.31**	0.19
Statue Klimt	D1	rate	-29.13	-7.95	-32.41	**-36.07**
		PSNR	0.49	0.00	0.55	**0.56**
	D2	rate	-11.23	-9.30	-10.63	**-15.40**
		PSNR	0.27	0.06	0.25	**0.33**

Finally, an important characteristic of PCC encoders is their complexity. In Table 4.12, the complexity of the proposed encoders compared to the PCC GEO V2 is presented. The complexity is calculated as a percentage of the time to encode and decode a PC with the proposed encoders ($t_{proposed}$) over the time to encode and decode a PC with PCC GEO V2 (t_{geo}) and is defined by eq. 4.4.

$$complexity = 100 \cdot \left(\frac{t_{proposed}}{t_{geo}} - 1 \right) \tag{4.4}$$

Table 4.12: Complexity of the proposed encoders compared to the PCC GEO V2 encoder.

	FTE				**PTE**	**FCME**				**PCME**
N	***64***	***128***	***256***	***512***	-	***64***	***128***	***256***	***512***	-
Complexity	−5%	−2%	−5%	30%	42%	−13%	−4%	−8%	−6%	−7%

5 Extending DL-based Geometry PCC to include color information

Many PCC solutions, like the one proposed, focus in compressing the geometry of PCs. While geometry is a very important feature of PCs, most of them have other attributes, most notable, color information. Therefore, it is desirable that, besides compressing the geometry information, the codec also compresses attribute information. In fact, the MPEG PCC codecs, G-PCC and V-PCC, support this functionality. To allow all DL PCC solutions to also compress attribute information, an extension to DL-based geometry PCC is proposed.

In the following sections, changes to the geometry PCC pipeline will be presented, most notably, DL architectural and training loss function changes. Additionally, an experiment testing and studying the proposed implementation will be presented that also studies the impact of different encoding color spaces.

5.1 Overall DL-based Geometry PCC Pipeline Changes

5.1.1 Architectural changes

As shown in Figure 3.2, most current state-of-the-art PCC employs a 3D block-based representation for the input PC (binary occupancy map). More specifically, the PC is first divided into 3D blocks where each voxel occupancy bit is either a 0 representing an empty voxel, or a 1 representing an occupied voxel. Then, each block is compressed independently, with the final main bitstream comprised of the information of the encoded blocks. On a more technical level, what is processed is a one channel 3D block containing the geometry information of the PC represented by a binary occupancy map. To extend this approach to include color information, instead of a one channel 3D block, a four channel 3D block can be used.

The approach is similar to that used in representing the geometry information in a 3D block. For each of the extra three channels, each one would have a value with the color component where there was a point and a 0 where there was no point.

This color encoding method implies architectural changes in the DL model, not very complex or extensive. Instead of outputting only one channel of 3D block information, the model needs to

output four channels of 3D block information. In practice, the number of output channels in the Synthesis Transform of Figure 3.2 needs to be set to four. Given the abstract nature of machine learning, the model would then learn to compress geometry and color information simultaneously. For example, the altered Synthesis Transform from the PCC GEO V2 work can be observed in Figure 5.1.

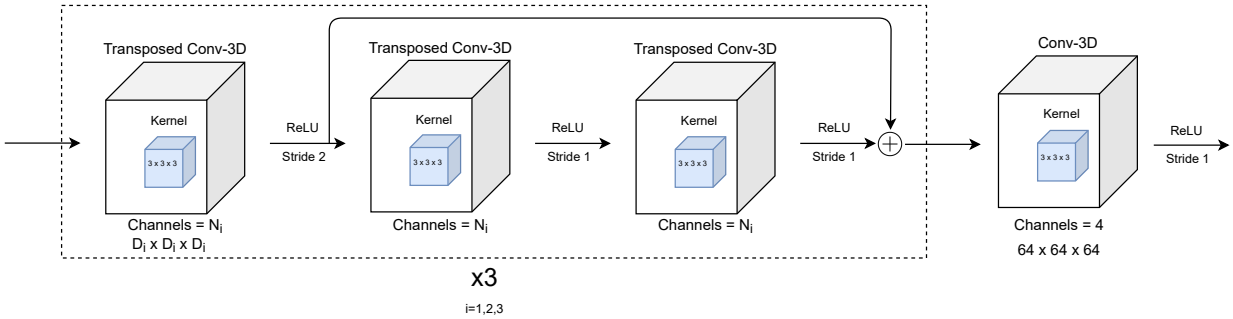


Figure 5.1: Altered Synthesis Transform from the PCC GEO V2 work to also process color information.

5.1.2 Model loss modification

Now that the model outputs color information, a color distortion loss component is now needed for use in the training of the model. In practice, the distortion component of the eq. 3.1 would have to contain two terms: one for geometry distortion and one for color distortion. Finally, a factor β is needed to control the trade-off between geometry and color distortion (eq. 5.1).

$$loss = \lambda \cdot (geometry\ distortion + \beta \cdot color\ distortion) + rate \tag{5.1}$$

5.2 Color PCC DL Model Training and Testing

Now that the general model pipeline has been designed, it's necessary to indicate, specifically, the hyper parameters and methods used. The subsequent sections will present the:

- Block partition method (same as described in Section 4.5.1);
- Training dataset;
- Loss function and its parameters;
- Experimental conditions (testing dataset, performance metrics and benchmarking);
- Experimental results.

Furthermore, the study will be focused on extending two state-of-the-art published works, PCC GEO V2 [20] and PCC GEO SLICING [6], to also compress PC color information.

5.2.1 Training dataset

For the training dataset, PCs in the Joint Photographic Experts Group (JPEG) Point Cloud Coding Common Training and Testing Conditions were used, voxalized with a depth of 10 bits. To transform the dataset into usable 3D blocks, first, the PCs suffered an octree partition of depth 4, leading into $64 \times 64 \times 64$ 3D blocks. Since, by nature, most of those blocks have only $5 - 10\%$ filled voxels, only the 4000 blocks with the greastest number of points were used for training. This means that many blocks from the partitioned PCs were not used in the training of the model.

5.2.2 Loss function

The geometry distortion and rate terms of the loss function, they are similar to the terms described in Section 4.5.3, repeated here for easier reading.

$$FL(v,u) = \begin{cases} -\alpha(1-v)^{\gamma}\log v, & u = 1 \\ -(1-\alpha)v^{\gamma}\log(1-v), & u = 0 \end{cases} \tag{5.2}$$

$$coding\ rate = \frac{\sum \ln(p_{qlr}(qlr)) + \sum \ln(p_{qhlr}(qhlr))}{\ln\left(\frac{1}{2}\right) \times N} \tag{5.3}$$

For the color distortion metric, since color is a continuous value, the mean squared error loss was used (eq. 5.4).

$$MSE = \frac{1}{N}\sum (color_{original} - color_{reconstructed})^2 \tag{5.4}$$

While $color_{reconstructed}$ is the color of the reconstructed voxel, $color_{original}$ can have different values depending on the context. In standard PC distortion evaluation applications, $color_{original}$ is the color of the point in the original PC closest to the reconstructed point. Although this loss can be implemented and used in training, not only is it very expensive computationally, but it did not offer any obvious gains in performance. Therefore, two other color distortion losses were used, denominated *colorV1* loss and *colorV2* loss and presented in Figure 5.2.

In the *colorV1* loss, the $color_{original}$ portion of the loss is the same as the input block color component, but $color_{reconstructed}$ are the reconstructed colors only for the voxels that had points in the original block, even if the model did not evaluate that voxel as having a point. This leads to a more independent color loss function such that the color reconstruction process does not depend as much on a good model geometry reconstruction.

In the *colorV2* loss, the $color_{reconstructed}$ are the reconstructed colors only for the voxels that the model evaluated as having a point, but $color_{original}$ is the average of the color component in the original block with a window of size $m \times m \times m$ (for each color component) centered in the reconstructed point voxel. This leads to a more lenient color loss function placing more emphasis on

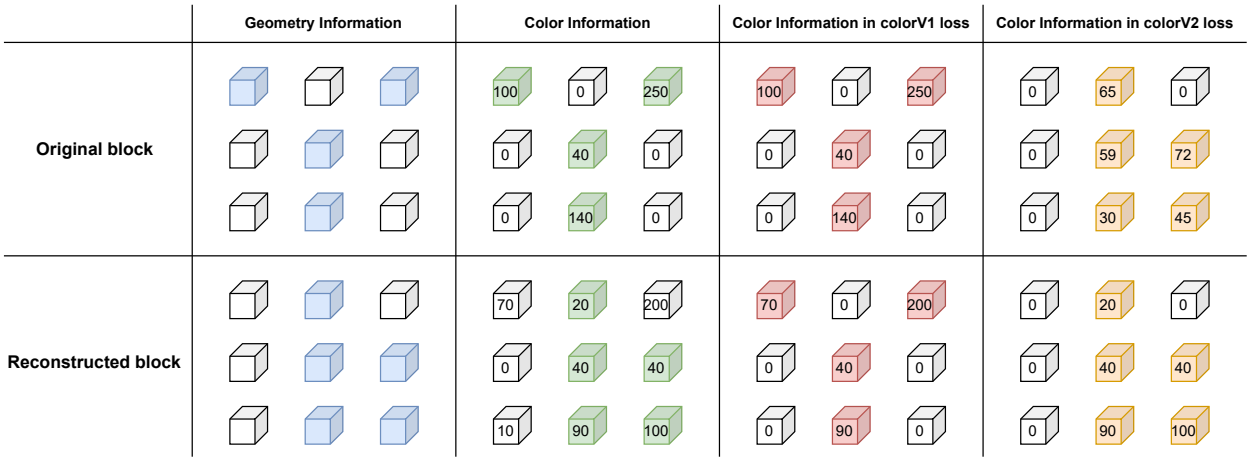


Figure 5.2: Illustration of the application of the *colorV1* loss and *colorV2* loss in the 2D case with only one color component. Blue cubes refer to filled voxels, green cubes refer to filled voxels in the geometry information, red cubes refer to color values in the *colorV1* loss and orange cubes refer to color values in the *colorV2* loss considering a 3×3 window.

reconstructing color voxels with values close to their nearest neighbors in the original block in case of an incorrect geometry reconstruction. In the conducted experiments, $m = 7$ was set empirically.

As for the β factor, it is highly dependent on the implementation of the geometry distortion loss and color distortion loss. However, in the conducted experiments, changing this value did not have an obvious impact on performance, with the biggest bottleneck being the capacity of the DL model itself and the difficulty of optimizing a continuous value variable. Nevertheless, a value of 1000 was set empirically for the PCC GEO V2 encoder and a value of 40 was set empirically for the PCC GEO SLICING encoder.

5.3 Experimental conditions

5.3.1 Testing dataset

Testing was done using two PCs, *Long Dress 1300* (Figure 4.10a), to represent dense PCs, and *Statue Klimt* (Figure 4.10f), to represent sparse PCs, were used. The sparsity of the PCs was measured by calculating, for all points in a PC, the average of the euclidean distance between the point and their 20 closest neighbors and averaging these distances over all points of the PC. The Long Dress PC had a mean distance of 1.73, indicating a dense PC (mean distance less than 2), and the Statue Klimt had a mean distance of 2.38, indicating a sparse PC (mean distance greater than 2). The threshold of 2 to distinguish dense and sparse PCs was set arbitrarily as a threshold that made intuitive sense. All the PCs were voxelized to have a depth of 10 bits, with an octree division of depth 4 to obtain $64 \times 64 \times 64$ 3D blocks.

5.3.2 Performance Metrics

To compare the RD performance of the various DL PCC models and the baseline codecs, the rate, geometry distortion and color distortion were measured. The rate and geometry distortion metric are the same as described in Section 4.6.2. Using the same software to calculate the geometry distortion, the color distortion is also calculated similarly to the D1 PSNR metric, except that instead of using the geometry coordinates, it uses the color components.

Finally, an overall geometry and color distortion will be presented, the PCQM metric.

5.3.3 Benchmarking

For benchmarking purposes, the V-PCC with the VVC video encoder (in intra mode with only one frame) was used as a baseline, compressing the test PCs with a similar RD trade-off to the DL models. The V-PCC parameters used in all the encodings were the default common test conditions provided by the V-PCC github repository[1]. This time, the G-PCC encoder was not used as a baseline because it was shown previously to have a much inferior performance to the V-PCC encoder.

5.4 Experimental results

RD performance is an important criterion when targeting compression efficiency. The DL PCC encoders proposed and the benchmarking encoders will be compared based on the rate and distortion operation points reached when encoding the test PCs, i.e., based on their RD performance. Also, for each DL model, both proposed color distortion losses will be evaluated. In the subsequent study, CL_1 refers to the $colorV1$ loss and CL_2 refers to the $colorV2$ loss. A cursory analysis will include only one RD point for the V-PCC, PCC GEO V2 and PCC GEO SLICING codecs, but graphs detailing the behavior over a wide range of RD points will be presented at the end.

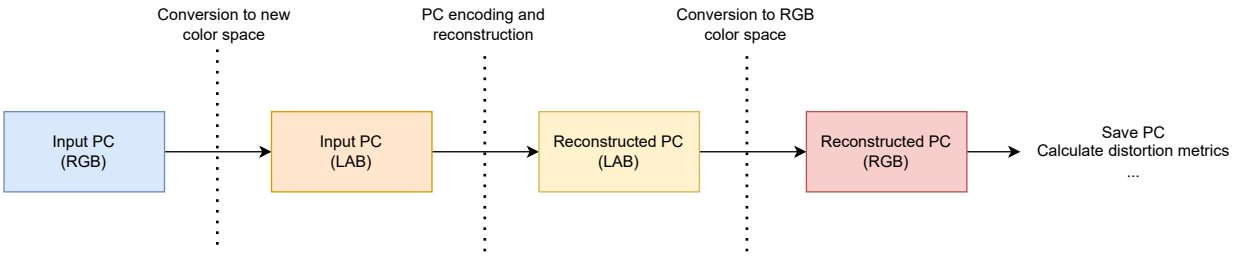


Figure 5.3: Example of the pipeline when encoding and decoding a PC in the LAB color space.

Finally, the DL models were also trained to compress color information in four different color spaces: RGB, YCbCr, LAB and HSV. Performance differences between them will also be studied. Since all the test PCs have RGB information, it will be necessary to convert RGB color to other

[1] `https://github.com/MPEGGroup/mpeg-pcc-tmc2.git`

color spaces and vice-versa. Figure 5.3 shows an example of this pipeline. The new color space is only used when encoding and decoding the PC. In the case of the V-PCC encoder, only its default coding color space will be used. Distortion metric calculation is always processed in the RGB color space, however, [8] converts the RGB color space to the YUV color space, so the C1, C2 and C3 metrics in the subsequent sections will refer to the color distortions in the Y, U and V color components, respectively. The YUV color space defines three color components: luminance (Y), blue projection (U) and red projection (V). The equations defining the color conversion from RGB to YUV can be found in Annex B.1.

5.4.1 RGB color space

The RGB color space defines three color components: red (R), green (G) and blue (Cr). All the test PCs have their color information in the RGB color space, so there will be no need for color conversions.

Table 5.1: Performance metrics and bpp for the different encoders with the Long Dress PC. The C1, C2 and C3 metrics refer to the three color components. The D1, D2, C1, C2 and C3 metrics are in dB. The PCQM values listed have been multiplied by 1000. The best value for all the metrics is in bold.

Encoder	**BPP**	**D1**	**D2**	**C1**	**C2**	**C3**	**PCQM**
V-PCC	**0.70**	72.1	76.0	**33.3**	**36.2**	**35.3**	**5.3**
PCC GEO V2 - CL_1	1.09	71.0	74.4	25.4	30.6	28.9	12.3
PCC GEO V2 - CL_2	0.95	73.5	77.5	24.9	32.0	30.2	11.0
PCC GEO SLICING - CL_1	0.98	73.9	78.0	27.3	33.0	30.9	7.6
PCC GEO SLICING - CL_2	0.80	**74.0**	**78.1**	25.1	32.7	30.8	9.8

In Table 5.1, it's possible to see a trend that will be observed also with other color spaces. For the dense PC Long Dress, the V-PCC encoder dominates in the bpp used and in the metrics C1, C2, C3 and PCQM. However, it struggles to keep up in the D1 and D2 geometry distortion metrics (see Figure 5.4).

In terms of the DL-based encoders, the PCC GEO SLICING encoder has superior overall performance over the PCC GEO V2 encoder. That is to be expected because the PCC GEO SLICING work builds upon and improves the PCC GEO V2 work.

In terms of the color distortion loss, the DL encoders have different behaviors. When using the $colorV2$ loss over the $colorV1$ loss, both of them manage to achieve a reduction in overall encoding bpp and an increase in D1 and D2 metrics (not so much in the case of the PCC GEO SLICING encoder). However, that behaviour is not observed in the case of the color distortion metrics. The PCC GEO SLICING encoder sees a reduction in the color distortion when using the $colorV2$ loss,

while the PCC GEO V2 encoder is the opposite, with the exception of the first color component.

In conclusion, with respect to the RGB color space and a denser PC, the V-PCC encoder is the clear winner (see Figure 5.4c), as complemented by the PCQM values presented, followed by the PCC GEO SLICING encoder and finally the PCC GEO V2 encoder.

Table 5.2: Performance metrics and bpp for the different encoders with the Statue Klimt PC. The C1, C2 and C3 metrics refer to the three color components. The D1, D2, C1, C2 and C3 metrics are in dB. The PCQM values listed have been multiplied by 1000. The best value for all the metrics is in bold.

Encoder	**BPP**	**D1**	**D2**	**C1**	**C2**	**C3**	**PCQM**
V-PCC	2.56	65.4	69.8	**29.0**	**34.4**	**35.9**	**7.2**
PCC GEO V2 - CL_1	1.81	64.8	68.9	23.7	29.5	32.0	17.2
PCC GEO V2 - CL_2	1.54	65.5	69.7	22.9	31.3	32.8	15.0
PCC GEO SLICING - CL_1	1.82	**65.6**	**70.0**	25.5	32.5	34.1	11.2
PCC GEO SLICING - CL_2	**1.45**	65.5	69.9	23.2	32.0	34.1	13.9

With respect to the sparse PC Statue Klimt, as one can observe in Table 5.2, it is not clear which encoder is the best. The conclusions are similar to the previous dense PC study, however, there are a few differences. Besides the encoders not being able to achieve performances as high as those of the dense PC, the V-PCC encoder uses a much higher bpp when compressing the sparse PC.

One other remark is that the PCC GEO SLICING encoder achieves similar performance in the D1 and D2 metrics when using either of the color losses. In practice, this indicates that this encoder sees less of an impact on the geometry distortion metrics when using different color distortion losses.

In conclusion, with respect to the RGB color space and a sparser PC, the V-PCC encoder is the winner in terms of overall distortion metrics (see Figure 5.5c), however, the PCC GEO SLICING encoder achieves a much lower bpp with similar geometry distortion metrics but somewhat lower color distortion metrics.

5.4.2 YCbCr color space

The YCbCr color space defines three color components: luminance (Y), chroma blue-difference (Cb) and chroma red-difference (Cr). The equations defining the color conversion from RGB to YCbCr and vice-versa can be found in Annex B.2.

Table 5.3: Performance metrics and bpp for the different encoders with the Long Dress PC. The C1, C2 and C3 metrics refer to the three color components. The D1, D2, C1, C2 and C3 metrics are in dB. The PCQM values listed have been multiplied by 1000. The best value for all the metrics is in bold.

Encoder	**BPP**	**D1**	**D2**	**C1**	**C2**	**C3**	**PCQM**
V-PCC	**0.70**	72.1	76.0	**33.3**	**36.2**	**35.3**	**5.3**
PCC GEO V2 - CL_1	0.81	71.2	74.7	24.5	28.8	28.4	14.8
PCC GEO V2 - CL_2	0.76	72.6	76.5	24.2	30.5	29.1	14.2
PCC GEO SLICING - CL_1	0.81	**74.2**	**78.3**	26.2	31.5	30.0	9.4
PCC GEO SLICING - CL_2	0.75	73.6	77.9	24.8	30.9	29.7	11.2

As one can observe in Table 5.3, encoding in the YCbCr color space results in an overall reduction in all the presented color distortion metrics comparing to encoding in the RGB color space. Meanwhile, the geometry distortion metrics are similar to those of the RGB color space. One remark is that, in the case of the dense PC, the PCC GEO SLICING encoder shows a significant increase in the D1 and D2 geometry metrics when using the $colorV1$ loss. Finally, the overall encoding bpp is also lower compared to the RGB color space. That being said, the V-PCC encoder still dominates in terms of the overall quality of the reconstructed PC.

Table 5.4: Performance metrics and bpp for the different encoders with the Statue Klimt PC. The C1, C2 and C3 metrics refer to the three color components. The D1, D2, C1, C2 and C3 metrics are in dB. The PCQM values listed have been multiplied by 1000. The best value for all the metrics is in bold.

Encoder	**BPP**	**D1**	**D2**	**C1**	**C2**	**C3**	**PCQM**
V-PCC	2.56	65.4	69.8	**29.0**	**34.4**	**35.9**	**7.2**
PCC GEO V2 - CL_1	1.38	64.8	68.9	22.4	27.1	31.4	22.8
PCC GEO V2 - CL_2	**1.31**	65.3	69.6	22.2	28.4	32.0	19.5
PCC GEO SLICING - CL_1	1.51	**65.7**	**70.1**	24.8	30.8	33.6	13.1
PCC GEO SLICING - CL_2	1.34	65.4	69.7	23.0	30.1	33.3	15.5

As for the sparse PC, the conclusions are similar to the dense PC when using the YCbCr color space.

5.4.3 LAB color space

The LAB color space defines three color components: lightness (L), red to green (A) and yellow to blue (B). The equations defining the color conversion from RGB to LAB and vice-versa can be found in Annex B.3.

Table 5.5: Performance metrics and bpp for the different encoders with the Long Dress PC. The C1, C2 and C3 metrics refer to the three color components. The D1, D2, C1, C2 and C3 metrics are in dB. The PCQM values listed have been multiplied by 1000. The best value for all the metrics is in bold.

Encoder	**BPP**	**D1**	**D2**	**C1**	**C2**	**C3**	**PCQM**
V-PCC	**0.70**	72.1	76.0	**33.3**	**36.2**	**35.3**	**5.3**
PCC GEO V2 - CL_1	0.87	69.3	72.2	23.7	30.3	27.0	18.8
PCC GEO V2 - CL_2	0.79	72.0	75.7	23.9	31.5	28.6	15.0
PCC GEO SLICING - CL_1	0.87	**74.4**	**78.6**	26.3	32.7	29.8	8.2
PCC GEO SLICING - CL_2	0.77	74.2	78.4	24.6	32.4	29.7	10.8

Table 5.6: Performance metrics and bpp for the different encoders with the Statue Klimt PC. The C1, C2 and C3 metrics refer to the three color components. The D1, D2, C1, C2 and C3 metrics are in dB. The PCQM values listed have been multiplied by 1000. The best value for all the metrics is in bold.

Encoder	**BPP**	**D1**	**D2**	**C1**	**C2**	**C3**	**PCQM**
V-PCC	2.56	65.4	69.8	**29.0**	**34.4**	**35.9**	**7.2**
PCC GEO V2 - CL_1	1.42	64.4	68.4	21.5	27.8	31.0	29.6
PCC GEO V2 - CL_2	**1.33**	65.1	69.1	21.8	29.8	32.1	24.0
PCC GEO SLICING - CL_1	1.64	**65.9**	**70.3**	25.0	32.4	33.4	11.8
PCC GEO SLICING - CL_2	1.40	65.7	70.1	23.0	31.8	33.2	14.9

When encoding in the LAB color space, the conclusions for the dense and sparse PC are similar to the conclusions of the YCbCr color space with respect to the RGB color space.

5.4.4 HSV color space

The HSV color space defines three color components: hue (H), saturation (S) and value (V). The equations defining the color conversion from RGB to HSV and vice-versa can be found in Annex B.4.

Table 5.7: Performance metrics and bpp for the different encoders with the Long Dress PC. The C1, C2 and C3 metrics refer to the three color components. The D1, D2, C1, C2 and C3 metrics are in dB. The PCQM values listed have been multiplied by 1000. The best value for all the metrics is in bold.

Encoder	**BPP**	**D1**	**D2**	**C1**	**C2**	**C3**	**PCQM**
V-PCC	**0.70**	72.1	76.0	**33.3**	**36.2**	**35.3**	**5.3**
PCC GEO V2 - CL_1	1.48	70.0	73.2	22.7	29.5	26.5	28.2
PCC GEO V2 - CL_2	1.22	71.5	75.1	22.8	28.4	24.7	27.6
PCC GEO SLICING - CL_1	1.28	73.5	77.5	24.4	30.6	28.5	13.6
PCC GEO SLICING - CL_2	0.91	**73.8**	**77.9**	23.5	28.8	25.0	19.9

When encoding with the HSV color space, with regards to the dense PC, overall, all the distortion metrics and bpp values are worse than the values when using the previous color spaces. Furthermore, the DL-based encoders, with respect to the color distortion metrics, prefer using the $colorV1$ loss.

In the case of dense PCs, encoding with the HSV color space seems to be an incorrect approach (see Figure 5.4).

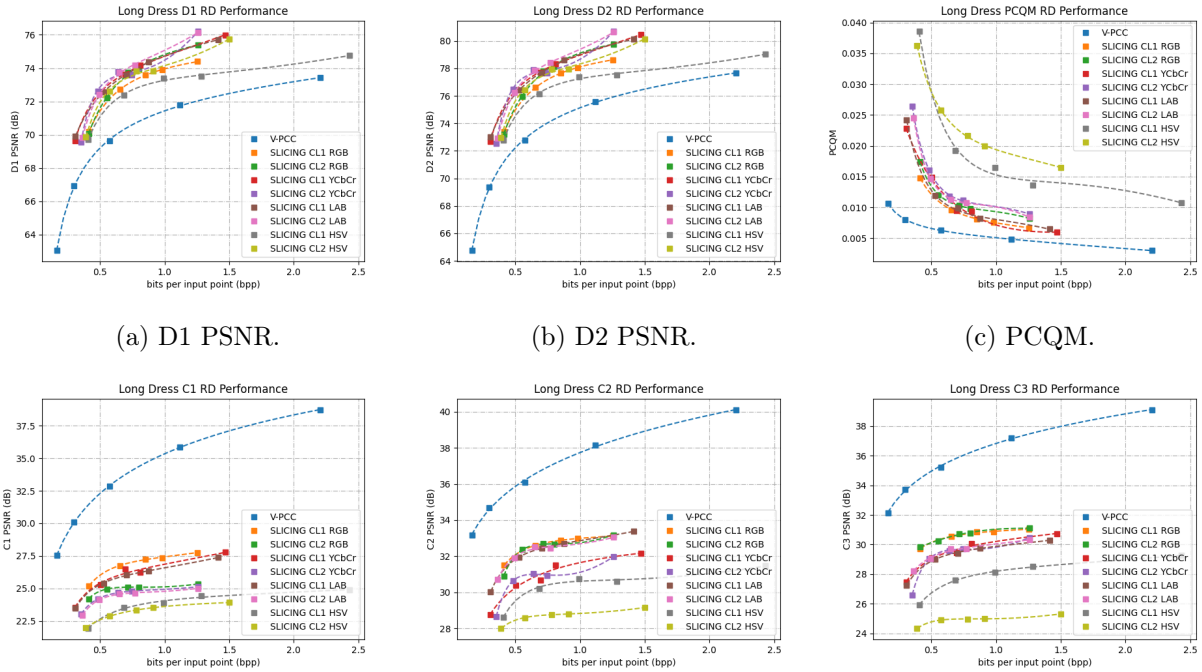
Table 5.8: Performance metrics and bpp for the different encoders with the Statue Klimt PC. The C1, C2 and C3 metrics refer to the three color components. The D1, D2, C1, C2 and C3 metrics are in dB. The PCQM values listed have been multiplied by 1000. The best value for all the metrics is in bold.

Encoder	**BPP**	**D1**	**D2**	**C1**	**C2**	**C3**	**PCQM**
V-PCC	2.56	65.4	69.8	**29.0**	**34.4**	**35.9**	**7.2**
PCC GEO V2 - CL_1	1.70	64.5	68.6	20.7	31.0	29.9	37.4
PCC GEO V2 - CL_2	1.74	65.0	69.2	21.1	31.2	29.5	30.8
PCC GEO SLICING - CL_1	1.99	65.6	69.9	23.7	32.9	32.1	15.3
PCC GEO SLICING - CL_2	**1.56**	**65.6**	**69.9**	22.4	32.3	30.1	19.5

In the case of the sparse PC, the reduction in quality of the reconstructed PC, when comparing with previous color spaces, is not as bad in terms of the geometry distortion metrics. Still, using the HSV color space does not offer a meaningful increase in performance in any of the metrics presented (see Figure 5.5).

5.4.5 Final experiment remarks

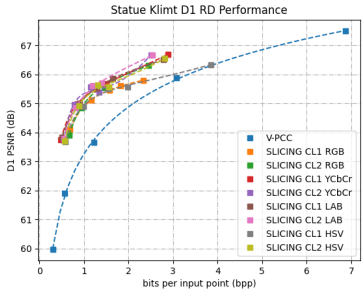
After analyzing the performance in encoding the PCs in four different color spaces, a superior encoding color space may not be as obvious as it seems. Based purely on the metrics presented, the RGB color space might be a good candidate for the best. However, one cannot focus only on numbers when selecting the best of something, because general public opinion is also a very important metric, especially in evaluating compression solutions. Therefore, we end this study by presenting images of all the best reconstructed PCs. For all the color spaces, as for the DL-based encoders, only the best PC in terms of the PCQM metric will be presented. Finally, graphs of the RD performance for the V-PCC and PCC GEO SLICING encoders for the relevant metrics are shown in Figures 5.4 and 5.5. In terms of the DL-based solution, all the color spaces and color loss functions RD curves are presented. The PCC GEO V2 encoder is not included because it was shown to have overall inferior performance to the PCC GEO SLICING encoder.



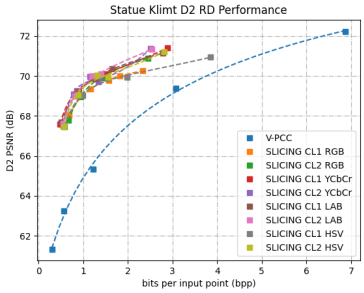
(a) D1 PSNR. (b) D2 PSNR. (c) PCQM.

(d) C1 PSNR (Y channel). (e) C2 PSNR (U channel). (f) C3 PSNR (V channel).

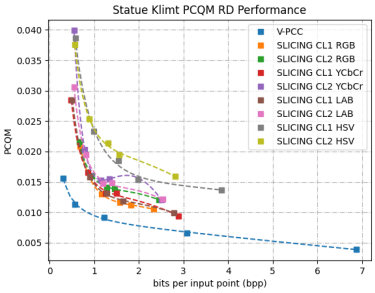
Figure 5.4: RD graphs for the relevant metrics on the Long Dress test PC.



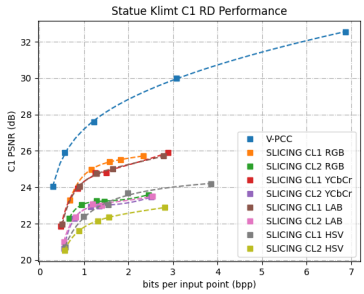
(a) D1 PSNR.



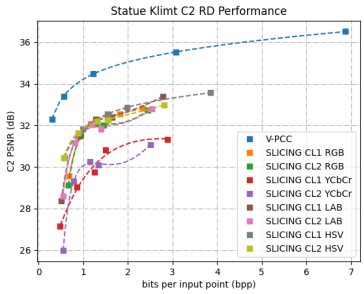
(b) D2 PSNR.



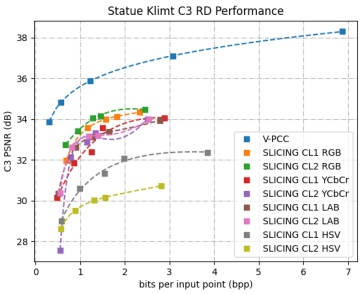
(c) PCQM.



(d) C1 PSNR (Y channel).

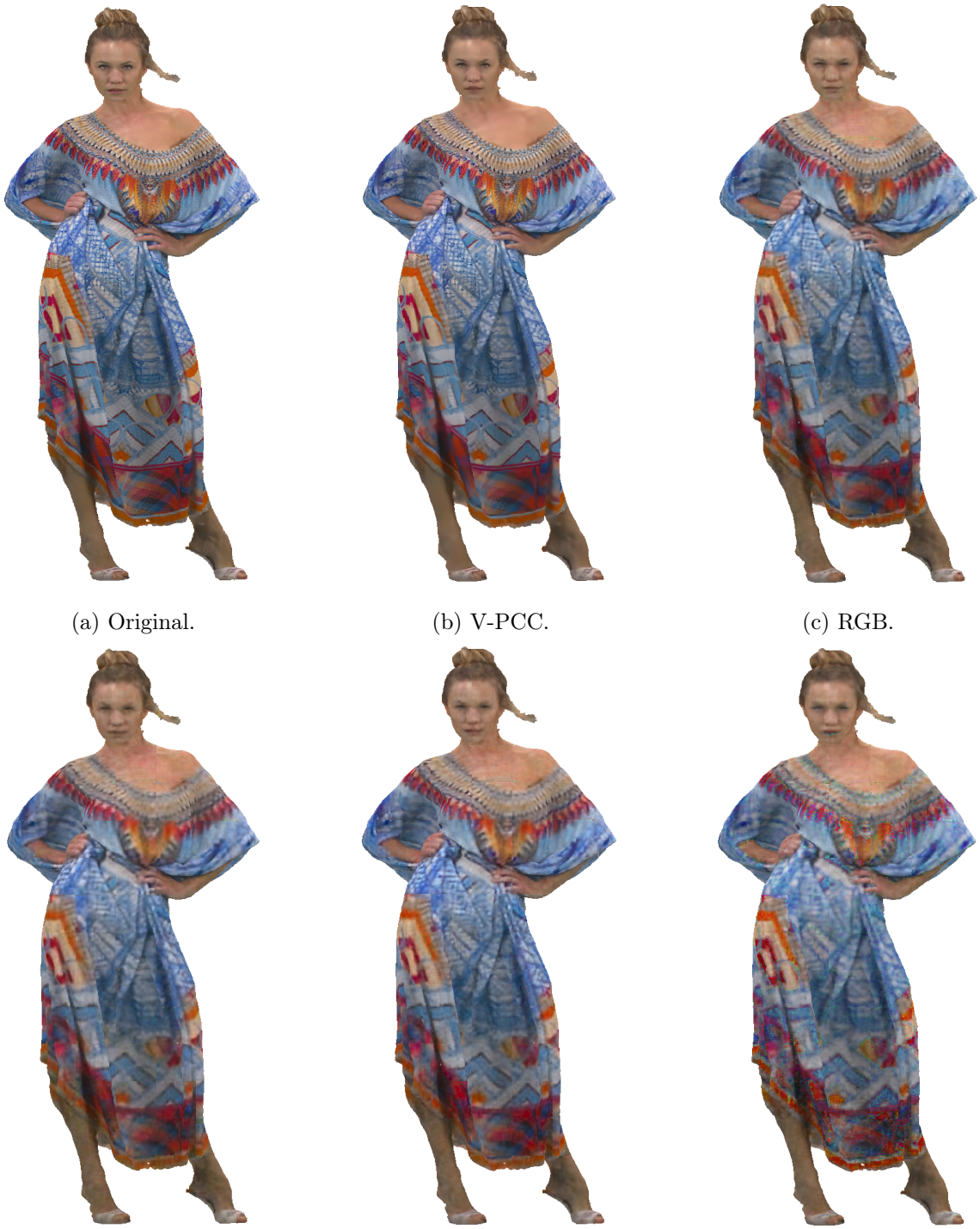


(e) C2 PSNR (U channel).



(f) C3 PSNR (V channel).

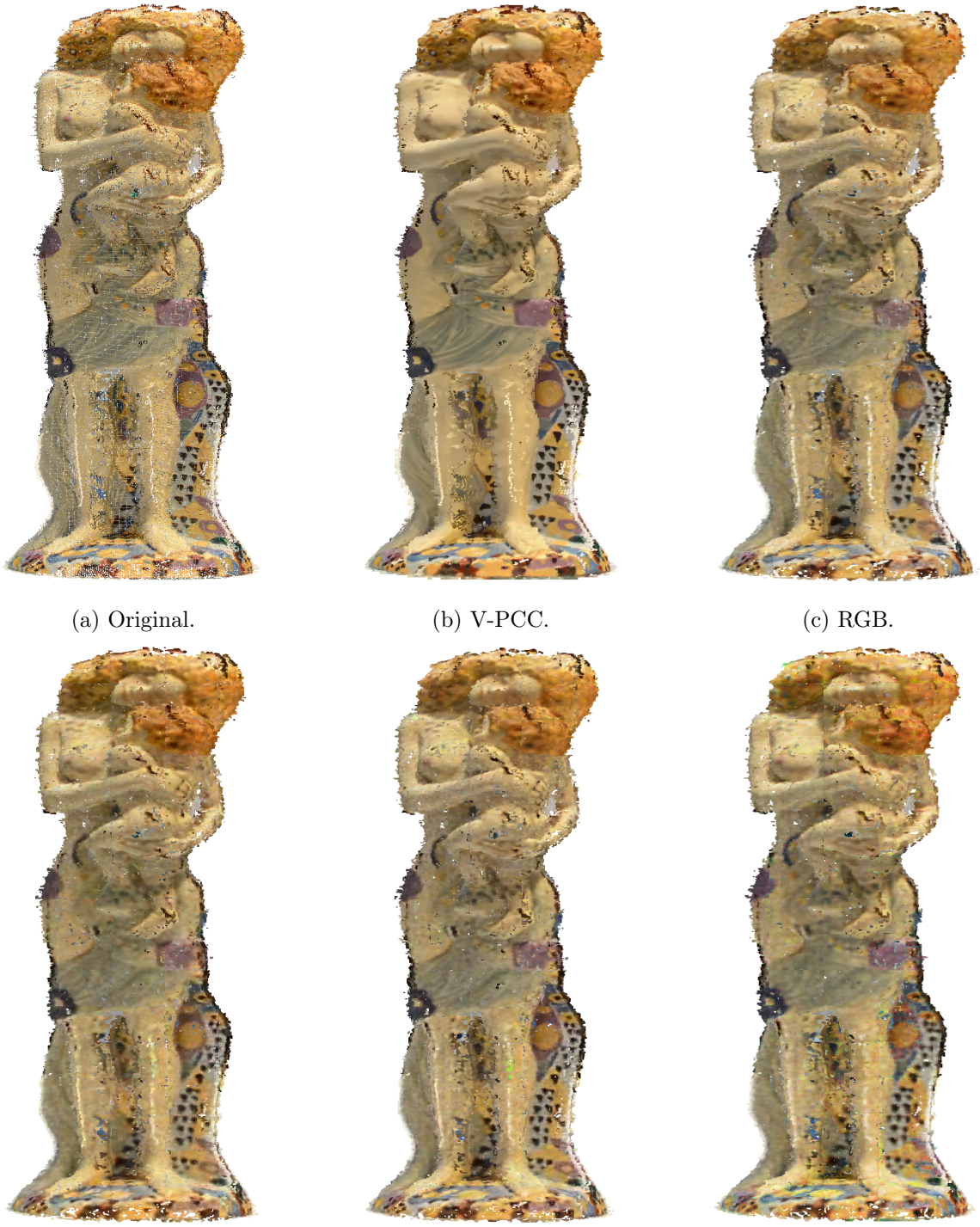
Figure 5.5: RD graphs for the relevant metrics on the Statue Klimt test PC



(a) Original. (b) V-PCC. (c) RGB.

(d) YCbCr. (e) LAB. (f) HSV.

Figure 5.6: Best of the reconstructed Long Dress PCs. For every color space, the reconstructed PC using the PCC GEO SLICING encoder with the *colorV*1 loss is shown.



(a) Original. (b) V-PCC. (c) RGB.

(d) YCbCr. (e) LAB. (f) HSV.

Figure 5.7: Best of the reconstructed Statue Klimt PCs. For every color space, the reconstructed PC using the PCC GEO SLICING encoder with the $colorV1$ loss is shown.

6 Conclusion

6.1 Final Dissertation remarks

This Dissertation presented two topics that pertained to advances in 3D PCC using DL: the proposal of new DL-based encoders with the objective of efficient PCC and the extension of state-of-the-art DL-based PC codecs to also compress color information.

In terms of the proposed encoders, four were presented based on the DL Transformer architecture and patch inputs. Trained with the Tensorflow framework, the best encoders achieved an increase in RD performance compared with the DL-based PCC benchmark and other relevant codecs, like G-PCC and V-PCC. The performance of these encoders was measured using the geometry PC distortion metrics D1 PSNR and D2 PSNR and the rate of the compressed PC. More specifically, considering the best of the proposed encoders, a decrease of up to 30% in rate and an increase of up to 0.59 and 0.56 for the D1 PSNR and D2 PSNR metrics, respectively, was observed.

Regarding the extension of DL-based PCC to also include color information, a study was conducted in four color spaces to determine the best color space to use in coding time. Furthermore, two loss functions for the color component were proposed to measure the color distortion in training time. While being a naive approach to compressing PC color information, it gave reasonable results compared with the V-PCC baseline, with the proposed solutions lacking behind in color distortion performance, but overcoming V-PCC in geometry distortion performance. The color distortion was measured using the C1 PSNR, C2 PSNR, C3 PSNR and PCQM metrics. More specifically, compared with the V-PCC baseline, the best of the proposed solutions achieved a reduction of up to 6 dB in terms of the C1 PSNR, C2 PSNR and C3 PSNR metrics and a reduction of up to 3 units in terms of the PCQM metric. In contrast, an increase of up to 2 dB was achieved regarding the D1 PSNR and D2 PSNR metrics. Considering these results and the implementation details of compressing PC color information, all future DL-based PCC solutions in the literature should strive to be able to perform both geometry and color PCC.

6.2 Future work

For future work, all the proposed solutions can be improved by performing several changes and studies, which are listed below:

- Regarding the black box nature of DL, a fine tuning of the models' hyperparameters can be performed to maximize the RD performance for a certain RD point. Furthermore, a study of other DL-based architectures can be conducted to understand their impact and performance in PCC, like extending the proposed AE architecture to a VAE variant.

- Regarding the training of the DL-based models, more sophisticated geometry and color loss functions can be used, like the neighborhood adaptive distortion loss. Furthermore, a reduction in training time can possibly be achieved by reducing the precision of the weights in the network. In terms of PC color compression, compacting the input of the network from four channels to three channels by mapping the geometry information to the color information can also achieve a reduction in training and inference time.

- Conduct a study of the proposed PCC solutions considering subjective PCC metrics, with the objective of better understanding their real world performance.

- Regarding the compression of both geometry and color PC information, the proposed solutions perform the compression of these two attributes simultaneously. Instead, the model can be separated into two models, where one is optimized to compress PC geometry information and the other is optimized to compress PC color information.

7 Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from `https://www.tensorflow.org/`.

[2] Gisle Bjontegaard. Calculation of average psnr differences between rd-curves. *VCEG-M33*, 2001.

[3] RECOMMENDATION ITU-R BT. Methodology for the subjective assessment of the quality of television pictures. *International Telecommunication Union*, 2002.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[5] Eugene d'Eon, Bob Harrison, Taos Myers, and Philip A Chou. 8i voxelized full bodies-a voxelized point cloud dataset. *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006*, 7:8, 2017.

[6] Nicolas Frank, Davi Lazzarotto, and Touradj Ebrahimi. Latent space slicing for enhanced entropy modeling in learning-based point cloud geometry compression. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4878–4882. IEEE, 2022. Software available from `https://github.com/mmspg/pcc-geo-slicing.git`.

[7] Chunyang Fu, Ge Li, Rui Song, Wei Gao, and Shan Liu. Octattention: Octree-based large-scale contexts model for point cloud compression. *arXiv preprint arXiv:2202.06028*, 2022.

[8] Moving Picture Experts Group. Mpeg point cloud compression distortion metric. Software available from `http://mpegx.int-evry.fr/software/MPEG/PCC/mpeg-pcc-dmetric`.

[9] André FR Guarda, Nuno MM Rodrigues, and Fernando Pereira. Point cloud coding: Adopting a deep learning-based approach. In *2019 Picture Coding Symposium (PCS)*, pages 1–5. IEEE, 2019.

[10] André FR Guarda, Nuno MM Rodrigues, and Fernando Pereira. Adaptive deep learning-based point cloud geometry coding. *IEEE Journal of Selected Topics in Signal Processing*, 15(2):415–430, 2020. Software available from `https://github.com/aguarda/ADLPCC.git`.

[11] André FR Guarda, Nuno MM Rodrigues, and Fernando Pereira. Deep learning-based point cloud geometry coding with resolution scalability. In *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE, 2020.

[12] André FR Guarda, Nuno MM Rodrigues, and Fernando Pereira. Neighborhood adaptive loss function for deep learning-based point cloud coding with implicit and explicit quantization. *IEEE MultiMedia*, 28(3):107–116, 2020.

[13] André FR Guarda, Nuno MM Rodrigues, and Fernando Pereira. Point cloud geometry scalable coding with a single end-to-end deep learning model. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 3354–3358. IEEE, 2020.

[14] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras and Tensorflow*. O'Reilly Media, 2019.

[15] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[17] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[18] Gabriel Meynet, Yana Nehmé, Julie Digne, and Guillaume Lavoué. Pcqm: A full-reference quality metric for colored 3d point clouds. In *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6. IEEE, 2020.

[19] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[20] Maurice Quach, Giuseppe Valenzise, and Frederic Dufaux. Improved deep point cloud geometry compression, 2020. Software available from `https://github.com/mauriceqch/pcc_geo_cnn_v2.git`.

[21] Anh H. Reynolds. Large-scale image recognition: Alexnet. Blog. Available from `https://anhreynolds.com/blogs/alexnet.html`.

[22] Raul Rojas. The backpropagation algorithm. In *Neural networks*, pages 149–182. Springer, 1996.

[23] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE international conference on robotics and automation*, pages 1–4. IEEE, 2011.

[24] Ruwen Schnabel and Reinhard Klein. Octree-based Point-Cloud Compression. In Mario Botsch, Baoquan Chen, Mark Pauly, and Matthias Zwicker, editors, *Symposium on Point-Based Graphics*. The Eurographics Association, 2006.

[25] Sebastian Schwarz, Gaëlle Martin-Cocher, David Flynn, and Madhukar Budagavi. Common test conditions for point cloud compression. *Document ISO/IEC JTC1/SC29/WG11 w17766, Ljubljana, Slovenia*, 2018.

[26] Sebastian Schwarz, Marius Preda, Vittorio Baroncini, Madhukar Budagavi, Pablo Cesar, Philip A. Chou, Robert A. Cohen, Maja Krivokuća, Sébastien Lasserre, Zhu Li, Joan Llach, Khaled Mammou, Rufael Mekuria, Ohji Nakagami, Ernestasia Siahaan, Ali Tabatabai, Alexis M. Tourapis, and Vladyslav Zakharchenko. Emerging mpeg standards for point cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(1):133–148, 2019.

[27] Dong Tian, Hideaki Ochimizu, Chen Feng, Robert Cohen, and Anthony Vetro. Geometric distortion metrics for point cloud compression. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3460–3464. IEEE, 2017.

[28] Asher Trockman and J Zico Kolter. Patches are all you need? *arXiv preprint arXiv:2201.09792*, 2022.

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[30] Jianqiang Wang, Dandan Ding, Zhu Li, and Zhan Ma. Multiscale point cloud geometry compression. In *2021 Data Compression Conference (DCC)*, pages 73–82. IEEE, 2021.

[31] Ian H Witten, Radford M Neal, and John G Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540, 1987.

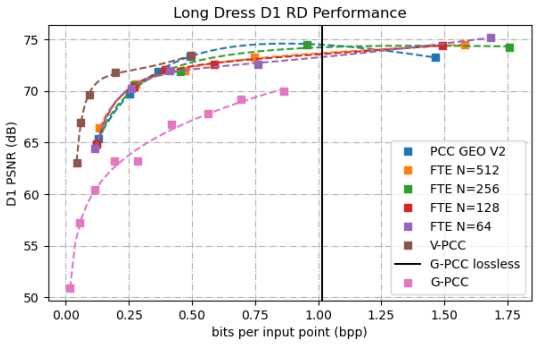
[32] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[33] Wei Yan, Shan Liu, Thomas H Li, Zhu Li, Ge Li, et al. Deep autoencoder-based lossy geometry compression for point clouds. *arXiv preprint arXiv:1905.03691*, 2019.

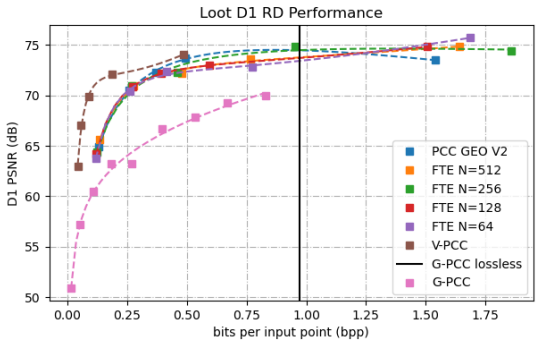
Appendix A

Proposed End-to-End AE-based PC Geometry Coding Results

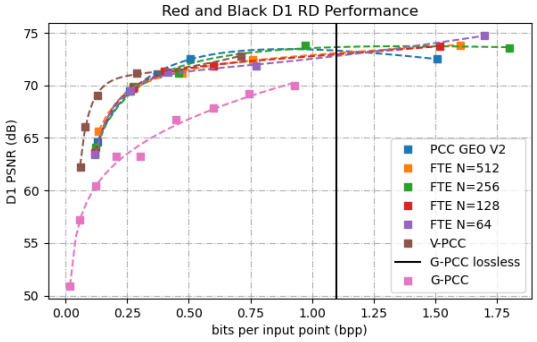
In Appendix A, the results for the proposed FTE, PTE, FCME and PCME are presented. They include RD performance graphs with the relevant metrics for every test PC and tables that present the BD gains compared to the relevant baselines.



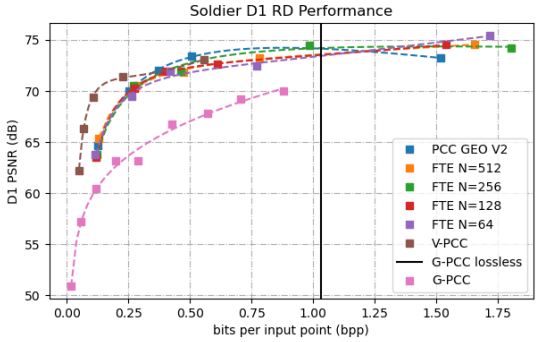
(a) Long Dress.



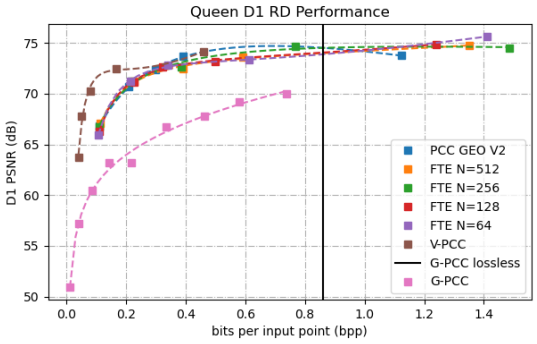
(b) Loot.



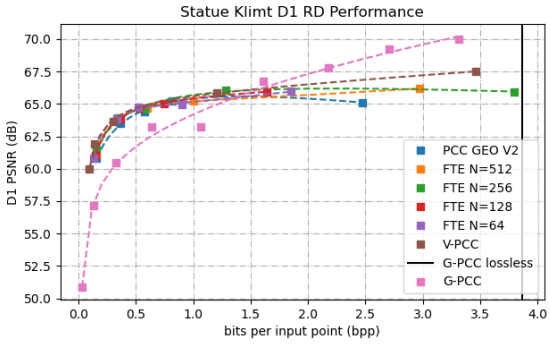
(c) Red and Black.



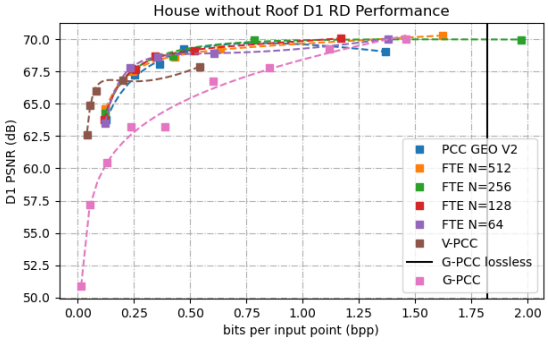
(d) Soldier.



(e) Queen.

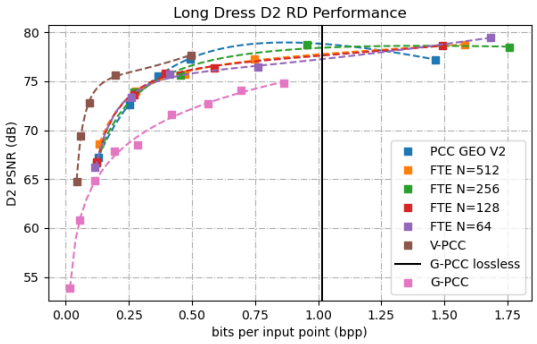


(f) Statue Klimt.

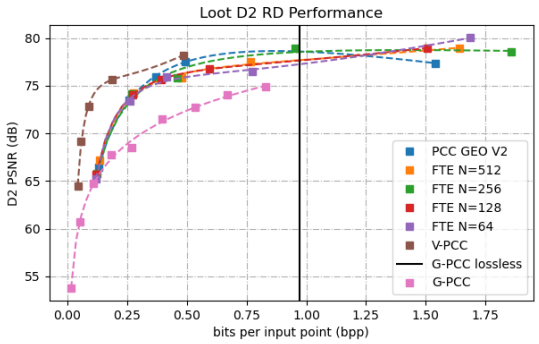


(g) House without Roof.

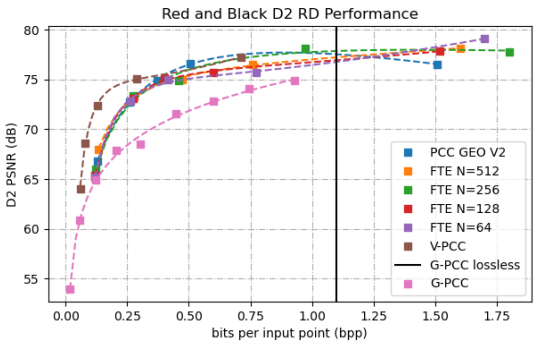
Figure A.1: D1 PSNR RD performance graphs for the FTE.



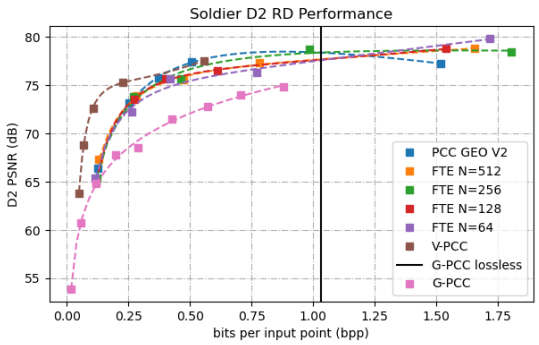
(a) Long Dress.



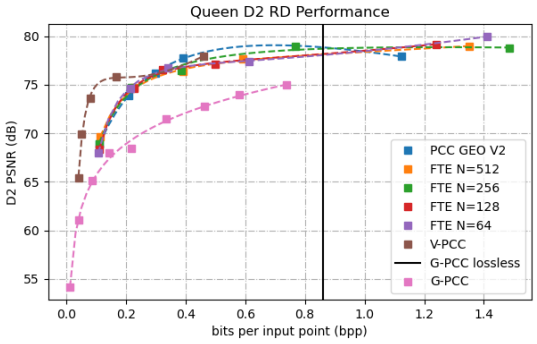
(b) Loot.



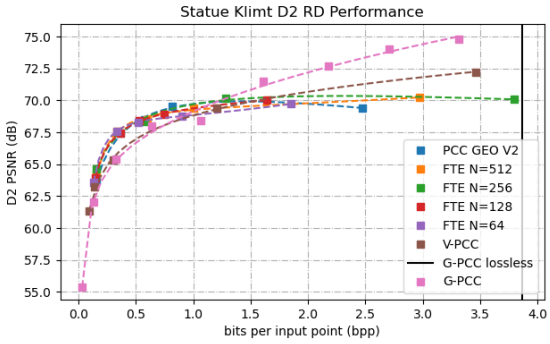
(c) Red and Black.



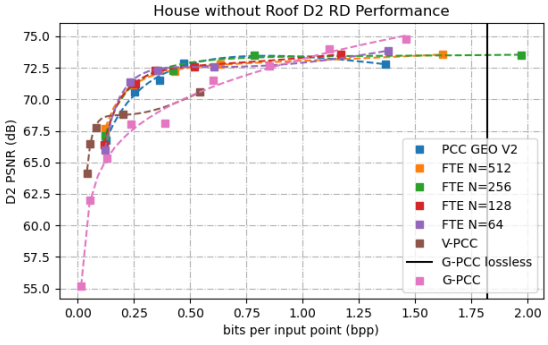
(d) Soldier.



(e) Queen.



(f) Statue Klimt.



(g) House without Roof.

Figure A.2: D2 PSNR RD performance graphs for the FTE.

Table A.1: BD metrics for the FTE with N feature map channels compared to the G-PCC encoder. The BD-rate is in percentage and the BD-PSNR is in dB, with the best values in bold.

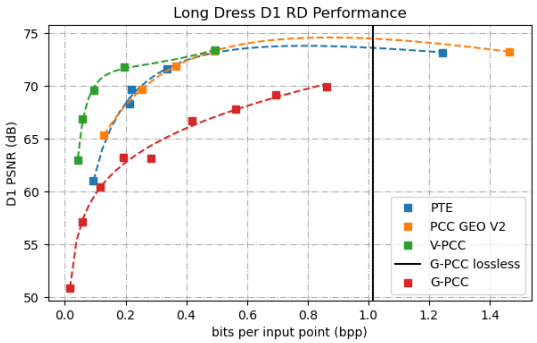
			N			
Point Cloud	**Metric**	**BD**	***512***	***256***	***128***	***64***
Long Dress	D1	rate	-250.36	-181.14	-229.17	**-267.91**
		PSNR	**5.48**	5.43	5.43	5.27
	D2	rate	**-129.21**	-87.66	-109.64	-120.64
		PSNR	**3.96**	3.83	3.85	3.63
Loot	D1	rate	-216.44	-154.80	-203.50	**-240.34**
		PSNR	5.36	**5.41**	5.34	5.14
	D2	rate	**-99.28**	-66.78	-86.51	-97.53
		PSNR	3.70	**3.71**	3.62	3.34
Red and Black	D1	rate	-218.98	-165.77	-191.79	**-225.94**
		PSNR	4.87	**4.90**	4.79	4.66
	D2	rate	**-115.18**	-79.44	-86.33	-103.31
		PSNR	**3.50**	3.45	3.29	3.12
Soldier	D1	rate	**-223.38**	-158.25	-191.63	-211.60
		PSNR	**5.30**	5.30	5.17	4.88
	D2	rate	**-109.67**	-71.97	-82.58	-83.40
		PSNR	**3.79**	3.74	3.58	3.10
Queen	D1	rate	-287.70	-235.65	-274.75	**-351.03**
		PSNR	5.76	**5.95**	5.80	5.86
	D2	rate	-161.88	-128.53	-145.34	**-172.70**
		PSNR	4.38	**4.55**	4.38	4.46
Statue Klimt	D1	rate	-106.11	-110.38	-127.03	**-139.80**
		PSNR	1.39	1.43	**2.38**	2.33
	D2	rate	-45.06	-44.31	-50.37	**-59.61**
		PSNR	0.31	0.31	**1.08**	0.96
House without Roof	D1	rate	-167.64	-184.14	**-187.31**	-183.19
		PSNR	3.30	3.46	**3.69**	3.32
	D2	rate	**-101.28**	-96.71	-97.33	-92.60
		PSNR	1.97	2.11	**2.27**	2.04

Table A.2: BD metrics for the FTE with N feature map channels compared to the V-PCC encoder. The BD-rate is in percentage and the BD-PSNR is in dB, with the best values in bold.

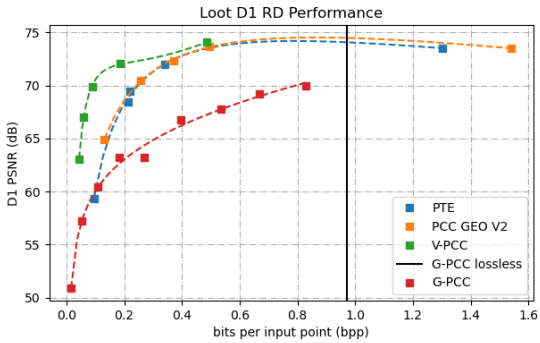
			N			
Point Cloud	**Metric**	**BD**	***512***	***256***	***128***	***64***
Long Dress	D1	rate	**54.28**	57.22	55.56	54.52
		PSNR	**-2.14**	-2.57	-2.36	-2.50
	D2	rate	**56.13**	58.57	57.18	56.45
		PSNR	**-2.93**	-3.49	-3.22	-3.39
Loot	D1	rate	58.55	59.84	58.75	**57.47**
		PSNR	**-2.54**	-2.89	-2.74	-2.91
	D2	rate	59.55	60.50	59.52	**58.44**
		PSNR	**-3.39**	-3.82	-3.65	-3.87
Red and Black	D1	rate	39.59	42.61	42.01	**39.32**
		PSNR	**-1.43**	-1.59	-1.64	-1.76
	D2	rate	**41.38**	44.34	44.54	41.39
		PSNR	**-1.93**	-2.20	-2.26	-2.40
Soldier	D1	rate	**47.11**	51.01	49.72	50.49
		PSNR	**-1.76**	-2.09	-2.07	-2.38
	D2	rate	**50.56**	53.01	52.43	54.42
		PSNR	**-2.65**	-3.05	-3.05	-3.56
Queen	D1	rate	53.14	52.25	53.30	**49.49**
		PSNR	-1.86	-1.87	-1.89	**-1.81**
	D2	rate	49.12	50.11	50.79	**47.53**
		PSNR	**-1.90**	-1.98	-1.99	-1.91
Statue Klimt	D1	rate	21.98	17.87	19.07	**15.64**
		PSNR	-0.46	-0.30	-0.29	**-0.27**
	D2	rate	-42.49	-44.28	-44.96	**-50.54**
		PSNR	0.60	0.64	**0.98**	0.90
House without Roof	D1	rate	**20.65**	38.60	39.88	35.25
		PSNR	0.42	**0.51**	0.38	0.48
	D2	rate	**-63.03**	-3.66	-2.63	-4.06
		PSNR	1.79	**1.82**	1.64	1.81

Table A.3: BD metrics for the FTE with N feature map channels compared to the PCC GEO V2 encoder. The BD-rate is in percentage and the BD-PSNR is in dB, with the best values in bold.

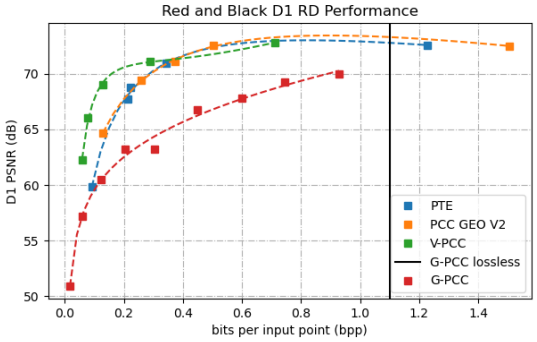
			N			
Point Cloud	**Metric**	**BD**	***512***	***256***	***128***	***64***
Long Dress	D1	rate	-1.02	-0.84	-1.95	**-3.57**
		PSNR	-0.22	**-0.14**	-0.24	-0.40
	D2	rate	-2.43	-1.35	-2.82	**-3.24**
		PSNR	-0.24	**-0.17**	-0.28	-0.48
Loot	D1	rate	2.62	2.31	0.16	**-3.07**
		PSNR	-0.33	**-0.11**	-0.28	-0.43
	D2	rate	2.26	2.17	0.09	**-2.78**
		PSNR	-0.39	**-0.12**	-0.33	-0.54
Red and Black	D1	rate	1.57	1.42	1.26	**-2.75**
		PSNR	-0.23	**-0.08**	-0.26	-0.37
	D2	rate	0.66	2.06	2.94	**-2.05**
		PSNR	-0.25	**-0.13**	-0.38	-0.51
Soldier	D1	rate	4.33	4.47	**4.22**	6.57
		PSNR	-0.37	**-0.21**	-0.40	-0.63
	D2	rate	**3.64**	4.29	4.19	8.64
		PSNR	-0.42	**-0.25**	-0.47	-0.83
Queen	D1	rate	5.59	0.99	2.14	**-6.82**
		PSNR	-0.36	**-0.11**	-0.30	-0.26
	D2	rate	5.04	1.14	2.13	**-5.71**
		PSNR	-0.41	**-0.15**	-0.36	-0.28
Statue Klimt	D1	rate	-27.88	-29.13	-25.97	**-34.70**
		PSNR	0.27	**0.49**	0.35	0.38
	D2	rate	-10.92	-11.23	-9.24	**-12.83**
		PSNR	0.11	**0.27**	0.07	0.04
House without Roof	D1	rate	-19.43	-18.14	-19.51	**-20.84**
		PSNR	0.27	**0.44**	0.34	0.22
	D2	rate	-15.97	-12.77	-15.91	**-20.90**
		PSNR	0.28	**0.43**	0.29	0.29



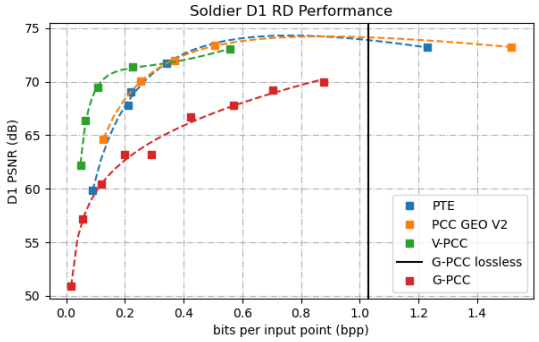
(a) Long Dress.



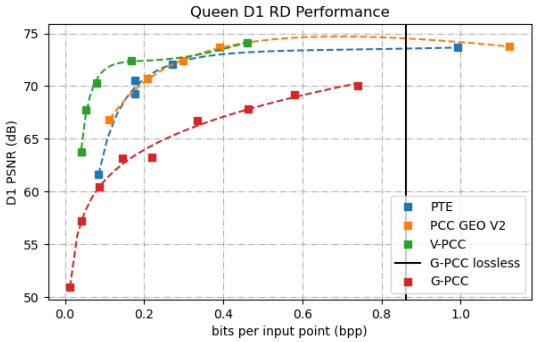
(b) Loot.



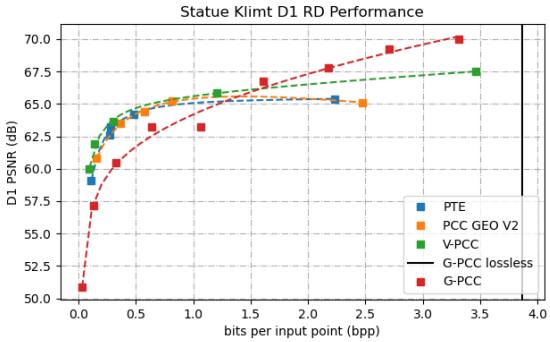
(c) Red and Black.



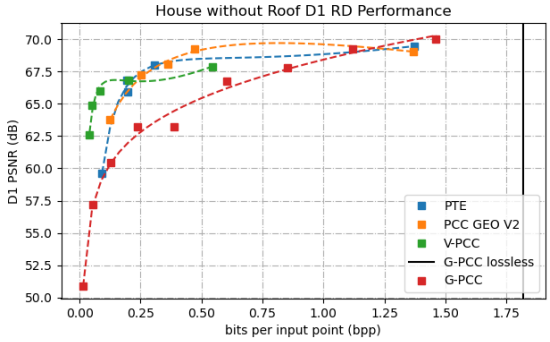
(d) Soldier.



(e) Queen.

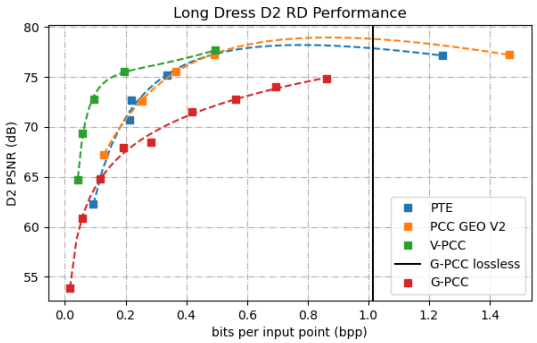


(f) Statue Klimt.

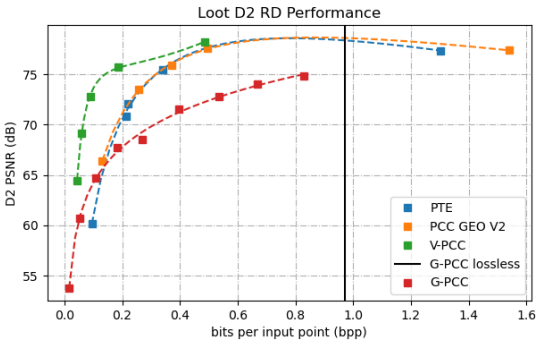


(g) House without Roof.

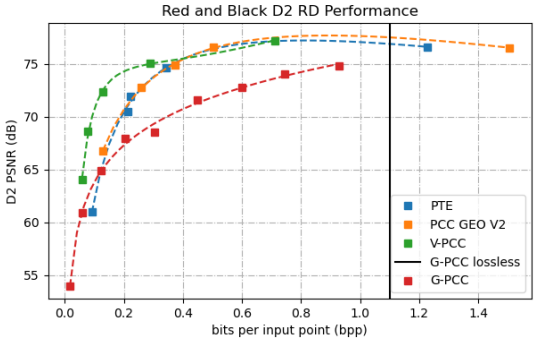
Figure A.3: D1 PSNR RD performance graphs for the PTE.



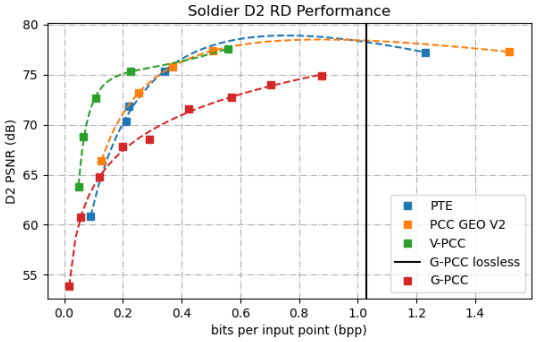
(a) Long Dress.



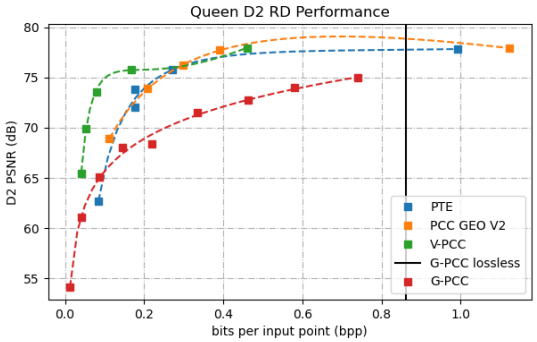
(b) Loot.



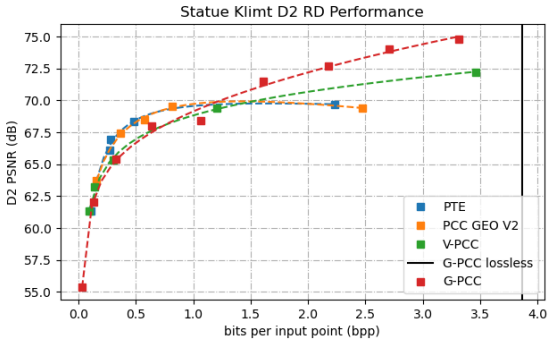
(c) Red and Black.



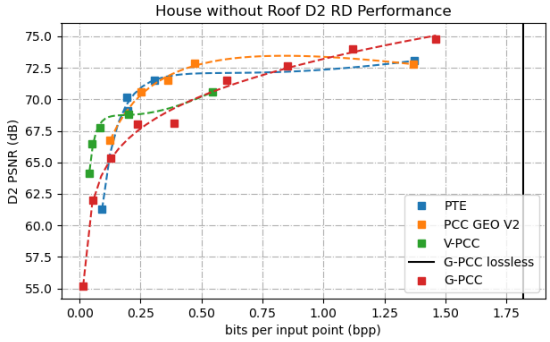
(d) Soldier.



(e) Queen.



(f) Statue Klimt.

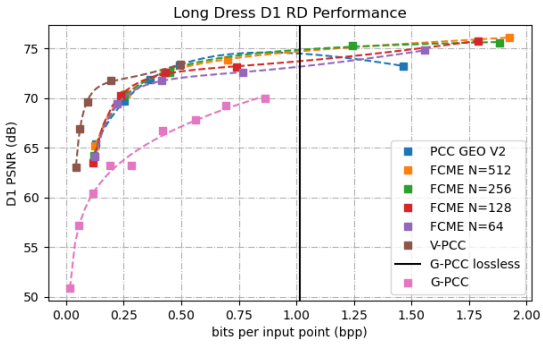


(g) House without Roof.

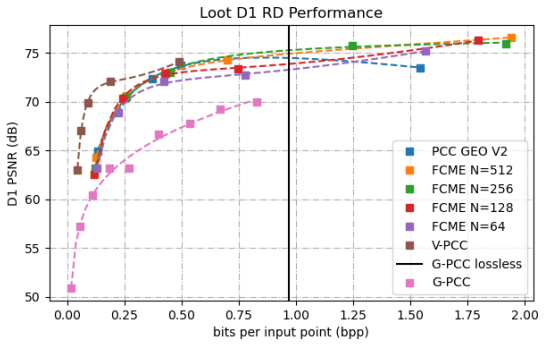
Figure A.4: D2 PSNR RD performance graphs for the PTE.

Table A.4: BD metrics for the PTE compared to the G-PCC, V-PCC and PCC GEO V2 encoders. The BD-rate is in percentage and the BD-PSNR is in dB.

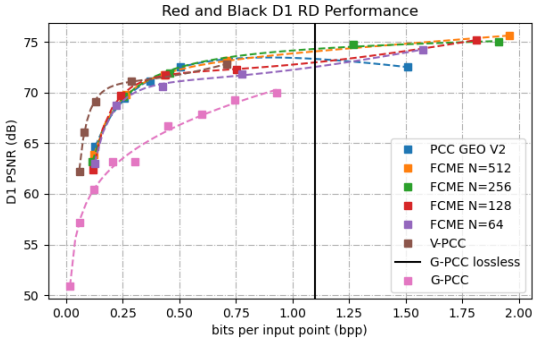
			Encoder		
Point Cloud	**Metric**	**BD**	***G-PCC***	***V-PCC***	***PCC GEO V2***
Long Dress	D1	rate	-141.91	58.39	-3.40
		PSNR	5.13	-3.40	-0.25
	D2	rate	-51.17	59.61	-2.32
		PSNR	3.43	-4.51	-0.22
Loot	D1	rate	-93.29	61.02	2.02
		PSNR	4.81	-4.10	-0.30
	D2	rate	-20.88	61.77	3.73
		PSNR	2.96	-5.30	-0.25
Red and Black	D1	rate	-117.10	43.70	-2.17
		PSNR	4.52	-2.27	-0.23
	D2	rate	-38.16	44.91	-1.27
		PSNR	2.91	-3.00	-0.29
Soldier	D1	rate	-117.00	52.95	5.24
		PSNR	4.92	-3.06	-0.24
	D2	rate	-34.27	55.01	6.45
		PSNR	3.22	-4.25	-0.23
Queen	D1	rate	-138.53	58.10	-6.09
		PSNR	5.32	-2.91	-0.50
	D2	rate	-49.57	55.73	-5.81
		PSNR	3.71	-3.31	-0.54
Statue Klimt	D1	rate	-105.06	33.75	-7.95
		PSNR	1.70	-0.75	0.00
	D2	rate	-32.33	-21.76	-9.30
		PSNR	0.65	0.66	0.06
House without Roof	D1	rate	-100.66	49.37	-0.63
		PSNR	2.72	-0.85	-0.26
	D2	rate	-13.86	37.11	-1.77
		PSNR	1.15	0.18	-0.34



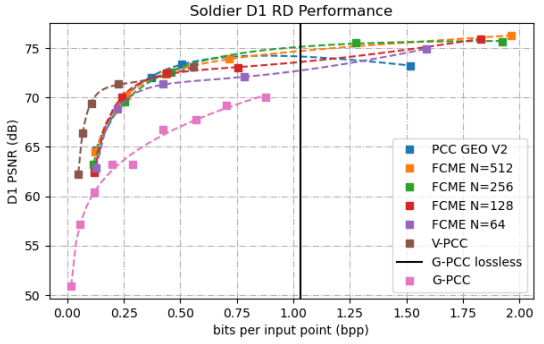
(a) Long Dress.



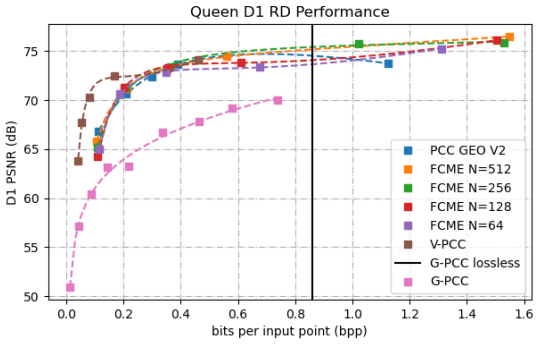
(b) Loot.



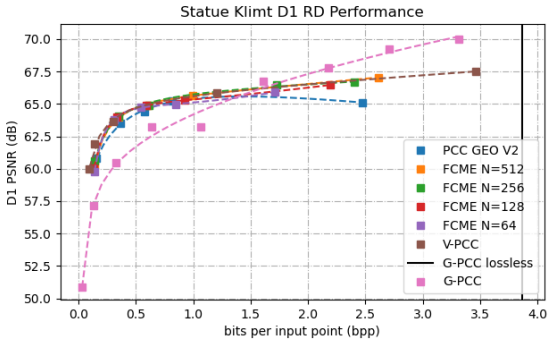
(c) Red and Black.



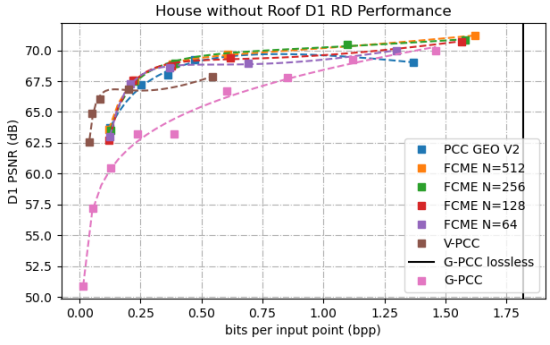
(d) Soldier.



(e) Queen.

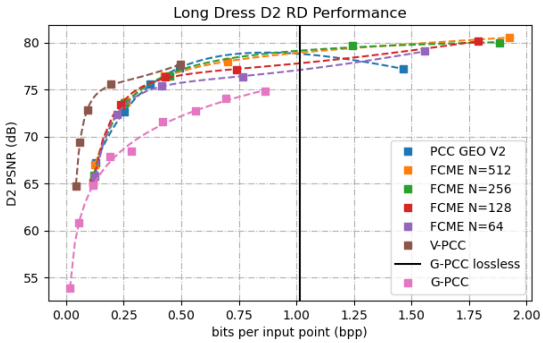


(f) Statue Klimt.

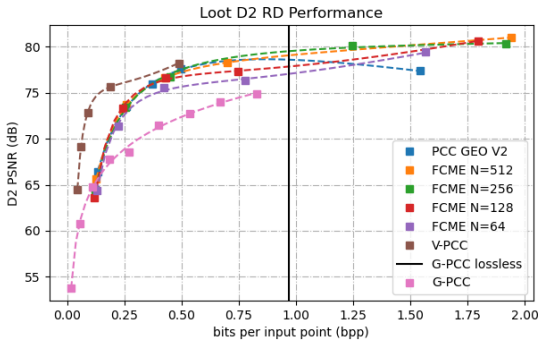


(g) House without Roof.

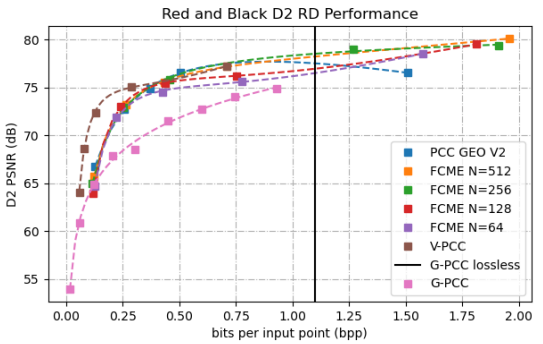
Figure A.5: D1 PSNR RD performance graphs for the FCME.



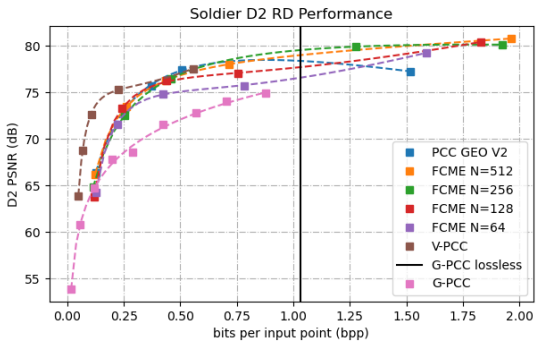
(a) Long Dress.



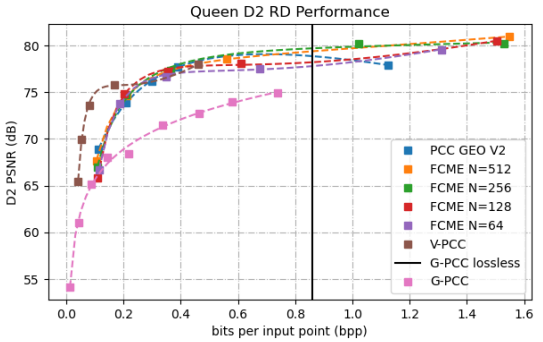
(b) Loot.



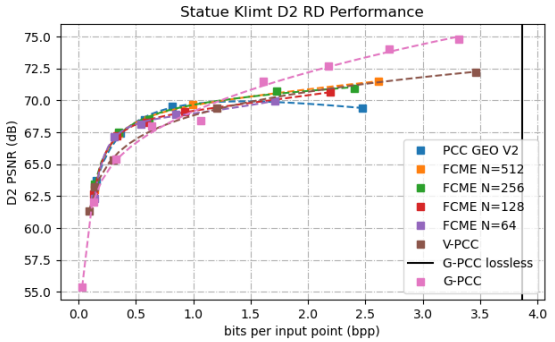
(c) Red and Black.



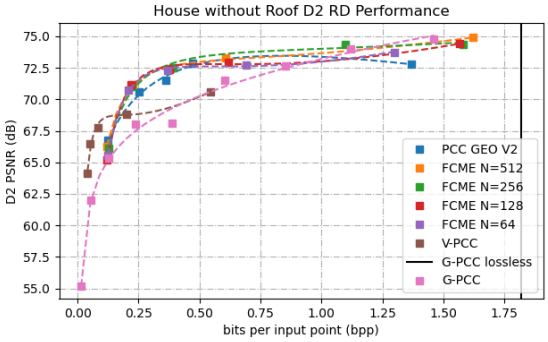
(d) Soldier.



(e) Queen.



(f) Statue Klimt.



(g) House without Roof.

Figure A.6: D2 PSNR RD performance graphs for the FCME.

Table A.5: BD metrics for the FCME with N feature map channels compared to the G-PCC encoder. The BD-rate is in percentage and the BD-PSNR is in dB, with the best values in bold., with the best values in bold.

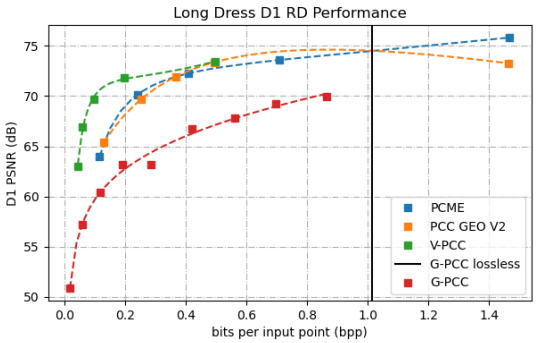
			N			
Point Cloud	**Metric**	**BD**	***512***	***256***	***128***	***64***
Long Dress	D1	rate	-225.81	-178.78	**-251.17**	-232.35
		PSNR	**5.77**	5.67	5.54	5.07
	D2	rate	-110.53	-82.91	**-112.26**	-101.92
		PSNR	**4.27**	4.13	3.97	3.42
Loot	D1	rate	-193.40	-135.82	**-203.54**	-175.68
		PSNR	**5.65**	5.53	5.34	4.70
	D2	rate	**-81.08**	-50.99	-77.32	-59.68
		PSNR	**4.00**	3.85	3.63	2.77
Red and Black	D1	rate	-201.88	-163.50	**-222.83**	-195.00
		PSNR	**5.17**	5.16	4.90	4.34
	D2	rate	**-97.92**	-75.94	-97.62	-84.15
		PSNR	**3.73**	3.69	3.39	2.86
Soldier	D1	rate	-211.29	-150.20	**-219.29**	-198.18
		PSNR	**5.61**	5.32	5.28	4.54
	D2	rate	**-99.86**	-62.78	-91.50	-75.93
		PSNR	**4.12**	3.70	3.71	2.76
Queen	D1	rate	-268.51	-190.06	**-282.82**	-282.37
		PSNR	6.25	**6.27**	6.04	5.74
	D2	rate	**-139.73**	-92.50	-127.48	-126.59
		PSNR	4.85	**4.88**	4.67	4.33
Statue Klimt	D1	rate	-107.56	-115.87	-123.65	**-126.42**
		PSNR	1.93	2.09	2.09	**2.26**
	D2	rate	-34.27	**-38.63**	-33.40	-35.18
		PSNR	0.67	**0.77**	0.61	0.76
House without Roof	D1	rate	**-210.02**	-194.37	-184.84	-169.33
		PSNR	**3.62**	3.58	3.34	3.31
	D2	rate	**-96.19**	-89.47	-79.41	-80.41
		PSNR	2.25	**2.32**	1.96	2.04

Table A.6: BD metrics for the FCME with N feature map channels compared to the V-PCC encoder. The BD-rate is in percentage and the BD-PSNR is in dB, with the best values in bold.

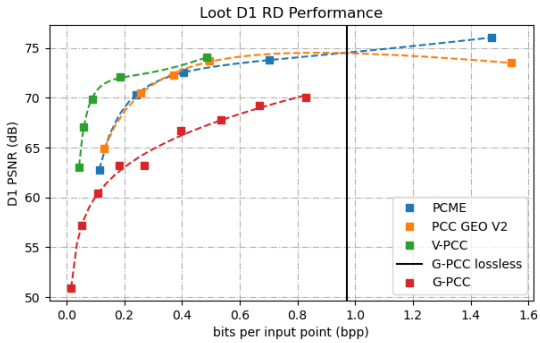
			N			
Point Cloud	**Metric**	**BD**	***512***	***256***	***128***	***64***
Long Dress	D1	rate	53.06	56.16	**51.83**	57.45
		PSNR	**-2.17**	-2.48	-2.35	-2.62
	D2	rate	54.60	57.10	**53.79**	58.97
		PSNR	**-2.98**	-3.34	-3.20	-3.53
Loot	D1	rate	56.48	59.58	**55.36**	62.73
		PSNR	**-2.63**	-2.99	-2.86	-3.33
	D2	rate	57.30	59.89	**56.32**	63.71
		PSNR	**-3.53**	-3.95	-3.79	-4.48
Red and Black	D1	rate	37.48	40.77	**35.08**	44.42
		PSNR	**-1.32**	-1.44	-1.52	-1.99
	D2	rate	39.28	42.07	**37.74**	45.43
		PSNR	**-1.90**	-2.05	-2.15	-2.59
Soldier	D1	rate	45.63	51.71	**44.74**	52.65
		PSNR	**-1.72**	-2.30	-1.97	-2.50
	D2	rate	48.37	53.33	**47.80**	55.67
		PSNR	**-2.62**	-3.37	-2.92	-3.64
Queen	D1	rate	**49.20**	53.13	49.46	53.40
		PSNR	-1.70	-1.74	**-1.68**	-1.82
	D2	rate	**47.95**	52.72	49.02	52.43
		PSNR	-1.81	-1.87	**-1.75**	-1.94
Statue Klimt	D1	rate	13.67	**13.50**	16.42	22.32
		PSNR	-0.19	**-0.13**	-0.28	-0.39
	D2	rate	-35.77	**-38.43**	-28.55	-25.90
		PSNR	0.83	**0.88**	0.65	0.67
House without Roof	D1	rate	**36.10**	43.71	42.51	43.68
		PSNR	0.55	**0.60**	0.43	0.44
	D2	rate	**-15.12**	17.59	15.16	17.01
		PSNR	1.82	**1.85**	1.71	1.72

Table A.7: BD metrics for the FCME with N feature map channels compared to the PCC GEO V2 encoder. The BD-rate is in percentage and the BD-PSNR is in dB, with the best values in bold.

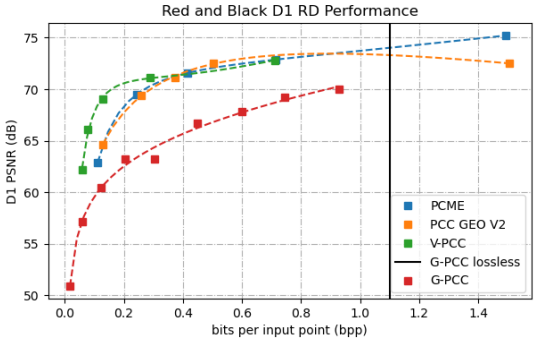
			N			
Point Cloud	**Metric**	**BD**	***512***	***256***	***128***	***64***
Long Dress	D1	rate	-12.60	-9.25	**-13.79**	2.49
		PSNR	**0.26**	0.24	-0.06	-0.59
	D2	rate	-12.75	-9.62	**-13.30**	2.27
		PSNR	**0.31**	0.29	-0.05	-0.71
Loot	D1	rate	-8.11	-3.84	**-9.14**	9.67
		PSNR	0.24	**0.28**	-0.13	-0.83
	D2	rate	-7.83	-3.82	**-8.94**	10.57
		PSNR	0.30	**0.35**	-0.13	-1.09
Red and Black	D1	rate	-9.86	-6.88	**-11.85**	6.18
		PSNR	0.25	**0.32**	-0.11	-0.70
	D2	rate	-8.82	-5.60	**-9.34**	4.99
		PSNR	0.21	**0.29**	-0.22	-0.81
Soldier	D1	rate	-6.16	1.02	**-7.30**	12.02
		PSNR	**0.17**	0.08	-0.25	-1.05
	D2	rate	-5.90	1.57	**-6.65**	12.59
		PSNR	**0.21**	0.06	-0.28	-1.34
Queen	D1	rate	**-12.71**	-9.47	-12.05	-0.47
		PSNR	0.30	**0.34**	-0.07	-0.42
	D2	rate	**-11.46**	-7.78	-10.35	0.56
		PSNR	0.30	**0.34**	-0.08	-0.49
Statue Klimt	D1	rate	-32.41	-31.01	**-35.32**	-26.27
		PSNR	0.55	**0.59**	0.44	0.29
	D2	rate	**-10.63**	-9.88	-4.61	-1.23
		PSNR	**0.25**	0.24	-0.05	-0.17
House without Roof	D1	rate	**-31.18**	-23.13	-26.50	-15.44
		PSNR	**0.59**	0.56	0.33	0.12
	D2	rate	**-31.77**	-19.23	-22.81	-14.54
		PSNR	0.56	**0.62**	0.29	0.19



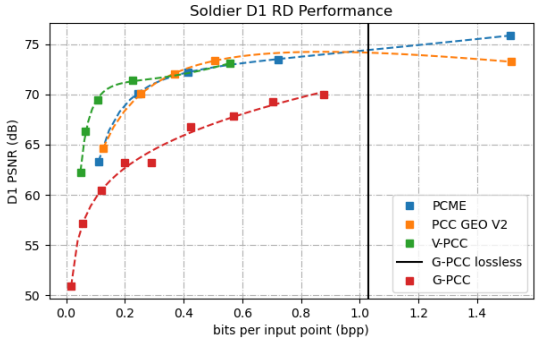
(a) Long Dress.



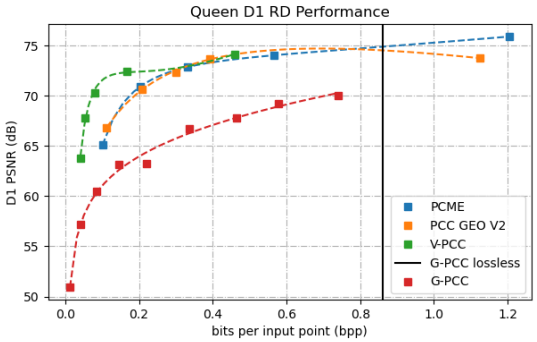
(b) Loot.



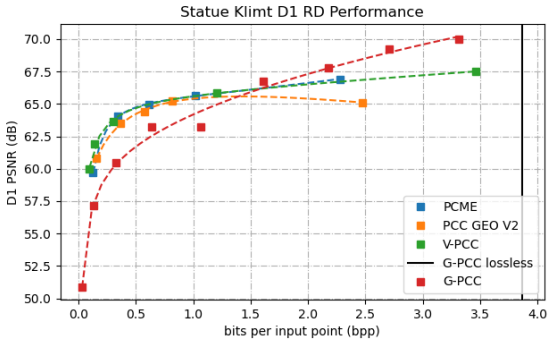
(c) Red and Black.



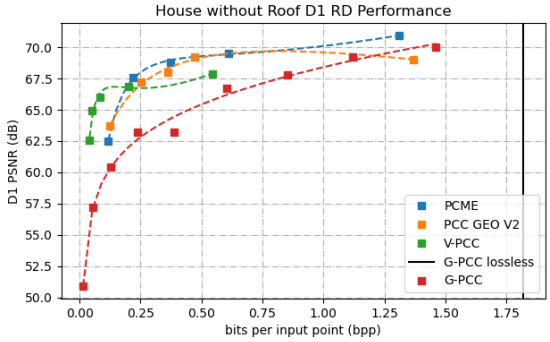
(d) Soldier.



(e) Queen.

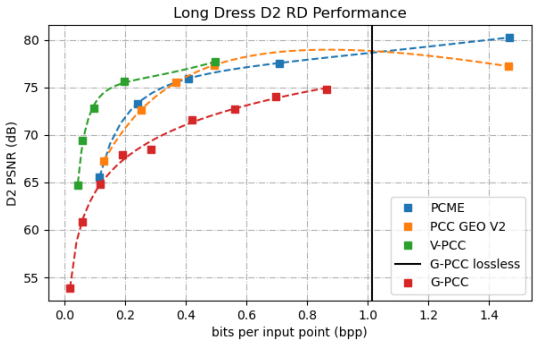


(f) Statue Klimt.

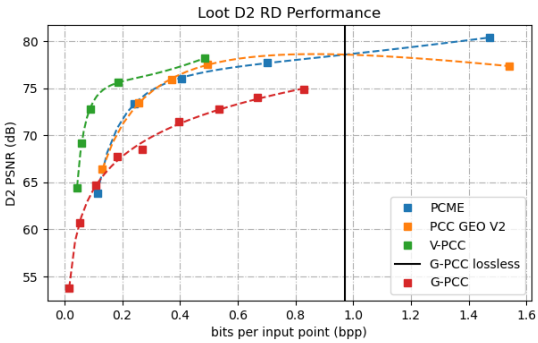


(g) House without Roof.

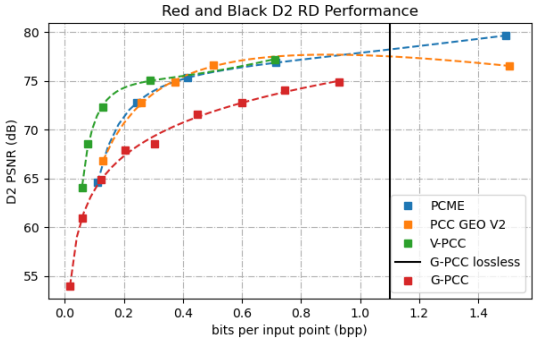
Figure A.7: D1 PSNR RD performance graphs for the PCME.



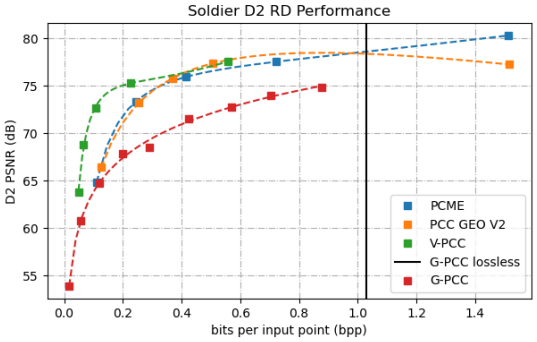
(a) Long Dress.



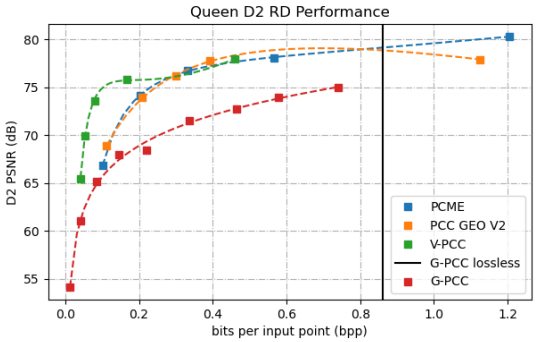
(b) Loot.



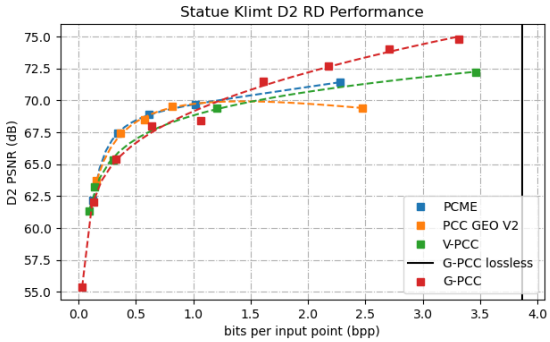
(c) Red and Black.



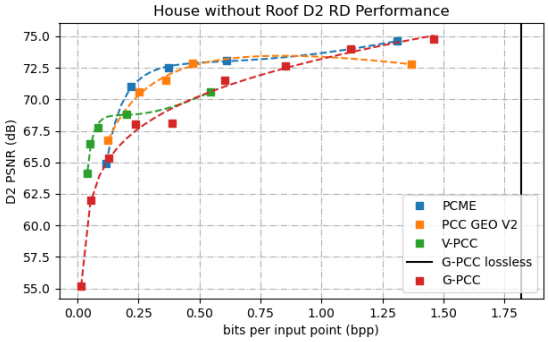
(d) Soldier.



(e) Queen.



(f) Statue Klimt.



(g) House without Roof.

Figure A.8: D2 PSNR RD performance graphs for the PCME.

Table A.8: BD metrics for the PCME compared to the G-PCC, V-PCC and PCC GEO V2 encoders. The BD-rate is in percentage and the BD-PSNR is in dB.

			Encoder		
Point Cloud	**Metric**	**BD**	***G-PCC***	***V-PCC***	***PCC GEO V2***
Long Dress	D1	rate	-227.08	53.58	-12.59
		PSNR	5.60	-2.42	0.16
	D2	rate	-103.96	55.01	-12.60
		PSNR	4.01	-3.29	0.19
Loot	D1	rate	-182.65	56.81	-7.76
		PSNR	5.41	-2.94	0.09
	D2	rate	-70.26	57.37	-7.69
		PSNR	3.67	-3.90	0.10
Red and Black	D1	rate	-201.02	36.99	-11.32
		PSNR	5.09	-1.48	0.27
	D2	rate	-92.28	38.90	-9.69
		PSNR	3.59	-2.11	0.23
Soldier	D1	rate	-218.21	44.73	-8.98
		PSNR	5.52	-1.88	0.15
	D2	rate	-98.97	47.20	-8.88
		PSNR	3.98	-2.79	0.17
Queen	D1	rate	-257.78	51.59	-7.93
		PSNR	6.00	-1.97	0.09
	D2	rate	-121.56	51.02	-6.08
		PSNR	4.51	-2.20	0.04
Statue Klimt	D1	rate	-118.40	11.65	-36.07
		PSNR	2.16	-0.20	0.56
	D2	rate	-39.08	-37.58	-15.40
		PSNR	0.89	0.95	0.33
House without Roof	D1	rate	-190.49	44.18	-28.01
		PSNR	3.65	0.41	0.50
	D2	rate	-80.16	25.53	-24.56
		PSNR	2.25	1.62	0.47

Table A.9: Summary of BD metrics for the proposed encoders compared to the PCC GEO V2 encoder. The BD-rate is in percentage and the BD-PSNR is in dB with the best values in bold. For the FTE, the 256 channel model was chosen as its overall best. For the FCME, the 512 channel model was chosen as its overall best.

			Encoder			
Point Cloud	**Metric**	**BD**	***FTE***	***PTE***	***FCME***	***PCME***
Long Dress	D1	rate	-0.84	-3.40	**-12.60**	-12.59
		PSNR	-0.14	-0.25	**0.26**	0.16
	D2	rate	-1.35	-2.32	**-12.75**	-12.60
		PSNR	-0.17	-0.22	**0.31**	0.19
Loot	D1	rate	2.31	2.02	**-8.11**	-7.76
		PSNR	-0.11	-0.30	**0.24**	0.09
	D2	rate	2.17	3.73	**-7.83**	-7.69
		PSNR	-0.12	-0.25	**0.30**	0.10
Red and Black	D1	rate	1.42	-2.17	-9.86	**-11.32**
		PSNR	-0.08	-0.23	0.25	**0.27**
	D2	rate	2.06	-1.27	-8.82	**-9.69**
		PSNR	-0.13	-0.29	0.21	**0.23**
Soldier	D1	rate	4.47	5.24	-6.16	**-8.98**
		PSNR	-0.21	-0.24	**0.17**	0.15
	D2	rate	4.29	6.45	-5.90	**-8.88**
		PSNR	-0.25	-0.23	**0.21**	0.17
Queen	D1	rate	0.99	-6.09	**-12.71**	-7.93
		PSNR	-0.11	-0.50	**0.30**	0.09
	D2	rate	1.14	-5.81	**-11.46**	-6.08
		PSNR	-0.15	-0.54	**0.30**	0.04
Statue Klimt	D1	rate	-29.13	-7.95	-32.41	**-36.07**
		PSNR	0.49	0.00	0.55	**0.56**
	D2	rate	-11.23	-9.30	-10.63	**-15.40**
		PSNR	0.27	0.06	0.25	**0.33**
House without Roof	D1	rate	-18.14	-0.63	**-31.18**	-28.01
		PSNR	0.44	-0.26	**0.59**	0.50
	D2	rate	-12.77	-1.77	**-31.77**	-24.56
		PSNR	0.43	-0.34	**0.56**	0.47

Appendix B

Color conversion transformations

B.1 YUV

From the RGB color space, the YUV color space is defined by eq. B.1.

$$\begin{aligned} Y &= \frac{0.21260 \cdot R}{255} + \frac{0.71520 \cdot G}{255} + \frac{0.07220 \cdot B}{255} \\ U &= -\frac{0.09991 \cdot R}{255} - \frac{0.33609 \cdot G}{255} + \frac{0.43600 \cdot B}{255} \\ V &= \frac{0.61500 \cdot R}{255} - \frac{0.55861 \cdot G}{255} - \frac{0.05639 \cdot B}{255} \end{aligned} \tag{B.1}$$

B.2 YCbCr

From the RGB color space, the YCbCr color space is defined by eq. B.2. The inverse transformation is defined by eq. B.3.

$$\begin{aligned} Y &= 16 + \frac{65.738 \cdot R}{256} + \frac{129.057 \cdot G}{256} + \frac{25.064 \cdot B}{256} \\ Cb &= 128 - \frac{37.945 \cdot R}{256} - \frac{74.494 \cdot G}{256} + \frac{112.439 \cdot B}{256} \\ Cr &= 128 + \frac{112.439 \cdot R}{256} - \frac{94.154 \cdot G}{256} - \frac{18.285 \cdot B}{256} \end{aligned} \tag{B.2}$$

$$\begin{aligned} R &= \frac{298.082 \cdot Y}{256} + \frac{408.583 \cdot Cr}{256} - 222.921 \\ G &= \frac{298.082 \cdot Y}{256} - \frac{100.291 \cdot Cb}{256} - \frac{208.120 \cdot Cr}{256} + 135.576 \\ B &= \frac{298.082 \cdot Y}{256} + \frac{516.412 \cdot Cb}{256} - 276.836 \end{aligned} \tag{B.3}$$

B.3 LAB

Typical RGB to LAB conversions use the intermediate XYZ color space to facilitate the conversion. From the RGB color space, the XYZ color space is defined by eq.B.6 with equations B.4 and B.5 presenting intermediate steps.

$$\begin{aligned} R' &= \frac{R}{255} \\ G' &= \frac{G}{255} \\ B' &= \frac{B}{255} \end{aligned} \tag{B.4}$$

$$\begin{aligned} R'' &= \begin{cases} (\frac{R'+0.055}{1.055})^{2.4}\,,\ R' > 0.04045 \\ \frac{R'}{12.92}\,,\ otherwise \end{cases} \\ G'' &= \begin{cases} (\frac{G'+0.055}{1.055})^{2.4}\,,\ G' > 0.04045 \\ \frac{G'}{12.92}\,,\ otherwise \end{cases} \\ B'' &= \begin{cases} (\frac{B'+0.055}{1.055})^{2.4}\,,\ B' > 0.04045 \\ \frac{B'}{12.92}\,,\ otherwise \end{cases} \end{aligned} \tag{B.5}$$

$$\begin{aligned} X &= 41.24 \cdot R'' + 35.76 \cdot G'' + 18.05 \cdot B'' \\ Y &= 21.26 \cdot R'' + 71.52 \cdot G'' + 7.22 \cdot B'' \\ Z &= 1.93 \cdot R'' + 11.92 \cdot G'' + 95.05 \cdot B'' \end{aligned} \tag{B.6}$$

From the XYZ color space, the LAB color space is defined by eq.B.9 with equations B.7 and B.8 presenting intermediate steps.

$$\begin{aligned} X' &= \frac{X}{95.047} \\ Y' &= \frac{Y}{100.000} \\ Z' &= \frac{Z}{108.883} \end{aligned} \tag{B.7}$$

$$\begin{aligned} X'' &= \begin{cases} \sqrt[3]{X'}\,,\ X' > 0.008856 \\ 7.787 \cdot X' + \frac{4}{29}\,,\ otherwise \end{cases} \\ Y'' &= \begin{cases} \sqrt[3]{Y'}\,,\ Y' > 0.008856 \\ 7.787 \cdot Y' + \frac{4}{29}\,,\ otherwise \end{cases} \\ Z'' &= \begin{cases} \sqrt[3]{Z'}\,,\ Z' > 0.008856 \\ 7.787 \cdot Z' + \frac{4}{29}\,,\ otherwise \end{cases} \end{aligned} \tag{B.8}$$

$$
\begin{aligned}
L &= 116 \cdot Y'' - 16 \\
A &= 500 \cdot (X'' - Y'') \\
B &= 200 \cdot (Y'' - Z'')
\end{aligned}
\tag{B.9}
$$

The conversion from the LAB color space to the RGB color space also uses the intermediate XYZ color space to facilitate the conversion. From the LAB color space, the XYZ color space is defined by eq.B.12 with equations B.10 and B.11 presenting intermediate steps.

$$
\begin{aligned}
X'' &= \frac{L+16}{116} + \frac{A}{500} \\
Y'' &= \frac{L+16}{116} \\
Z'' &= \frac{L+16}{116} - \frac{B}{200}
\end{aligned}
\tag{B.10}
$$

$$
\begin{aligned}
X' &= \begin{cases} \sqrt[3]{X''}\,, & X'' > 0.206897 \\ \frac{X'' - \frac{4}{29}}{7.787}\,, & otherwise \end{cases} \\
Y' &= \begin{cases} \sqrt[3]{Y''}\,, & Y'' > 0.206897 \\ \frac{Y'' - \frac{4}{29}}{7.787}\,, & otherwise \end{cases} \\
Z' &= \begin{cases} \sqrt[3]{Z''}\,, & Z'' > 0.206897 \\ \frac{Z'' - \frac{4}{29}}{7.787}\,, & otherwise \end{cases}
\end{aligned}
\tag{B.11}
$$

$$
\begin{aligned}
X &= 95.047 \cdot X' \\
Y &= 100.000 \cdot Y' \\
Z &= 108.883 \cdot Z'
\end{aligned}
\tag{B.12}
$$

From the XYZ color space, the RGB color space is defined by eq.B.15 with equations B.13 and B.14 presenting intermediate steps.

$$
\begin{aligned}
R'' &= 0.032406 \cdot X - 0.015372 \cdot Y - 0.004986 \cdot Z \\
G'' &= -0.009689 \cdot X + 0.018758 \cdot Y + 0.000415 \cdot Z \\
B'' &= 0.000557 \cdot X - 0.002040 \cdot Y + 0.010570 \cdot Z
\end{aligned}
\tag{B.13}
$$

$$
\begin{aligned}
R' &= \begin{cases} 1.055 \cdot R''^{\frac{1}{2.4}} - 0.055\,, & R'' > 0.0031308 \\ 12.92 \cdot R''\,, & otherwise \end{cases} \\
G' &= \begin{cases} 1.055 \cdot G''^{\frac{1}{2.4}} - 0.055\,, & G'' > 0.0031308 \\ 12.92 \cdot G''\,, & otherwise \end{cases} \\
B' &= \begin{cases} 1.055 \cdot B''^{\frac{1}{2.4}} - 0.055\,, & B'' > 0.0031308 \\ 12.92 \cdot B''\,, & otherwise \end{cases}
\end{aligned}
\tag{B.14}
$$

$$
\begin{aligned}
R &= 255 \cdot R' \\
G &= 255 \cdot G' \\
B &= 255 \cdot B'
\end{aligned}
\tag{B.15}
$$

B.4 HSV

Assuming hue in degrees, the conversion from the RGB color space to the HSV color space is defined by eq.B.16.

$$
\begin{aligned}
H &= \begin{cases} 360 - \arccos\left(\frac{R-0.5\cdot G-0.5\cdot B}{\sqrt{R^2+G^2+B^2-R\cdot G-R\cdot B-G\cdot B}}\right), B > G \\ \arccos\left(\frac{R-0.5\cdot G-0.5\cdot B}{\sqrt{R^2+G^2+B^2-R\cdot G-R\cdot B-G\cdot B}}\right), \textit{otherwise} \end{cases} \\
S &= \begin{cases} 1 - \frac{\min(R,G,B)}{\max(R,G,B)}, \max(R,G,B) > 0 \\ 0, \textit{otherwise} \end{cases} \\
V &= \frac{\max(R,G,B)}{255}
\end{aligned}
\tag{B.16}
$$

To convert the HSV color space to the RGB color space, three auxiliary values must first be defined (eq.B.17). Depending on the value of hue, the conversion to the RGB color space is then defined in eq. B.18.

$$
\begin{aligned}
M &= 255 \cdot V \\
m &= M \cdot (1 - S) \\
z &= (M - m) \cdot (1 - |\frac{H}{60} \pmod 2 - 1|)
\end{aligned}
\tag{B.17}
$$

$$
\begin{aligned}
0 \leq H < 60 \rightarrow & \begin{cases} R = M \\ G = z + m \\ B = m \end{cases} \\
60 \leq H < 120 \rightarrow & \begin{cases} R = z + m \\ G = M \\ B = m \end{cases} \\
120 \leq H < 180 \rightarrow & \begin{cases} R = m \\ G = M \\ B = z + m \end{cases} \\
180 \leq H < 240 \rightarrow & \begin{cases} R = m \\ G = z + m \\ B = M \end{cases} \\
240 \leq H < 300 \rightarrow & \begin{cases} R = z + m \\ G = m \\ B = M \end{cases} \\
300 \leq H < 360 \rightarrow & \begin{cases} R = M \\ G = m \\ B = z + m \end{cases}
\end{aligned}
\tag{B.18}
$$

Appendix C

Article - EUVIP 2022

Explorations on 3D point clouds coding using transformers and patches

Miguel Marques, Luís A. da Silva Cruz
University of Coimbra, Department of Electrical and Computer Engineering and Instituto de Telecomunicações
Coimbra, Portugal
miguel.marques@co.it.pt, lcruz@deec.uc.pt

***Abstract*—As 3D point clouds become more common as a representation of 3D visual content, the need to efficiently compress these data grows ever stronger. Research has shown that deep learning based approaches to point cloud coding see an increase in performance when compared with competing encoders like those developed by MPEG. This article examines and evaluates the use of the Transformer architectures and patch-based inputs combined with well developed deep learning static point cloud compression solutions described in the literature. To that end, we propose four new deep learning encoders. The obtained results show an improvement over an octree based encoder proposed by MPEG and the baseline PCC Geo v2 codec in terms of a geometry fidelity metric. The article also presents an ablation study conducted to analyze the impact of several encoder related parameters and structures that can guide future research in deep learning point cloud compression.**

***Index Terms*—Point cloud, compression, deep learning**

I. Introduction

Point clouds (PCs) are a data structure recognized as being of great importance to represent 3D visual content. They're used in several applications, such as the representation of LIDAR signals, the preservation of cultural artifacts and augmented and virtual reality. In a way, PCs are the 3D extension of 2D images. Put simply, they can be represented by a group of unordered 3D points, each one represented by their cartesian coordinates (x, y, z). Moreover, each one of these points can have multiple different attributes such as RGB color, transparency, normal vectors and/or other application relevant characteristics. However, the push to achieve realistic and immersive scenes leads to PCs having millions or tens of million of points. As such, their file size can become very large very quickly, leading to the need to efficiently compress these data structures. One class of PC coding methods is based on the use of deep convolutional processing structures designed using machine learning techniques. Recently, Transformer-based models have seen a surge in applications to computer vision tasks [1] due to their inherently more powerful architecture. It is then only natural to explore the application of this type of architecture to deep learning (DL) based point cloud compression [2] algorithms. However, as pointed by [3], a lot of the works that adapt the Transformer architecture to their desired application tend to feed a patch-based input to the network. While they achieve better results, not a lot of research has been made in deciphering exactly where this performance gain comes from. Is it due to the Transformer architecture, or to the use of patch-based inputs? In this work, we will present several experiments in DL-based point cloud compression (PCC) that try to answer this question while also proposing a model that combines both approaches.

This research was funded by the Portuguese research funding agency Fundação para a Ciência e a Tecnologia (FCT) under Projects UIDB/EEA/50008/2020 and LA/P/0109/2020.

II. Related Work

Current state-of-the-art DL-based PC codecs employ an end-to-end compression AE-like model [4] [5]. Both the quantization and entropy coding steps are considered during training using a variational auto encoder (AE) to learn the context model of the entropy coding. Furthermore, the model considers a rate distortion (RD) loss, $L = \lambda \cdot D + R$, using a lagrangian multiplier to choose a specific RD trade-off at training time. An example of a basic pipeline of an end-to-end PCC model can be seen in Fig. 1. While [4] [5] use a binary map to represent the occupancy state of the individual voxels of the PC to be encoded, there are other methods that operate directly on point coordinates [6]. This work will use the former representation as the input of the DL model. There exist other non-DL based PCs encoding methods with very good performance, with relevance to those developed by the the Moving Picture Experts Group (MPEG). Recently MPEG finished two standards for PC coding [7]: the Geometry-based Point Cloud Compression (G-PCC) and the Video-based Point Cloud Compression (V-PCC). The G-PCC encoder leverages octree structures [8] that partition the PC into blocks, encoding the occupancy as nodes in a tree. The V-PCC encoder projects the PC into different 2D views and then uses a video codec to encode the 2D projections. This encoder can handle dynamic PCs and achieves very good encoding performance, even when encoding a single PC.

III. Proposed Methods

A. Model Architecture

The proposed base model consists in a 3D AE architecture featuring latent entropy modeling with the objective of compressing PC geometry. The general pipeline is the same as previously mentioned and represented in Fig. 1. The processing flow starts with the division of input PC into octree blocks, so that each block will be encoded separately. The input of the DL model is then a $n \times n \times n$ 3D block of

978-1-6654-6623-3/22/$31.00 © 2022 IEEE

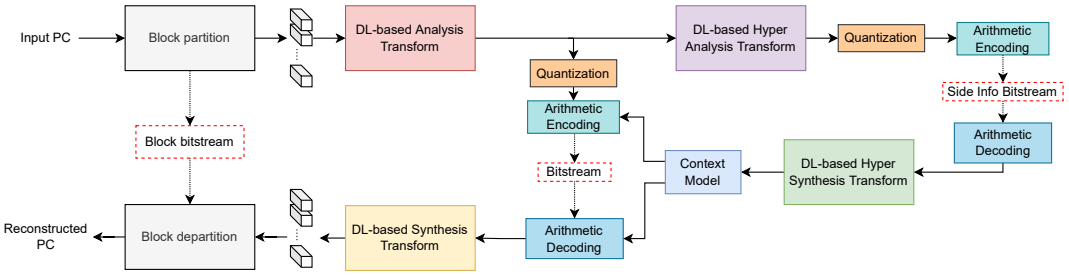


Fig. 1: End-to-end point cloud compression model based on [4].

voxel occupancy flags with a 1 when that voxel contains a point and a 0 otherwise. A latent representation of the input block is then computed by the DL-based multi-layer Analysis Transform which is then quantized and coded by an adaptive entropy coder. The entropy coder modelling is based on hyperprior concepts, where the latent representation is passed to a DL-based Hyper Analysis Transform that updates and encodes the entropy model used in the entropy encoding of the block's latent representation. This operation generates a side bitstream that is also stored or transmitted alongside the main bitstream. On the decoder side, the DL-based Hyper Synthesis Transform recovers the entropy model used to entropy decode the main bitstream and obtain an approximation to the encode-side latent representation, which then gets passed to the DL-based Synthesis Transform to recover an approximation to the original block. Finally, an octree-based PC rebuilding step is applied to all the reconstructed blocks. This last step relies on information describing the octree partitioning (first step of processing) which is also stored as a bitstream. In the end, the compressed final bitstream is made up of three different bitstreams:

- The main bitstream comprised of the information representing the encoded blocks.
- The side bitstream comprising the information of the context model for the entropy coding.
- The block bitstream describing the octree structure resulting from the initial block partitioning of the input point cloud.

The focus of this work will be on alternative processing structures for the DL-based Analysis Transform. Four different encoders will be proposed, based on different Analysis Transform structures:

- Full Transformer Encoder (FTE).
- Progressive Transformer Encoder (PTE).
- Full ConvMixer Encoder (FCME).
- Progressive ConvMixer Encoder (PCME).

The Transformer-based encoders are based on the architecture proposed in [9], but adapted to process 3D information. Similarly, the ConvMixer encoders are 3D adaptations of existing structures, in this case of the architecture described in [3]. Graphical representations of the basic processing structures and the analysis transform structures can be seen in Fig. 2. The objective of all the architectures is to process an incoming $M \times M \times M$ tensor into an $M/8 \times M/8 \times M/8$ H-channels tensor. In the case of the *full* variant of the encoders, the patch embedding layer, that is to say, the first convolutional layer, generates an N channel tensor from the input block's patches. In this case, a patch is considered to be an $8 \times 8 \times 8$ block of the input block, however, the weights that consider the value of each input voxel to that patch are learned by the model during training. The resulting tensor is then processed by the Transformer Encoder Layer (TEL) or ConvMixer Encoder Layer (CMEL) and the final convolutional layer brings the number of channels back to the desired H. In this variant, the encoders combine the Transformer/ConvMixer architecture with patch-based inputs. In the *progressive* variant of the encoders, the $M \times M \times M$ input tensor is progressively compressed by the convolutinal layers with stride 2, while also gaining more and more channels. Before a new convolutional layer, the tensor is processed in a TEL or CMEL layer. As for the training loss function, the previously mentioned RD function, $L = \lambda \cdot D + R$, was used. The chosen distortion metric, given the great imbalance of filled voxels and empty voxels in a block, was the focal loss [10], defined by eqn. (1), where u is the original voxel value and v is the reconstructed voxel value and γ and α are two hyper parameters.

$$FL(v,u) = \begin{cases} -\alpha(1-v)^{\gamma}\log v, & u = 1 \\ -(1-\alpha)v^{\gamma}\log(1-v), & u = 0 \end{cases} \quad (1)$$

The γ parameter controls the relevance of hard to classify voxels by downplaying the loss impact of the easy ones. The higher the parameter is, the more relevant the hard voxels are. The $\alpha \in [0,1]$ parameter controls the importance of occupied voxels. The higher the parameter is, the more important the occupied voxels are. In this work those two parameters were set empirically to $\alpha = 0.75$ and $\gamma = 2$. The final classification of a voxel as filled or not-filled is done using adaptive thresholding of the network outputs, as proposed in [4].

B. Training Configuration and Dataset

All the models were implemented and trained in Tensorflow [11], version 2.5.2 on a Ubuntu 20.4 server with an NVIDIA RTX3090 GPU.

For training, the ModelNet40 dataset [12] was used, a mesh dataset of 40 different object classes. To transform the dataset into usable 3D point clouds, first, each mesh surface was

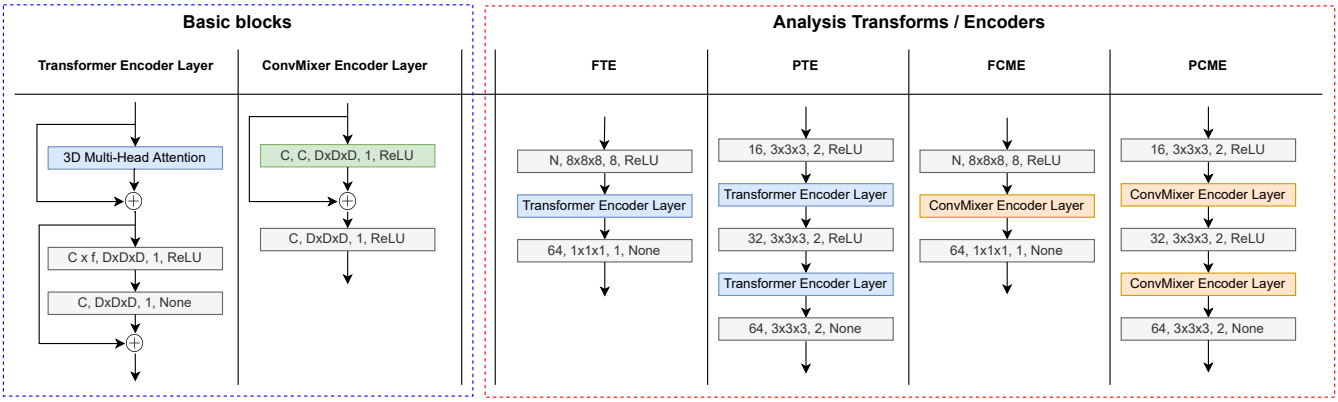


Fig. 2: Overview of the four proposed encoders. Grey blocks represent 3D convolutional layers and green blocks depthwise 3D convolutional layers. Text inside the convolutional blocks lists "Channels, Kernel Size, Stride, Activation Function" parameters. Text inside the depthwise convolutional blocks lists "Channels, Groups, Kernel Size, Stride, Activation Function" parameters.

sampled randomly, followed by a 9 bit voxelization of the sampled point cloud. Then, the 200 largest PCs in terms of point counts were partitioned using an octree decomposition of depth 3, leading to $64 \times 64 \times 64$ 3D blocks. Since most of those blocks have only $5 - 10\%$ filled voxels, only the 4000 blocks with the greatest number of points were used for training. In terms of model-related hyper parameters, a kernel size of $3 \times 3 \times 3$ for the TELs and CMELs was used and, in the case of the TELs, a factor f of $4/3$ was used. In this context, f is a factor that multiplies the number of channels in the first convolutional layer of the TELs (see the TEL in Fig. 2). These values were chosen empirically. In the case of the *full* variant encoders, when it came to studying the impact of N (number of channels in the first convolutional layer of its encoder) on the encoding performance, four values were used: 64, 128, 256 and 512. Finally, the lagrangian multiplier λ used in the loss function to control the RD tradeoff, five values were used: $3e^{-4}$, $1e^{-4}$, $5e^{-5}$, $2e^{-5}$ and $5e^{-6}$. In the testing phase, the five values selected lead to five different values of RD performance, one for each DL model.The Adam optimizer was used to minimize the training loss with a learning rate of 10^{-4}. After some preliminary tests the batch size was fixed at 32 blocks. A similar experiment was performed with 10 bit voxelized PCs for the training dataset without significantly different results.

IV. Results

A. Testing configuration

Two PCs were used to test the models and compute their average encoding performance, namely *longdress* [13] a relatively dense PC, and *Statue Klimt* [14] a more sparse PC. Figure 3 shows two views of these point clouds. These PCs were voxelized with 10 bit voxel precision, and then partitioned using an octree with depth 4 to obtain $64 \times 64 \times 64$ 3D blocks. To compare the RD performance of the various DL PCC models and the benchmarks, both the rate and the geometry distortion were measured and represented in a RD

(a) Longdress original. (b) Klimt original.

Fig. 3: Point clouds used to evaluate models.

graph. The number of bits needed to store all the bitstreams necessary to reconstruct the PC, was divided by the total number of points in the original PC to obtain the coding rate expressed in bits-per-point (bpp). As for the geometry distortion, the logarithmic versions of the common point-to-point and point-to-plane distances were used, i.e. D1 PSNR and D2 PSNR metrics, respectively, as defined in [15] and implemented in *mpeg-pcc-dmetric*, version 0.13.5 [16]. Finally, the Bjontegaard Deltas (BD) [17] were calculated using the PSNRs and bitrates of the proposed and some of the baseline encoders. For benchmarking purposes, the G-PCC (in octree mode) and V-PCC with the VVC video codec (in intra mode with default settings) MPEG codecs were used as baselines, compressing the test PCs at various rates. Furthermore, PCC GEO V2 [4], another DL-based PCC model was used as another baseline, since it was the work that this study was based on. The ADLPCC codec [5] was also included in the tests since it is a recent DL-based PC coder. It should be noted that the coding experiments listed in [5] are based on a 9-bit precision version of *Statue Klimt*, whereas in this paper a 10-

bit version was used.

B. Experimental results

1) FTE: As is the case with other DL-based PCC solutions, FTE significantly outperforms the G-PCC encoder. More so in the dense *longdress* PC than in the more sparse and noisy *Statue Klimt* PC. In fact, as one can observe in the graphs of Fig. 4a, the DL-based FTE and PCC GEO V2 have clearly inferior performance compared to the G-PCC encoder at high bitrates for the sparse PC. This can be caused by a multitude of reasons, but one of the strongest ones is the fact that the training dataset, ModelNet40, comprises almost exclusively dense PCs. Therefore, it is not surprising that the final model has a bias towards better compressing dense PCs. On the other hand, supported by the fact that the G-PCC and V-PCC encoders require higher bitrates for the sparse PC to be able to achieve a closer D1 PSNR value to the dense PC, compressing a sparse PC is inherently a much harder task. In terms of the impact of the capacity of the model N, comparing to the V-PCC encoder (Table I), the results are mixed between the dense and sparse PCs. For the dense PC, more channels have a marginal increase in performance. For the sparse PC, fewer channels see an increase in performance. An intuitive reason for this change is that, with fewer channels, the model has less capacity to overfit to dense PCs, leading to a more general DL-based encoder. Nevertheless, the V-PCC encoder outperforms the FTE in both testing PCs overall.

Comparing the FTE to the PCC GEO V2 encoder, the PCC GEO V2 encoder marginally outperforms the FTE in the case of the dense PC, with a slight increase in rate. However, with the sparse PC, the FTE comes out on top, even with a reduction in rate. In fact, the PCC GEO V2 encoder has a bias towards better compressing dense PCs, as will be the case in subsequent analysis. Concentrating now on the influence of N in the RD performance, the results in Table I are strange. In terms of the BD-PSNR gain, the 256 channel model dominates the results in all PCs. In terms of BD-rate, the 64 channel model outperforms the other models. Given the *black box* nature of DL, the explanation for these results can be hard to grasp. Considering the results in comparing with the V-PCC (and G-PCC) encoder, one likely explanation might just be that the number of channels in question were sweet spots considering all the other test conditions.

2) PTE: Analysing Table III, as was the case with the FTE, the PTE outperforms the G-PCC encoder in a similar fashion, being more biased towards dense PCs. Also, analogous to the FTE, the V-PCC encoder outperforms the PTE in a similar manner, being more biased towards dense PCs. On the other hand, the PCC GEO V2 encoder outperforms the PTE on the dense PC while falling slightly behind on the sparse PC. However, this comes at the cost of a slight increase in rate, so the two encoders come out similar in performance. It seems that the patch-based inputs may have had more of an impact on performance than the Transformer's attention-based architecture. Nonetheless, this hypothesis leads us into the ConvMixer-based encoders.

3) FCME: As can be seen in Fig. 4c, the FCME clearly outperforms the G-PCC encoder. Similar to previous encoders, it also has a bias towards dense PCs and underperforms the G-PCC encoder for high bitrates on sparse PCs.

In the case of the V-PCC encoder, as can be seen in Table II, once again, it outperforms the DL-based solutions for denser PCs, struggling to keep up on the sparse PC. In terms of the impact of N, the 512 channel model has better BD-PSNR gains on the dense PC, while the 128 channel model has better BD-rates. In the sparse PC, the 256 channel model is better overall.

Comparing the FCME to the PCC GEO V2 encoder, the FCME outperforms the PCC GEO V2 encoder on every test PC. In terms of BD-PSNR gain, either the 512 or 256 channel models see an improvement in performance. The same can be said for the 512 or 128 channel models for an improvement in BD-rate. In this case, increasing the capacity of the model usually increases its overall performance, albeit marginally and with an impact on model complexity.

So, is the increase in performance mainly contributed by the patch-based input representation after all? To answer this question, we have to conduct an analysis of the final encoder, the PCME.

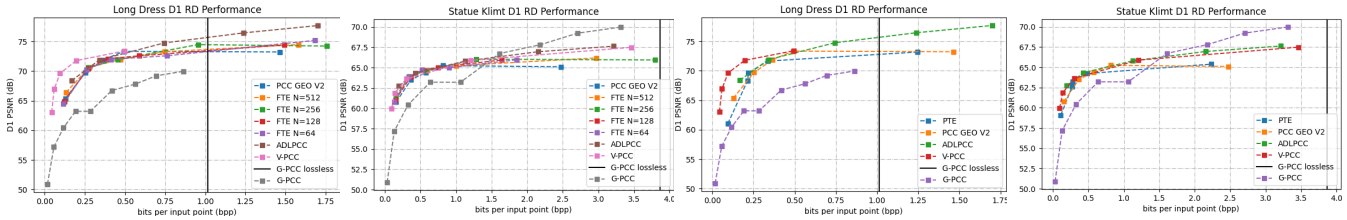
4) PCME: As was the case with the FCME, the PCME outperforms the G-PCC encoder in a similar fashion, being more biased towards dense PCs. Meanwhile, the V-PCC encoder outperforms the PCME at lower rates, being more biased towards dense PCs. On the other hand, the PCME outperforms the PCC GEO V2 encoder on all test PCs. This improvement in performance is translated by an increase in PSNR with a significant reduction in rate. On the other hand, looking at Fig. 4d, this gain in performance is not general. It mostly occurs at lower and higher bitrates. With medium bitrates, the PCC GEO V2 encoder is slightly better.

On another note, comparing Table II and Table IV, the PSNR gains of the PCME are, on the dense PC, not as high as its full variant counterpart. This fact makes it clear that patch-based inputs can have a positive impact on performance.

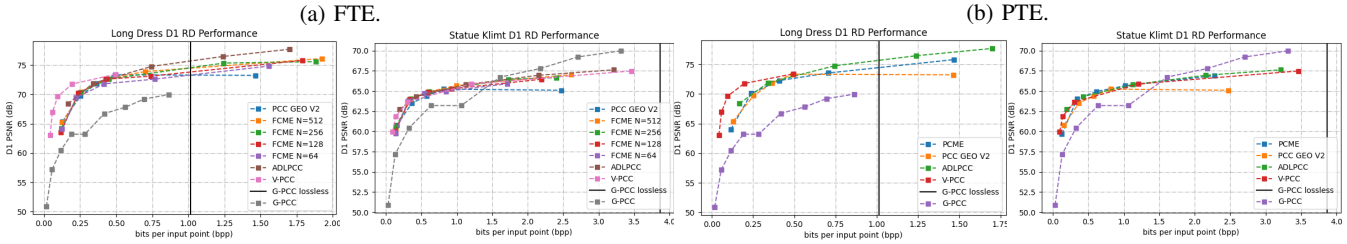
C. Qualitative study

While it is very important to use objective metrics when evaluating codec's RD performance, subjective quality is equally, if not more important. While no full subjective quality comparisons were done involving the results of this study, we present in Fig. 5 some images showing the difference between the original test PCs and the reconstructed PCs. An edge case for each PC is presented, that is to say, the reconstructed PC with the highest bitrate model for the PCME and PCC GEO V2 encoder were chosen for this part of the study. This was also the bitrate model where the most obvious visual difference between the original and reconstructed PC could be perceived.

In the case of the longdress test PC, it is clear from Fig. 5a that the PCC GEO V2 encoder struggles more to reconstruct the woman's hair than the PCME. On the other hand, the sparse Statue Klimt PC is not a great sight to behold in any of the reconstructed PCs. That being said, there are a lot more



(a) FTE. (b) PTE.



(c) FCME. (d) PCME.

Fig. 4: Testing PCs D1 PSNR RD performance graphs for the proposed encoders.

TABLE I: BD metrics for the FTE with N feature map channels compared to the V-PCC encoder (left table) and the PCC GEO V2 encoder (right table). The BD-rate is in percentage and the BD-PSNR is in dB. The best value is in bold.

			N							N			
Point Cloud	**Metric**	**BD**	***512***	***256***	***128***	***64***	**Point Cloud**	**Metric**	**BD**	***512***	***256***	***128***	***64***
longdress	D1	rate	**54.28**	57.22	55.56	54.52	longdress	D1	rate	-1.02	-0.84	-1.95	**-3.57**
		PSNR	**-2.14**	-2.57	-2.36	-2.50			PSNR	-0.22	**-0.14**	-0.24	-0.40
	D2	rate	**56.13**	58.57	57.18	56.45		D2	rate	-2.43	-1.35	-2.82	**-3.24**
		PSNR	**-2.93**	-3.49	-3.22	-3.39			PSNR	-0.24	**-0.17**	-0.28	-0.48
Statue Klimt	D1	rate	21.98	17.87	19.07	**15.64**	Statue Klimt	D1	rate	-27.88	-29.13	-25.97	**-34.70**
		PSNR	-0.46	-0.30	-0.29	**-0.27**			PSNR	0.27	**0.49**	0.35	0.38
	D2	rate	-42.49	-44.28	-44.96	**-50.54**		D2	rate	-10.92	-11.23	-9.24	**-12.83**
		PSNR	0.60	0.64	**0.98**	0.90			PSNR	0.11	**0.27**	0.07	0.04

TABLE II: BD metrics for the FCME with N feature map channels compared to the V-PCC encoder (left table) and the PCC GEO V2 encoder (right table). The BD-rate is in percentage and the BD-PSNR is in dB. The best value is in bold.

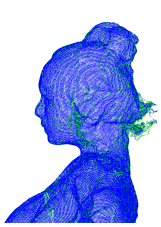
			N							N			
Point Cloud	**Metric**	**BD**	***512***	***256***	***128***	***64***	**Point Cloud**	**Metric**	**BD**	***512***	***256***	***128***	***64***
longdress	D1	rate	53.06	56.16	**51.83**	57.45	longdress	D1	rate	-12.60	-9.25	**-13.79**	2.49
		PSNR	**-2.17**	-2.48	-2.35	-2.62			PSNR	**0.26**	0.24	-0.06	-0.59
	D2	rate	54.60	57.10	**53.79**	58.97		D2	rate	-12.75	-9.62	**-13.30**	2.27
		PSNR	**-2.98**	-3.34	-3.20	-3.53			PSNR	**0.31**	0.29	-0.05	-0.71
Statue Klimt	D1	rate	13.67	**13.50**	16.42	22.32	Statue Klimt	D1	rate	-32.41	-31.01	**-35.32**	-26.27
		PSNR	-0.19	**-0.13**	-0.28	-0.39			PSNR	0.55	**0.59**	0.44	0.29
	D2	rate	-35.77	**-38.43**	-28.55	-25.90		D2	rate	**-10.63**	-9.88	-4.61	-1.23
		PSNR	0.83	**0.88**	0.65	0.67			PSNR	**0.25**	0.24	-0.05	-0.17

TABLE III: BD metrics for the PTE compared to the G-PCC, V-PCC and PCC GEO V2 encoders. The BD-rate is in percentage and the BD-PSNR is in dB.

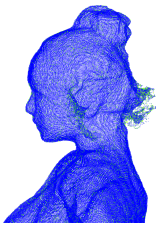
			Encoder		
Point Cloud	**Metric**	**BD**	***G-PCC***	***V-PCC***	***PCC GEO V2***
longdress	D1	rate	-141.91	58.39	-3.40
		PSNR	5.13	-3.40	-0.25
	D2	rate	-51.17	59.61	-2.32
		PSNR	3.43	-4.51	-0.22
Statue Klimt	D1	rate	-105.06	33.75	-7.95
		PSNR	1.70	-0.75	0.00
	D2	rate	-32.33	-21.76	-9.30
		PSNR	0.65	0.66	0.06

TABLE IV: BD metrics for the PCME compared to the G-PCC, V-PCC and PCC GEO V2 encoders. The BD-rate is in percentage and the BD-PSNR is in dB.

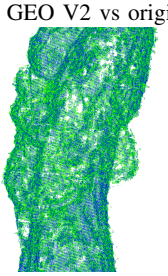
			Encoder		
Point Cloud	**Metric**	**BD**	***G-PCC***	***V-PCC***	***PCC GEO V2***
longdress	D1	rate	-227.08	53.58	-12.59
		PSNR	5.60	-2.42	0.16
	D2	rate	-103.96	55.01	-12.60
		PSNR	4.01	-3.29	0.19
Statue Klimt	D1	rate	-118.40	11.65	-36.07
		PSNR	2.16	-0.20	0.56
	D2	rate	-39.08	-37.58	-15.40
		PSNR	0.89	0.95	0.33



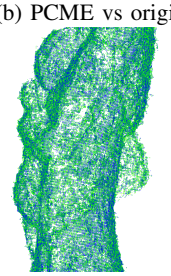
(a) PCC GEO V2 vs original.



(b) PCME vs original.



(c) PCC GEO V2 vs original.



(d) PCME vs original.

Fig. 5: Difference between the original test PCs and the reconstructed PCs. Points in blue mean that the reconstructed point is close (closer than, on average, four voxels) to the original, while points in green mean that the reconstructed point is far (farther than, on average, four voxels) from the original.

green points (inaccurate predictions) in the reconstructed PC from the PCC GEO V2 encoder than the PCME. Overall, visually, the highest bitrate model of the PCME has better performance than the highest bitrate model of the PCC GEO V2 encoder.

V. Conclusion

This paper introduced four new encoders for DL-based PCC. All of them encompassed the Transformer and ConvMixer architectures with the objective of studying their impact on PCC performance, as well as the impact of input-based representations when utilized in conjunction with these DL architectures. Overall, an improvement over G-PCC and [4] was obtained in terms of BD-PSNR and BD-rate. In conclusion, attention-based TELs did not seem to contribute much to the overall performance of the model. What made the biggest difference was the depthwise and pointwise convolutions of the CMELs accompanied by patch-based inputs. These operations enabled more efficient feature extraction, which ultimately resulted in more efficient PC compression. In the end, the better encoders were the ConvMixer-based ones, with the Transformer-based ones lacking behind. A summary of the best metrics for each test PC is displayed in the following Table V.

TABLE V: Summary of BD metrics for the proposed encoders compared to the PCC GEO V2 encoder. The BD-rate is in percentage and the BD-PSNR is in dB with the best value in bold. For the FTE, the 256 channel model was chosen as its overall best. For the FCME, the 512 channel model was chosen as its overall best.

			Encoder			
Point Cloud	**Metric**	**BD**	***FTE***	***PTE***	***FCME***	***PCME***
longdress	D1	rate	-0.84	-3.40	**-12.60**	-12.59
		PSNR	-0.14	-0.25	**0.26**	0.16
	D2	rate	-1.35	-2.32	**-12.75**	-12.60
		PSNR	-0.17	-0.22	**0.31**	0.19
Statue Klimt	D1	rate	-29.13	-7.95	-32.41	**-36.07**
		PSNR	0.49	0.00	0.55	**0.56**
	D2	rate	-11.23	-9.30	-10.63	**-15.40**
		PSNR	0.27	0.06	0.25	**0.33**

References

[1] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[2] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, "Pct: Point cloud transformer," *Computational Visual Media*, vol. 7, no. 2, pp. 187–199, 2021.

[3] A. Trockman and J. Z. Kolter, "Patches are all you need?" *arXiv preprint arXiv:2201.09792*, 2022.

[4] M. Quach, G. Valenzise, and F. Dufaux, "Improved deep point cloud geometry compression," 2020. [Online]. Available: https://github.com/mauriceqch/pcc_geo_cnn_v2.git

[5] A. F. Guarda, N. M. Rodrigues, and F. Pereira, "Adaptive deep learning-based point cloud geometry coding," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 2, pp. 415–430, 2020. [Online]. Available: https://github.com/aguarda/ADLPCC.git

[6] W. Yan, S. Liu, T. H. Li, Z. Li, G. Li *et al.*, "Deep autoencoder-based lossy geometry compression for point clouds," *arXiv preprint arXiv:1905.03691*, 2019.

[7] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li, J. Llach, K. Mammou, R. Mekuria, O. Nakagami, E. Siahaan, A. Tabatabai, A. M. Tourapis, and V. Zakharchenko, "Emerging mpeg standards for point cloud compression," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 1, pp. 133–148, 2019.

[8] R. Schnabel and R. Klein, "Octree-based Point-Cloud Compression," in *Symposium on Point-Based Graphics*, M. Botsch, B. Chen, M. Pauly, and M. Zwicker, Eds. The Eurographics Association, 2006.

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[10] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[11] M. A. et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[12] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[13] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, "8i voxelized full bodies-a voxelized point cloud dataset," *ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG) input document WG11M40059/WG1M74006*, vol. 7, p. 8, 2017.

[14] "Mpeg point cloud datasets." [Online]. Available: http://mpegfs.int-evry.fr/MPEG/PCC/DataSets/pointCloud/CfP

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

[16] "Mpeg point cloud compression distortion metric." [Online]. Available: http://mpegx.int-evry.fr/software/MPEG/PCC/mpeg-pcc-dmetric

[17] G. Bjontegaard, "Calculation of average psnr differences between rd-curves," *VCEG-M33*, 2001.