# A Deep Point Cloud Geometry Coding Toolbox

Maurice Quach, Giuseppe Valenzise, Frédéric Dufaux

# A DEEP POINT CLOUD GEOMETRY CODING TOOLBOX

*Maurice Quach* *Giuseppe Valenzise* *Frédéric Dufaux*

Université Paris-Saclay, CNRS, CentraleSupélec, Laboratoire des signaux et systèmes,
91190, Gif-sur-Yvette, France.

## ABSTRACT

This short paper describes a TensorFlow toolbox for point cloud geometry coding based on deep neural networks. This coding method employs a deep auto-encoder trained with a focal loss to learn good representations for voxel occupancy. The software provides several coding parameters to achieve different rate-distortion trade-offs, and comes with pre-trained models to reproduce the results of the published paper. It also offers a number of utility functions for evaluating and comparing the codec. To our knowledge, this is the first publicly available open-source toolbox for deep-learning-based point cloud coding.

## 1. DESCRIPTION AND OVERVIEW

Deep point cloud compression (PCC) is an emerging topic in multimedia coding, in particular for what concerns the coding of voxelized point clouds geometry [1, 2, 3]. As in other research fields, reproducing the results of existing approaches is of paramount importance to conceive new coding algorithms, especially when using deep neural networks, where the large number of variables involved makes it difficult to compare methods. This open source package offers a toolbox to compress point cloud geometry using deep neural networks. It implements all the contributions and experiments in our previous work [4]. The point cloud is partitioned into blocks and each block is compressed with a convolutional neural network, optimized with respect to a focal loss. In order to achieve different rate-distortion points, we train a network for each desired rate-distortion trade-off. At the decoder side, several parameters can be modified to adapt to the local point cloud density (balancing weight in the focal loss, optimal thresholding). Details on the choice of these parameters are reported in [4]. We complete the toolbox with a number of utility functions for evaluating and comparing the performance with other methods; we also provide pre-trained models and the datasets used to obtain them. We also provide instructions for users that want to retrain the networks on their own datasets.

We make the entire source code available at `https://github.com/mauriceqch/pcc_geo_cnn_v2` under the MIT license. We also provide an overview of the package in Figure 1. The source code contains a working implementation of all the compression models (conditions) described in the paper and includes the necessary code to reproduce all the experiments. Also, these experiments can easily be extended with other models or to other test point clouds by changing the configuration file `ev_experiment.yml`.

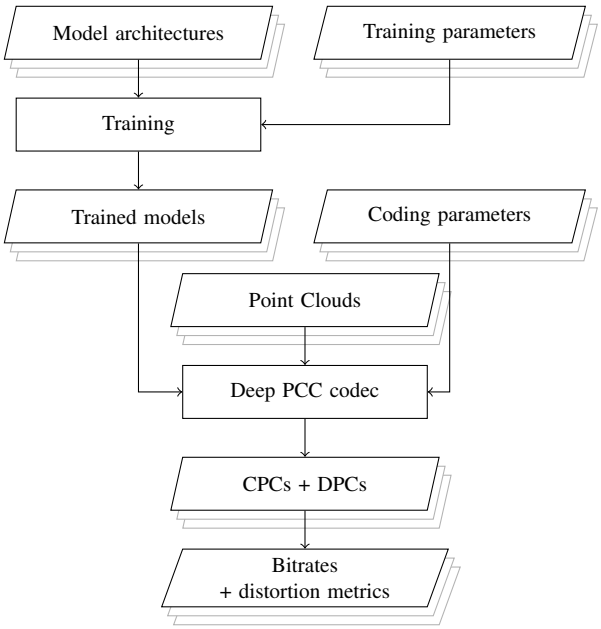


**Fig. 1**. System overview for the open source package. CPC stands for compressed PC and DPC for decompressed PC.

This open source package is intended for researchers and engineers working on PCC. It has two main purposes: i) facilitating the reproduction and comparison with the family of codecs in [4]; ii) providing the tools and a starting point to foster new research on deep PCC. In particular, the adopted licensing scheme allows the use of this software as a base for implementation in an industrial context.

## 2. EXAMPLE OF UTILIZATION

The package provides tools for each step of PCC. In particular, tools are provided to train the compression models, en-

Funded by the ANR ReVeRy national fund (REVERY ANR-17-CE23-0020). ICME2021 "Point cloud capture and compression" track.


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

**Fig. 2**. Qualitative evaluation on "soldier_vox10_0690". For our method and G-PCC Trisoup (baseline), we show the decompressed point cloud and its D1 squared errors. The errors are displayed according to the color scale on the right and are truncated to the 99th percentile (3.0). In parentheses, we specify the D1 PSNR along with the number of bits per input point (bpp).

code and decode PCs and evaluate them. Note that the package works with voxelized point clouds.

### 2.1. Training

In order to train a compression model, we need a model architecture and training parameters. Model architectures are defined in `model_configs.py` and are identified by a unique name. The training parameters include the training dataset, the rate-distortion tradeoff parameter (`--lmbda`), the focal loss parameters (`--alpha` and `--gamma`), the batch size and the model architecture (`--model_config`). Usually, multiple compression models are trained for a single architecture with different rate-distortion trade-offs. Below is a sample training command:

```
python tr_train.py \
    'ModelNet40_200_pc512_oct3_4k/**/*.ply' \
    model \
    --resolution 64 --lmbda 1.00e-04 --alpha 0.75 --gamma 2.0
         --batch_size 32 --model_config c3p
```

### 2.2. Coding a point cloud

Then, we can compress and decompress a point cloud using the previously trained model. Coding parameters include the model architecture, the octree level used for block partitioning and the optimal thresholding target metrics (`--opt_metrics`). In the following example, the software encodes and decodes a PC, employing the point-to-point metric (D1) to optimize the thresholding at the decoder:

```
python compress_octree.py \
    --input_files soldier_vox10_0690.ply \
    --output_files soldier_vox10_0690_d1.ply.bin \
    --checkpoint_dir model \
    --opt_metrics d1_mse --resolution 1024 --model_config c3p
         --octree_level 4 \
    --dec_files soldier_vox10_0690_d1.ply.bin.ply
```

### 2.3. Visualization and evaluation

For visualization, we can transfer the colors from the original to the decompressed point cloud with the following command:

```
python map_color.py \
    soldier_vox10_0690.ply \
    soldier_vox10_0690_d1.ply.bin.ply \
    soldier_vox10_0690_d1.ply.bin.ply.color.ply
```

Then, we can use the `mpeg-pcc-dmetric` to compute distortion metrics:

```
~/code/MPEG/mpeg-pcc-dmetric/test/pc_error_d \
    -a soldier_vox10_0690.ply \
    -b soldier_vox10_0690_d1.ply.bin.ply \
    --resolution 1023 --dropdups 2 --neighborsProc 1
```

A sample compressed point cloud is shown in Figure 2; it is compared with the original and the same point cloud compressed with G-PCC. More details are available on Github.

## 3. REFERENCES

[1] M. Quach, G. Valenzise, and F. Dufaux, "Learning Convolutional Transforms for Lossy Point Cloud Geometry Compression," in *2019 IEEE Intl. Conf. on Image Process. (ICIP)*, Sept. 2019, pp. 4320–4324, ISSN: 1522-4880.

[2] A. F. R. Guarda, N. M. M. Rodrigues, and F. Pereira, "Point Cloud Coding: Adopting a Deep Learning-based Approach," in *2019 Picture Coding Symposium (PCS)*, Nov. 2019, pp. 1–5, ISSN: 2330-7935.

[3] J. Wang et al., "Learned Point Cloud Geometry Compression," *arXiv:1909.12037 [cs, eess]*, Sept. 2019.

[4] M. Quach, G. Valenzise, and F. Dufaux, "Improved Deep Point Cloud Geometry Compression," in *2020 IEEE 22nd Intl. Workshop on Multimedia Signal Process. (MMSP)*, Sept. 2020, pp. 1–6, ISSN: 2473-3628.