TransPCC: Towards Deep Point Cloud Compression via Transformers

Zujie Liang
Sun Yat-sen University, Guangzhou, China
liangzj9@mail2.sysu.edu.cn

Fan Liang*
Sun Yat-sen University, Guangzhou, China
Peng Cheng Laboratory, Shenzhen, China
isslf@mail.sysu.edu.cn

ABSTRACT

High-efficient point cloud compression (PCC) techniques are necessary for various 3D practical applications, such as autonomous driving, holographic transmission, virtual reality, etc. The sparsity and disorder nature make it challenging to design frameworks for point cloud compression. In this paper, we present a new model, called **TransPCC** that adopts a fully Transformer auto-encoder architecture for deep Point Cloud Compression. By taking the input point cloud as a set in continuous space with learnable position embeddings, we employ the self-attention layers and necessary point-wise operations for point cloud compression. The self-attention based architecture enables our model to better learn point-wise dependency information for point cloud compression. Experimental results show that our method outperforms state-of-the-art methods on large-scale point cloud dataset.

CCS CONCEPTS

• **Computing methodologies** → **Point-based models**; **3D imaging**; **Reconstruction**.

KEYWORDS

Point Cloud Compression, Auto-encoder, Transformers

ACM Reference Format:
Zujie Liang and Fan Liang. 2022. TransPCC: Towards Deep Point Cloud Compression via Transformers. In *Proceedings of the 2022 International Conference on Multimedia Retrieval (ICMR '22), June 27–30, 2022, Newark, NJ, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3512527.3531423

1 INTRODUCTION

Recently, 3D point cloud data is playing a more and more important role in many pratical applications. In particular, point clouds are essential for applications like virtual reality, holographic transmission, sensing for autonomous vehicle, etc. Point clouds are sets of 3D points identified by their coordinates, which constitute the geometry of the point cloud. Each point can also be associated with some attributes like colors, normals and reflectance. Large-scale and dense point clouds entails a huge memory and transmission cost. Hence, how to represent the point cloud in a compact and structural way is a fundamental problem in practice.

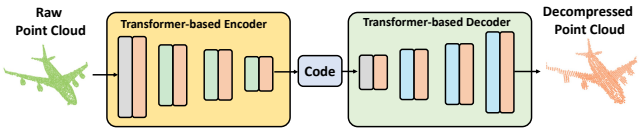

Figure 1: TransPCC is designed for point cloud compression task. We propose to adopt a transformer-based auto-encoder architecture to learn the complex dependencies among the points. The encoder takes the raw point clouds as inputs (green points) and extracts a compact representation from them, while the decoder directly operates on the compact embedding and decompresses the point cloud (orange points).

Conventional point cloud compression (PCC) methods are developed under the efforts of the experts in MPEG [17], such as Video-based PCC (V-PCC) and Geometry-based PCC (G-PCC) approaches. The V-PCC maps 3D points to 2D information format and then applies the 2D video codec to code projected planes, while G-PCC relied on 3D structure such as octree to encode the 3D content directly. These standard methods show good compression performance yet highly rely on handcrafted coding strategies and parameters.

Nowadays, deep learning has demonstrated its powerful ability in feature representation learning, which has also been applied in data compression tasks. With a deep auto-encoder structure, the encoder learns a small and compact feature representation (that allows for compression) and the decoder tries to reconstruct the original point cloud data in a data-driven manner. The voxel-based methods [7, 14, 20] firstly represent point cloud in a voxel grid, and then employ 3D convolution neural networks (CNN) in discrete geometric domain as feature extractor. However, these methods underutilizes the sparsity of the point clouds and induce heavy computational and memory costs that grow cubically as the spatial resolution increases. Another point-based methods [6, 10, 21] directly operate on the raw input points without voxelization. These methods typically present lightweight complexity cost, yet still suffer from poor reconstruction quality especially on large-scale point cloud scene.

In this paper, we propose a novel paradigm, called **TransPCC** that adopts a fully Transformer auto-encoder architecture for deep Point Cloud Compression. Recently, the great success of Transformers in Natural Language Processing [2, 3, 15, 19] has inspired the development of Transformer networks for Computer Vision

*Corresponding author. This work was partially supported by the National Natural Science Foundation of China (No. U20A20185).

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
ICMR '22, June 27–30, 2022, Newark, NJ, USA.
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9238-9/22/06…$15.00
https://doi.org/10.1145/3512527.3531423

tasks [4, 8, 16]. Further, the success of Transformers in 2D CV studies has inspired 3D point cloud research [12, 22]. They introduce a Transformer-based framework in the encoding process to perform point cloud representation learning. In this work, instead of only utilizing its representation learning ability, we take a step further to extend the application of transformer-based architecture into the whole pipeline (both the encoding and decoding process) and reveal its remarkable effectiveness in point cloud compression task. Through taking the input point cloud as a set embedded in continuous space, we employ the self-attention layer as the primary feature aggregation operator throughout the whole encoder and decoder network. In the encoder, we perform Farthest Point Sampling (FPS) to reduce the cardinality of the point set and the self-attention operator to model all pairwise interactions between points. We handle the large-scale point clouds by restricting self-attention blocks to considering nearest neighbors of individual points, which enables scalability to large scenes with millions of points. While in the decoder, we carry out interpolation procedure to progressively upsample the point set and the self-attention operator for feature propagation and aggregation. The aggregated features obtained from the Transformer can not only effectively improve the compression performance, but also brings powerful reconstruction ability. Note that all operations in our TransPCC framework are performed among the point sets, making our solution an efficient approach in practice without voxelization. Additionally, the whole auto-encoder architecture is easily trained in an end-to-end manner.

We conduct extensive experiments following the test conditions in [21] on the large-scale point cloud *SemanticKITTI* [1] dataset, which shows the superior compression efficiency of our proposed method, outperforming previous state-of-the-art methods by a large margin.

To sum up, our contributions are summarized as follows:

- To the best of our knowledge, we are the first to propose a fully Transformer based architecture for 3D point cloud compression. We design the end-to-end TransPCC framework, which adopts an auto-encoder structure fully based on self-attention operators and point-wise operations.
- Extensive experiments reveal that the proposed TransPCC achieves state-of-the-art compression performance on large-scale point cloud dataset.

2 METHODOLOGY

In this section, we introduce our approach named TransPCC. The overview of our proposed approach is illustrated in Fig. 2. In summary, the TransPCC adopts an end-to-end Auto-encoder transformer structure to perform point cloud compression and decompression. The encoder extracts a compact and memory-efficient representation from the original input point cloud by alternately applying down-sample and self-attention operators. Specifically, the encoder includes four stages that execute on progressively down-sampled point sets. The down-sampling rates for the stages are [1, 2, 2, 2], resulting in the cardinality of the point cloud set produced in each stage is [N, N/2, N/4, N/8]. The number of blocks and the down-sampling rates can be varied to achieve different compression ratio. For the decoder, it consists of four stages to progressively upsample the points and apply the self-attention operators for feature aggregation to achieve point reconstruction. Note that the decoder directly operates on a set of points, which avoids discretization. Through comparing the reconstructed points with the input point cloud in an appropriate metric, the self-supervised network will learn parameters in an end-to-end manner.

2.1 Encoder Blocks

We denote the input 3D point cloud as $\mathcal{P} = \left\{\left(\boldsymbol{x}_i, \boldsymbol{f}_i\right)\right\}, |\mathcal{P}| = N$, which consists of coordinates $\mathcal{X} = \left\{\boldsymbol{x}_i \in \mathbb{R}^3\right\}$ and point features $\mathcal{F} = \left\{\boldsymbol{f}_i \in \mathbb{R}^D\right\}$. Before we perform self-attention operation to the input points, we need to define position encoding, which enables the operator to fit in local structure in the data [19]. In 3D point cloud domain, the 3D point coordinates themselves are a natural candidate for position encoding. For this reason, we choose to perform trainable, parameterized position encoding δ, which is defined as follows:

$$\delta = \theta\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right) \tag{1}$$

where $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are the 3D point coordinates for points i and j. The encoding function θ is an MLP with two linear layers and one ReLU function. This position encoding θ is trained end-to-end with the other modules of our network.

Then, the self-attention layer used in our network is implemented as a set operator, *i.e.*,

$$\boldsymbol{y}_i = \sum_{\boldsymbol{f}_j \in \mathcal{F}(i)} \rho\left(\gamma\left(\varphi\left(\boldsymbol{f}_i\right) - \psi\left(\boldsymbol{f}_j\right) + \delta\right)\right) \odot \left(\alpha\left(\boldsymbol{f}_j\right) + \delta\right) \tag{2}$$

where $\boldsymbol{y}_i$ is the output feature. φ, ψ, and α are linear projections for point-wise feature transformations. δ is a position encoding function and ρ is softmax function. γ is an MLP that produces attention vectors for feature aggregation. Note that the attention weights here are vectors that can modulate individual feature channels. We construct a transformer block by the linear projections for dimension reduction, the self-attention layer for information interaction, and a residual connection for preventing the vanishing gradients. To reduce the cardinality of the point sets progressively, we build a down-sampling module after each transformer block, which includes Farthest Point Sampling (FPS) [13] operation and a kNN model for feature vector pooling. Each input feature goes through a linear transformation, followed by batch normalization, ReLU, and max pooling. We constuct the encoder network by stacking the transformer block and the down-sampling block progressively. The last layer consists of an MLP to compress the features of size $\mathbb{R}^{N \times D_{\text{out}}}$ to the desired dimension $\mathbb{R}^{N \times D_{\text{emb}}}$.

2.2 Decoder Blocks

The goal of the decoder network is to reconstruct the whole point clouds from the compact embedding. Here we propose a transformer-based decoder for feature propagation and aggregation, which brings powerful reconstruction ability. To map features from the down-sampled input point set onto its higher resolution point set, we process it by a linear layer, a batch normalization layer, a ReLU non-linear function and trilinear interpolation. These interpolated features from the preceding decoder stage are then passed into the same transformer block as in the encoder for feature refinement and information interaction. We feed the feature of the last layer to an MLP to refine the coordinates.

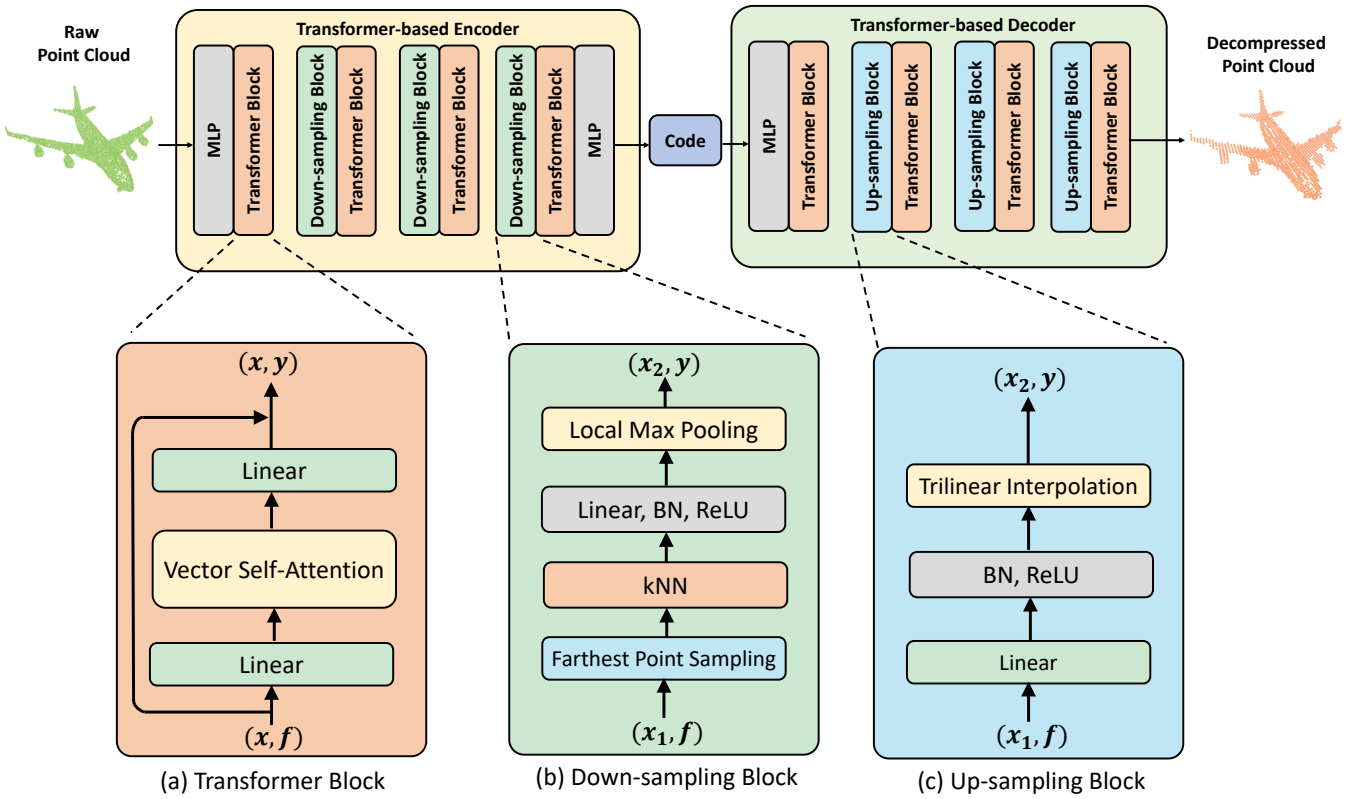

Figure 2: The architecture of our TransPCC network and detailed structure design for each block. The whole process is consisting of encoder part for data compression and decoder part for data decompression. The encoder takes a point cloud as input, and extracts a compact and memory-efficient representation from the original input point cloud by alternately applying down-sampling blocks and transformer operators. The compressed codeword is transferred from sender to receiver and will later be used by the decoder to decompress the point cloud. For the decoder, it progressively up-sample the points and apply the transformer blocks for feature aggregation to achieve point reconstruction. The self-supervised network will learn parameters in an end-to-end manner enforced by the loss function between the input and output. Encoder and decoder will be saved separately by sender and receiver to finish the whole process.

2.3 Optimization

The loss function for point cloud compression should provide a quantitative measurement for the quality of the reconstructed points. Directly measuring the distance between two points, such as euclidean distance, are unsuitable since the point clouds are unordered. Hence, we choose Chamfer Distance ($\mathcal{L}_{CD}$) [5] as our loss function, which measures the mean distance of one point to its nearest neighbor between two point sets:

$$\begin{aligned} \mathcal{L}_{\mathrm{CD}}\left(\mathcal{P}_{\mathrm{in}}, \mathcal{P}_{\mathrm{out}}\right) &= \overline{d^2}\left(\mathcal{P}_{\mathrm{in}}, \mathcal{P}_{\mathrm{out}}\right) + \overline{d^2}\left(\mathcal{P}_{\mathrm{out}}, \mathcal{P}_{\mathrm{in}}\right), \\ \bar{d}^2\left(\mathcal{P}_i, \mathcal{P}_j\right) &= \frac{1}{|\mathcal{P}_i|} \sum_{\boldsymbol{x}_i \in \mathcal{P}_i} \min_{\boldsymbol{x}_j \in \mathcal{P}_j} \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_2^2 \end{aligned} \tag{3}$$

where $\mathcal{P}_{\mathrm{in}}$ denotes the input point cloud and $\mathcal{P}_{\mathrm{out}}$ the decompressed point cloud. This metric is invariant to the permutation of points, and prevents the network from flooding the whole space with points.

Taking consideration of valid intermediate results in different scale of each decoder block and achieve multi-scale decompression, we add the Chamfer Distances between the input point cloud and the intermediate output points of all the decoder blocks to make up the Multi-scale loss function. which is defined as

$$\mathcal{L}_{\mathrm{Multi}} = \sum_j \mathcal{L}_{\mathrm{CD}}\left(\mathcal{P}_{\mathrm{in}}, \hat{\mathcal{P}}_j\right) \tag{4}$$

Finally, we construct our total loss $\mathcal{L}_{total}$ by summing the $\mathcal{L}_{\mathrm{CD}}$ and $\mathcal{L}_{\mathrm{Multi}}$ together in the following equation:

$$\mathcal{L}_{total} = \mathcal{L}_{\mathrm{CD}} + \lambda \mathcal{L}_{\mathrm{Multi}} \tag{5}$$

where the λ is a loss weight to trade-off the impact of the Multi-scale loss. We jointly optimize the parameters of the whole network

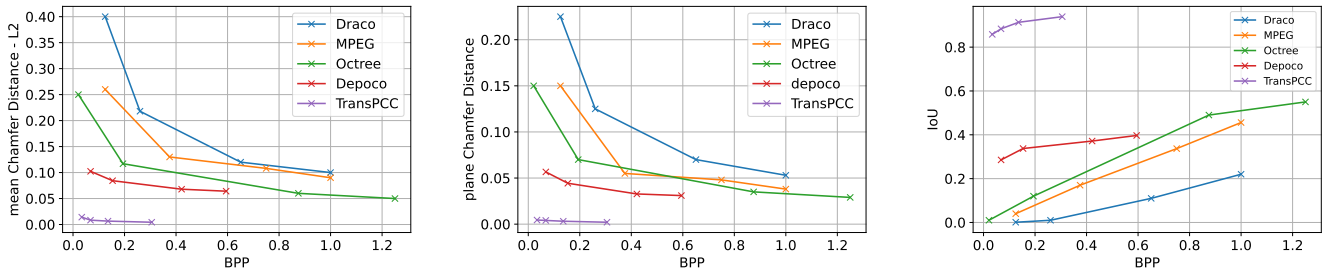

Figure 3: Compression results on the test split of the SemanticKITTI dataset. We compare our proposed TransPCC against the state-of-the-art methods, including Draco, the MPEG Anchor [11], a binary Octree implementation, and the deep learning based point cloud compression model Depoco [21]. Our approach outperforms the previous methods by a large margin under the same BPP in all metrics.

including both the encoder and decoder in an end-to-end training scheme.

3 EXPERIMENTS

3.1 Experiment Setup

3.1.1 Datasets. To validate the effectiveness of our proposed method on the point cloud compression task, we choose to conduct experiments on the *SemanticKITTI* [1] dataset. We follow all the same pre-process procedure and the same data split suggested in [21], where using sequence 00 to 10 (except 08) for training, a small subset of the training data for validation, and sequence 08 for testing.

3.1.2 Evaluation metrics. To quantitative evaluate the quality of a compression algorithm, we focus on both compression ratio and reconstruction error. A good compression algorithm should have efficient compression ratio and low reconstruction error. We adopt the Bits Per Points (BPP) required for storing the encoding of the input point cloud For the evaluation of compression ratio, As for the measurement of the reconstruction error, We use 3 metrics following [21]. The first one is mean Chamfer Distance D_d,

$$\begin{aligned} D_d\left(\mathcal{P}_{\text{in}}, \mathcal{P}_{\text{out}}\right) &= \bar{D}_d\left(\mathcal{P}_{\text{in}}, \mathcal{P}_{\text{out}}\right) + \bar{D}_d\left(\mathcal{P}_{\text{out}}, \mathcal{P}_{\text{in}}\right), \\ \bar{D}_d\left(\mathcal{P}_i, \mathcal{P}_j\right) &= \frac{1}{|\mathcal{P}_i|} \sum_{\boldsymbol{x}_i \in \mathcal{P}_i} \min_{\boldsymbol{x}_j \in \mathcal{P}_j} d\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right). \end{aligned} \tag{6}$$

which indicates the distance $\bar{D}_d$ from the ground truth point cloud $\mathcal{P}_{in}$ to the reconstruction $\mathcal{P}_{out}$ and vise versa, where we adopt the L2-norm to calculate the distance between two points, *i.e.*, $d = \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|_2$. Another metric is the symmetric plane Chamfer Distance $D_\perp = D_d\left(\mathcal{P}_{\text{in}}, \mathcal{P}_{\text{out}}\right)$ with $d = \left|\boldsymbol{n}^T\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right)\right|$, where $\boldsymbol{n} \in \mathbb{R}^3$ denotes the ground truth normal of that point. The third metric is the Intersection-Over-Union (IoU) between occupancy grids G for both point clouds, *i.e.*,

$$IoU = \frac{|G_{\text{in}} \cap G_{\text{out}}|}{|G_{\text{in}} \cup G_{\text{out}}|}. \tag{7}$$

where the occupancy grids G_{in} and G_{out} have a resolution of $20 \times 20 \times 10$ cm^3 as used by [9].

3.1.3 Implementation details. The proposed model is implemented in PyTorch [1] and trained on one NVIDIA RTX3090 GPU. For the fair comparison, we keep the data pre-processing steps and hyperparameter settings the same as the Depoco model [21] in the released implementation[2]. We use the Adam optimizer with weight decay and initial learning rate set to $1e-4$ and $1e-3$, respectively. The batch size is set to 16. We optimize the total loss $\mathcal{L}_{total}$ with the weight $\lambda = 0.2$ for 100 epochs.

3.2 Results and Analysis

We present the compression performance on the SemanticKITTI dataset in Fig. 3. We choose Draco[3], the MPEG Anchor [11], a binary Octree implementation, and the deep learning based point cloud compression model Depoco [21] as our baselines for comparison. As we can see, our proposed method outperforms the all the baselines in terms of Chamfer Distance based metrics as well as IoU. It is noteworthy that our TransPCC achieves 6.5 times lower reconstruction errors for bit rates under 0.2 BPP compared to the strongest baseline Depoco, which is also a deep learning approach based on KPConv [18]. The large performance gap between TransPCC and Depoco demonstrates our method learns a more expressive point feature, which is very essential for point cloud compression task.

4 CONCLUSIONS

In this paper, we have presented TransPCC, a Transformer framework for deep point cloud compression via an auto-encoder architecture. The entire network architecture is fully built on top of the self-attention operator and point-wise operations. Experimental results verify that the TransPCC achieves a new state-of-the-art performance on the large-scale dense point cloud dataset, revealing the remarkable effectiveness of Transformers in 3D point cloud compression. In future, we will do further study on the Transformer-based deep point cloud compression, such as specific network designs, new operator development, and so on.

[1] https://github.com/jokieleung/TransPCC
[2] https://github.com/PRBonn/deep-point-map-compression
[3] https://github.com/google/draco

REFERENCES

[1] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. 2019. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9297–9307.

[2] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. 2019. Transformer-XL: Attentive Language Models beyond a Fixed-Length Context. In *ACL (1)*.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

[5] Haoqiang Fan, Hao Su, and Leonidas J Guibas. 2017. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 605–613.

[6] Linyao Gao, Tingyu Fan, Jianqiang Wan, Yiling Xu, Jun Sun, and Zhan Ma. 2021. Point Cloud Geometry Compression Via Neural Graph Sampling. In *ICIP*. 3373–3377. https://doi.org/10.1109/ICIP42928.2021.9506631

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

[8] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. 2019. Local relation networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3464–3473.

[9] Lila Huang, Shenlong Wang, Kelvin Wong, Jerry Liu, and Raquel Urtasun. 2020. Octsqueeze: Octree-structured entropy model for lidar compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1313–1323.

[10] Tianxin Huang and Yong Liu. 2019. 3D Point Cloud Geometry Compression on Deep Learning. In *ACM Multimedia* (Nice, France) *(MM '19)*. Association for Computing Machinery, New York, NY, USA, 890–898. https://doi.org/10.1145/3343031.3351061

[11] Rufael Mekuria, Kees Blom, and Pablo Cesar. 2016. Design, implementation, and evaluation of a point cloud codec for tele-immersive video. *IEEE Transactions on Circuits and Systems for Video Technology* 27, 4 (2016), 828–842.

[12] Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 2021. 3d object detection with pointformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7463–7472.

[13] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. 2017. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *Advances in Neural Information Processing Systems* 30 (2017).

[14] Maurice Quach, Giuseppe Valenzise, and Frederic Dufaux. 2019. Learning convolutional transforms for lossy point cloud geometry compression. In *ICIP*. IEEE, 4320–4324.

[15] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. [n.d.]. Language models are unsupervised multitask learners. ([n. d.]).

[16] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. 2019. Stand-Alone Self-Attention in Vision Models. *Advances in Neural Information Processing Systems* 32 (2019).

[17] Sebastian Schwarz, Marius Preda, Vittorio Baroncini, Madhukar Budagavi, Pablo Cesar, Philip A Chou, Robert A Cohen, Maja Krivokuća, Sébastien Lasserre, Zhu Li, et al. 2018. Emerging MPEG standards for point cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 1 (2018), 133–148.

[18] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, Francois Goulette, and Leonidas J. Guibas. 2019. KPConv: Flexible and Deformable Convolution for Point Clouds. In *ICCV*.

[19] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[20] Jianqiang Wang, Hao Zhu, Haojie Liu, and Zhan Ma. 2021. Lossy Point Cloud Geometry Compression via End-to-End Learning. *TCSVT* (2021), 1–1. https://doi.org/10.1109/TCSVT.2021.3051377

[21] Louis Wiesmann, Andres Milioto, Xieyuanli Chen, Cyrill Stachniss, and Jens Behley. 2021. Deep Compression for Dense Point Cloud Maps. *IEEE Robotics and Automation Letters* 6, 2 (2021), 2060–2067. https://doi.org/10.1109/LRA.2021.3059633

[22] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. 2021. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 16259–16268.