PCIT: A Point Cloud Invariance Transformer

Changjie Qiu[1],Zhicheng Yang[2],Haiyun Tao[3],Shangpeng Han[4],Chengbin Zheng[5]
School of Informatics, Xiamen University, Xiamen, China
[1]23020211153904,[2]23020211153909,[3]23020211153967,[4]31520211154046,[5]23020211153991@stu.xmu.edu.cn

Abstract

Point clouds data is a type of set, which embedded in a continuous space. This makes point clouds structurally different from images and precludes immediate application in deep network designing.We proposes a new framework named PCIT(Point Cloud Invariance Transformer) for the task of point cloud learning. PCIT is based on transformer, which achieves huge success in natural language processing and displays great potential in image processing. We investigated several improvement directions for the transformer, and made comparative experiments and effect summary. On this basis, we enhanced the ordinary transformer for the point cloud. We used multi-kernel dynamic self attention and kernel weight layer to ensure the order invariance and locality of the point cloud. Extensive experiments demonstrate that PCIT achieves the state-of-the-art performance on shape classification tasks.

Introduction

Dealing with sparse and irregular point clouds data has emerged as an important issue for many areas such as robotics, autonomous driving, augmented reality, etc. As point clouds are disordered and unstructured, it's more challenging to design neural networks to process them.

Qi et al [9] pioneered PointNet for feature learning on point clouds by using multi-layer perceptrons (MLPs), max-pooling and rigid transformations to ensure invariance under permutations and rotation. Some recent works [12, 7, 1, 15] have considered to define convolution operators that can aggregate local features for point clouds. These methods either reorder the input point sequence or voxelize the point cloud to obtain a canonical domain for convolutions.

Recently, Transformer [13], the dominant framework in natural language processing, has been applied to image vision tasks, Inspired by the great performance given by transformer, Various other methods employ attention and transformer.Yan et al. [17] proposed PointASNL to deal with noise in point cloud processing, using a self-attention mechanism to update features for local groups of points. Hertz et al. [4] proposed PointGMM for shape interpolation with both multi-layer perceptron (MLP) splits and attentional splits. Guo et al. [2] use PCT ,which is based on transformer rather than using self-attention as an auxiliary module to extract semantics from a point cloud.

However, in most of above frameworks, They do not take full advantage of the order invariance and rotation invariance, which are two important features of point clouds.So in this work, we propose the Point Cloud Invariance Transformer(PCIT) to utilize this two characters of points cloud.

In order to consider order invariance and rotation invariance into our framework, we make several adjustments for this. These include:

- We use the method of multi-kernel and kernel weight layer to make the network select the field of view according to the point cloud.so that it has order invariance.
- We use dynamic convolution instead of ordinary convolution to simulate the direction, so that it has local rotation invariance.

With the above adjustments, the PCIT is more suitable for point cloud tasks. And many experiments demonstrate that the PCIT performance better than state-of-the-art on shape classification and normal estimation tasks. The main contributions of this paper are summarized as following:

- We evaluate the utility of point cloud processing based on various modified Transformer structure.
- We proposed a transformer based framework named PCIT for point cloud learning, which has good performance on point cloud learning task.

Related Work

Figure 2 shows the collected papers, We can see that there are various optimization methods for the original self-attention framework, but only a few of them can be applied to the point cloud domain. In addition, we also researched relevant papers in the newly emerged point cloud domain in recent years to find a suitable base model.

2.1 point cloud processing

In 2016, pointnet [9] proposed by Stanford University kicked off the learning of point cloud processing based on deep learning. Then pointnet++[10], also proposed by Stanford University, uses the network based on hierarchical pointnet and query ball grouping to capture the local structure, which greatly enhances the structural effect.

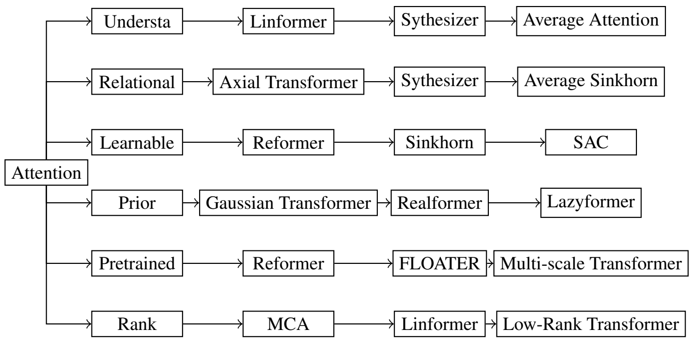

Figure 1: Optimization Directions. Illustration of various optimization directions based on original self-attention.

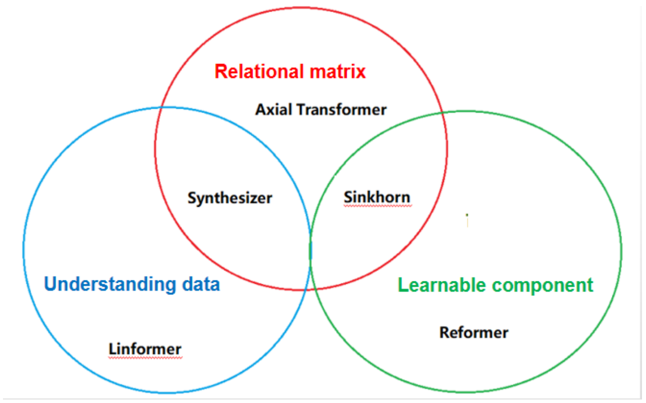

Figure 2: Illustration of of improvement direction of various front transformer frameworks.

Several other works consider how to define the convolution operation on the point cloud. A main method is to convert the point cloud into a regular voxel sequence, because this sequence allows convolution. Segcloud[12] proposed a method for point by point segmentation, which uses trilinear interpolation to map the convolution features of three-dimensional voxels to the point cloud, and maintains global consistency through fully connected conditional random fields. Atzmon et al.[1] proposed a PCNN framework with extension and restriction operators to map between point-based representation and voxel based representation, and convolute voxels to extract point features. Mccnn[3] proposed by Hermosilla et al allows non-uniform sampling of point clouds. In this problem, convolution is regarded as a Monte Carlo integration problem. Similarly, pointconv[15] proposed by Wu et al. Performs 3D convolution through Monte Carlo estimation and importance sampling.

Another method redefines convolution as the operation of irregular point cloud data. [7] A point cloud convolution network, point CNN, is proposed. This network can learn the x-transform from the input points, and then use it to the input features associated with the points, so as to extract the implicit order, rearrange them into the target specification order, and then apply the element multiplication and sum operation of the typical convolution operator to the x-transform to transform the features. Tatarchenko et al.[11] proposed a convolution network, tangent convolution, which can learn surface geometric features from projected virtual tangent images. The SPG [5] proposed by Landrieu et al. Divides the scanned scene into similar elements and establishes a hypergraph structure.

The point cloud processing method based on self attention framework has gradually emerged in the past year. PointASNL[17] proposed noise processing in point cloud processing, using self attention mechanism to update the characteristics of local point groups. Pointgmm[4] usesuses the shape interpolation method of multi-layer perceptron (MLP) separation and self attention separation to update the global point cloud features. PCT[2] uses the concern based transformer framework for point cloud feature processing.

2.2 optimization transformers

In summary, we selected the five best self-attention optimization frameworks shown in Figure 3, and the point cloud processing model PCT based on the self-attention framework, specifically, we will test the replacement of the original self-attention module in the PCT model, which is just facing the problems faced by the self attention framework described above. The first is the data understanding method. Linformer uses the mapping method to reduce the data redundancy, while Synthesizer directly puts forward a new idea, which does not consider the relationship among the features of the input sequence. Both methods are based on the processing of the data features and reduce the data redundancy on the premise of ensuring the accuracy. Then comes the relational matrix method. Axial Transformer changes the method of calculating relational matrix when calculating multidimensional input sequence, Adopts the self-attention calculation method for a single dimension or a single axis. Sinkhorn constructs a sorting network, which sorts the input sequence and then performs self-attention calculation in blocks. In addition, there are three ways of Synthesizer, which all change the object of relational calculation. The problem to be solved here is the time complexity problem described above. Reformer constructs a hash table to re-bucket the input sequence. Sinkhorn creates a learnable component to process the input sequence, which makes the calculation more local and global when focusing on itself. The problem to be solved here is the local and global problem of point cloud feature aggregation described above.

In general, the three improved methods shown in figure 1 are just facing the problems faced by the self attention framework described above. The first is the data understanding method. Linformer uses the mapping method to reduce the data redundancy, while Synthesizer directly puts forward a new idea, which does not consider the relationship among the features of the input sequence. Both methods are based on the processing of the data features and reduce the data redundancy on the premise of ensuring the accuracy. Then comes the relational matrix method. Axial Transformer changes the method of calculating relational

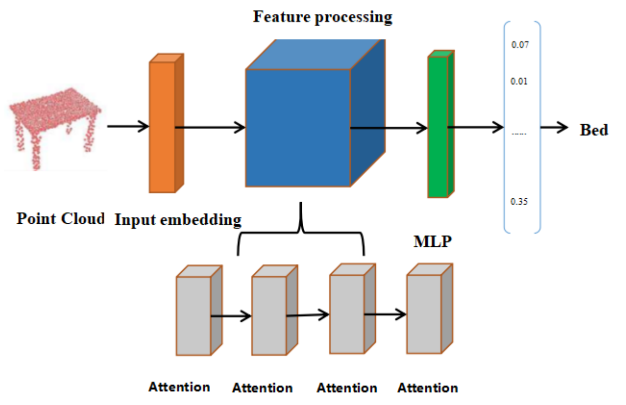

Figure 3: Overall Structure. Illustration of of improvement direction of various front transformer frameworks.

matrix when calculating multidimensional input sequence, Adopts the self-attention calculation method for a single dimension or a single axis. Sinkhorn constructs a sorting network, which sorts the input sequence and then performs self-attention calculation in blocks. In addition, there are three ways of Synthesizer, which all change the object of relational calculation. The problem to be solved here is the time complexity problem described above. Reformer constructs a hash table to re-bucket the input sequence. Sinkhorn creates a learnable component to process the input sequence, which makes the calculation more local and global when focusing on itself. The problem to be solved here is the local and global problem of point cloud feature aggregation described above.

Finally, after testing, we summarized the comparative effects of various transformers, as described in the table above

3.1 Overall

The overall flow framework is shown in Figure 3. The design method is basically the same as the PCT of the basic point cloud processing network. The input feature processing part of the original PCT network adopts the form of multi-head basic self-attention. Since the self-attention structure has the feature that the input and output dimensions remain unchanged (plug anywhere and play anywhere). Then, the core part of PCIT is the self-attention module here. Compared with the basic self-attention of baseline, we have replaced it.

3.2 PCIT

First, the overall structure of PCIT, as shown in Figure 4.

The overall structure is a multi-branch structure. The input point cloud features are used as the module input, and the module output is the output of feature processing results with identical dimensions. The first part is the upper part, and the detailed structure is shown in Figure 5.

The core part is the dynamic self-attentioning part of Figure 5, which adopts the multi-core and multi-scale dynamic self-attentioning structure, and uses multi-scale dynamic convolution to calculate the query matrix, key ma-

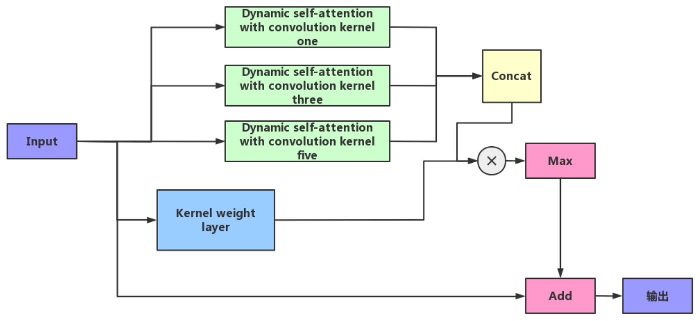
Input
Dynamic self-attention with convolution kernel one
Dynamic self-attention with convolution kernel three
Dynamic self-attention with convolution kernel five
Concat
Kernel weight layer
×
Max
Add
输出

Figure 4: PCIT. Illustration of the overall structure of PCIT.

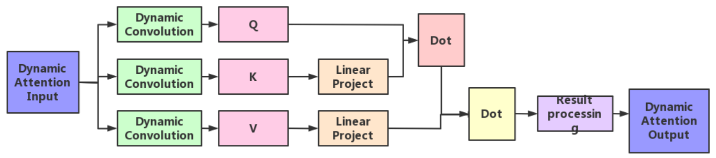

Figure 5: Dynamic Attention of PCIT. Illustration of the dynamic attention of PCIT.

trix and value matrix under their respective kernel sizes, as shown in Figure 5. Assuming that the dynamic self-attention layer with convolution kernel 1 is calculated at this time, the dynamic convolution with convolution kernel 1 is used to calculate the query matrix, key matrix and value matrix respectively. The design of dynamic convolution is shown in Figure 6.

The idea here draws lessons from [6] Dynamic convolution. After receiving the input, the dynamic convolution layer is divided into two paths, one is to calculate the dynamic convolution weight, and the other is to use the obtained dynamic convolution on the input. Firstly, the convolution weight is calculated. Similar to static convolution, dynamic convolution is also composed of k convolution ker-

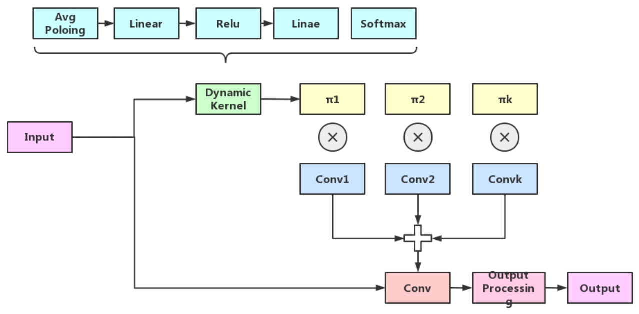

Figure 6: Dynamic Convolution of attention. Illustration of the dynamic convolution of attention.

nels, and it will also be trained by back propagation. Then, the input is extracted by class self-attention, and the weights of k convolution kernels are obtained, and they are linearly weighted into a new convolution kernel, which is the adaptive convolution kernel that pays attention to the input. The core idea of dynamic convolution is that, compared with the static convolution kernel of static convolution, dynamic convolution uses the self-attention method to make each nuclear energy get an adaptive weight, and assign its importance before convolution operation.

Returning to Figure 6, after using dynamic convolution layer to calculate multi-scale query matrix, key matrix and value matrix, two linear projection layers similar to Linformer[14] are constructed to project to the low dimension to reduce the calculation time, and then the query matrix is multiplied by the key matrix and then multiplied by the value matrix like basic self-attention.

Returning to Figure 5, after obtaining the multi-scale dynamic self-attentioning results, the multi-scale results are spliced according to the kernel size. Then, it is the core weight layer in the lower half. The input is still the original input of the module. A multi-layer MLP(Multi-Layer Perceptron) and an adaptive pool layer are constructed. Finally, a Softmax layer is used to get the weight ratio of each multi-scale core, which is regarded as convolution cores of different sizes occupying the overall attention weight. Then, the weight is multiplied by the three results obtained from multi-scale self-attention, and finally, the maximum value of each position in the multi-scale dimension is taken as the representative of the feature of the point.

Finally, after the result is superimposed with the original input, several layers of simple output processing are performed.

Next, explain the reason of model construction from two characteristics of point cloud. See section 5 for specific characteristics. Firstly, the order invariance of point cloud is adopted in this paper. The convolution with a kernel size of 1 is used to ensure the order invariance of point cloud, because the convolution with a kernel size of 1 only considers the features of this point without considering the local part. At the same time, in order to ensure that the point cloud has enough local information, the convolution with a kernel size of 3 and a kernel size of 5 is introduced, and then a multi-kernel weight layer and a maximization layer are constructed. They choose different kernel sizes for each point in order to adapt to the input situation. Moreover, in the recognition features of point clustering, the local neighborhood of points can often give better classification features, because the local neighboring points are often the points with close straight line distance, that is, the points of the same object, so the local features can not be lost in the point cloud processing.

Then the rotation invariance of the point cloud. Here, the dynamic convolution method is adopted to solve this problem. Often, the method to solve the rotation invariance is to rotate the point cloud set uniformly to a set or suitable angle for model analysis, and there are many ways of rotation. Here, convolution operation is regarded as class rotation, and different convolution kernels are regarded as rotation matrices. The importance weight of each rotation matrix, that is, the possibility of calculating the rotation direction of each point, is obtained by using the method of class self-attention. Finally, the final convolution kernels are accumulated. This structure uses this convolution kernel to replace the whole rotation method STN in Pointnet[9].

At this point, this structure ensures the two characteristics of point cloud, namely, sequence invariance and rotation invariance. At the same time, it also has a dynamic effect that is adaptive according to input, and it also has an important locality of point cloud data. Finally, the horizontal comparison experiment results are the best.

Experiments

4.1 Classification on ModelNet40 dataset

We adopted the same test methodology as described in the original PCT, which is using a point cloud classification task based on point cloud processing for testing. That is, the dataset is Modelnet dataset. The goal of the Princeton ModelNet project is to provide researchers in computer vision, robotics, and cognitive science with a dataset that contains a collection of 3D CAD models of many different objects. In order to select which objects to use for the collection and construction of the dataset, statistical information obtained from the SUN database, which provides a ranking table of the most common objects in life theory, was used to determine the object table to build. After building the object table, the names of each object category were searched through an online search engine for possible 3D CAD models that exist on the Web for each object. Then, a quality evaluation tool is built in house and people will be hired on Amazon Mechanical Turk (a forum for posting AI-related tasks) to manually determine if each CAD model searched belongs to the specified category, thus purifying the dataset.

Table 1:Comparative test on shapenet dataset.

Method	Accuracy	Parameters	Memory	Cumulative
Baseline PCT	92.6%	2.94M	39.83	2.18
PCT+Sinkhorn	92.8%	3.33M	45.32	2.28
PCT+Linformer	91.8%	3.33M	42.32	2.28
PCT+Reformer	92.9%	3.07M	43.44	2.21
PCT+Sythesizers	92.4%	3.60M	45.34	2.35

ModelNet40[16] contains 12311 CAD models for 40 object categories, which is widely used for tasks such as point cloud shape classification and surface normal estimation. For a fair comparison, we use the same dataset partitioning as in the PCT paper, with 9843 objects segmented for training and 2468 objects for testing. The same sampling strategy as PointNet was used to sample each object uniformly to 1024 points. During the training period, random translation within (-0.2, 0.2) and random directional scaling within (0.67, 1.5) were used, and random droupout was applied to the inputs as a way to enhance the input data and increase the robustness of the model. During testing, no methods such as data augmentation or manual intervention of voting were used. For all models (base model and model after replacement), the batch size is 32 and undergoes 250 training iterations with an initial learning rate of 0.0001, which is adjusted at the end of each iteration using the common learning rate decay method cosine annealing algorithm.

Specifically, we first reproduced the original text on point cloud classification. We reperformed a round of training and testing, and then modularized the five best-effective attention improvement frameworks described in the previous paper. Finally, these modules are replaced with the original attention modules in the PCT, which is a feasible step since most of the attention frameworks are plug-and-play. Then the dataset-oriented training is re-run separately and the respective classification accuracy is tested after the training. After this, we summarized the effects and analyzed the goodness and generalizability of the effects using various properties of the point cloud, shown in table 1, the overall stucture is shown in figure 4.

4.2 Segmentation task on ShapeNet dataset

We also adopted the same test methodology as described in the original PCT, we performed an experimental evaluation on the ShapeNet dataset[8], which contains 16,880 models.As it mentioned, We divided 14,006 data from the training set as the training set and 2874 as the test set. This dataset has 16 object categories and 50 part labels, each instance contains no fewer than two parts. During training, we apply the random translation between [0.2, 0.2], and random anisotropic scaling between [0.67, 1.5] to augment the input data.

Table 2:Segmentation effect of point cloud processing based on improved self-attention

Model	pIOU	mIOU
PCT	86.4	61.33
PCTSinkhorn	86.8	61.20
PCTLinformer	85.4	59.41
PCTReformer	87.3	63.45
PCTSythesizers	80.5	59.40
PCTAxial	76.8	57.64
PCTMyformer	89.8	65.79

4.3 Computational requirements analysis

The specific analysis of each structure is then developed using the two most important features of point clouds, order invariance and rotation invariance:

• Sinkhorn Transformer: A self-attention improvement based on sorting network. It can keep the order invariance of the point cloud well, because the order of different orders as input will be unified after sorting, and the operation of sorting to chunking can maintain the local rotation invariance of the point cloud to a certain extent, it sacrifices more memory to store the sorting network and parameters, and thus obtains the highest accuracy rate.

• Linformer: The main assumption of its structure is that the-re is redundancy in the features of each point in the input sequence, and the use of linear projection can reduce the time complexity and capture the important features. There is no guarantee whether different features in different orders are discarded.

• Reformer: A self-attention improvement based on hash tables, which is designed in a similar way to Sinkhorn Transformer. Take into account the input sequence ordering. Also this sorting method ensures the order invariance of the point cloud, but it does not actually maintain good local rotation invariance due to the redundancy and disorder within the hash hash bucket.

• Sythesizers: A self-attention improvement that reverses the way the attention matrix is computed, pioneers a way to dis-regard the interaction of input point feature relations, which can be seen in the accuracy rate, and this point can also be paradigmatically used in point cloud processing. Since the input feature processing module is connected with an input embedding module, in fact, the features of each point of the sequence already contain its own features and local neighborhood features at the time of input, and it is sufficient to consider only this point itself and its neighborhood when calculating the relationship of the sequence points.

• Axial Transformer: A self-attention improvement based on dimensional offset, is different from the previous ones in this task. Instead of integrating the point neighborhood features when inputting the feature processing module, this feature and the neighborhood features are directly stitched together to construct a multidimensional feature for input. Obviously this approach has a very low accuracy, it is completely unable to adapt the order invariance of the point cloud, and it performs a secondary processing of the neighborhood features, so the memory occupation is also large.

In the end, it is obvious to perceive that the results are better for the self-attention structures that can guarantee the order invariance of the point cloud, and the results that rely too much on the local and global aspects will not be very good instead. In addition, the scale is also a part of great concern from the point cloud perspective, and it is clear that future work in the direction of attention-based point cloud processing should focus more on the attention matrix calculation. Compared to the multiple optimization directions shown above, point clouds will be more sequential at the network level, which means that attention structures with more homogeneity should take the stage sooner.So the PCIT get the highest accuracy, because this structure ensures the two characteristics of point cloud, namely, sequence invariance and rotation invariance. At the same time, it also has a dynamic effect that is adaptive according to input, and it also has an important locality of point cloud data. Finally, the horizontal comparison experiment results are the best.

Conclusion

In this paper, we propose PCIT, a dynamic attention network, which effectively guarantee the order invariance and the rotation invariance of the point cloud. In addition, We made a systematic investigation on the structure of self-attention. Based on the results of substitution and comparison experiments, we analyzes the applicability of self-attention in the field of point cloud.

References

[1] Matan Atzmon, Haggai Maron, and Yaron Lipman. "Point Convolutional Neural Networks by Extension Operators". In: *ACM Trans. Graph.* 37.4 (July 2018). ISSN: 0730-0301. DOI: 10.1145/3197517.3201301. URL: https://doi.org/10.1145/3197517.3201301.

[2] M. H. Guo et al. "PCT: Point cloud transformer". In: 7.2 (2021), p. 13.

[3] Pedro Hermosilla et al. "Monte Carlo convolution for learning on non-uniformly sampled point clouds". In: *ACM Transactions on Graphics (TOG)* (2018).

[4] A. Hertz et al. "PointGMM: a Neural GMM Network for Point Clouds". In: *arXiv*. 2020.

[5] Loic Landrieu and Martin Simonovsky. "Large-scale point cloud semantic segmentation with superpoint graphs". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4558–4567.

[6] Juho Lee et al. "Set transformer: A framework for attention-based permutation-invariant neural networks". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 3744–3753.

[7] Y. Li et al. "PointCNN". In: (2018).

[8] Jonathan Pilault, Amine Elhattami, and Christopher Pal. "Conditionally adaptive multi-task learning: Improving transfer learning in nlp using fewer parameters & less data". In: *arXiv preprint arXiv:2009.09139* (2020).

[9] C. R. Qi et al. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[10] Charles R. Qi et al. "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space". In: (2017).

[11] Maxim Tatarchenko* et al. "Tangent Convolutions for Dense Prediction in 3D". In: *CVPR* (2018).

[12] L. P. Tchapmi et al. "SEGCloud: Semantic Segmentation of 3D Point Clouds". In: (2017).

[13] Ashish Vaswani et al. "Attention Is All You Need". In: *arXiv* (2017).

[14] Sinong Wang et al. "Linformer: Self-attention with linear complexity". In: *arXiv preprint arXiv:2006.04768* (2020).

[15] W. Wu, Z. Qi, and F. Li. "PointConv: Deep Convolutional Networks on 3D Point Clouds". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[16] Zhirong Wu et al. "3d shapenets: A deep representation for volumetric shapes". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1912–1920.

[17] X. Yan et al. "PointASNL: Robust Point Clouds Processing using Nonlocal Neural Networks with Adaptive Sampling". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.