



Learning Key Features Transformer Network for Point Cloud Processing

Guobang You, Yikun Hu, Yimei Liu, Haoyan Liu, and Hao Fan^(✉)

College of Computer Science and Technology, Ocean University of China, Qingdao, China

fanhao@ouc.edu.cn

Abstract. Due to the unordered and irregular nature of point cloud data, it is challenging for neural networks to learn from it. Attention mechanisms have shown promising results in point cloud processing. It is also inherently permutation-invariant when dealing with a set of points, which makes it ideal for point cloud learning. In this paper, an attention-based encoder-decoder architecture called KFT-Net (Key Features Transformer) is designed for point cloud classification and segmentation tasks. KFT-Net has improved upon previous methods by capturing long-range contextual information, preserving important attention scores, and utilizing convolutional neural networks to extract local features. Additionally, to enhance the computational efficiency, we introduce the Top-k operation into the attention mechanism and utilize the average pooling operation to improve attention score calculation and feature extraction efficiency. Extensive experiments validate the effectiveness of the KFT-Net, demonstrating impressive performance in point cloud classification and segmentation tasks.

Keywords: Point cloud · Self-attention · Feature extraction · Classification · Segmentation

1 Introduction

Point cloud data is represented in coordinate form, where each point contains positional information and other attributes, such as normals and colors. Point cloud classification and segmentation tasks are crucial in various applications. However, point cloud data is often sparse, irregular, and noisy, which makes it challenging to extract meaningful information using traditional methods.

This paper proposes an encoder-decoder architecture based on self-attention [20] called KFT-Net, which aims to improve point cloud classification and segmentation performance. KFT-Net can enhance the ability of long-range dependency capturing, global feature modeling, and local feature extraction. To improve the vanilla attention mechanism, the Top-k operation is incorporated in the attention matrix, and the average pooling operation is applied on $K(key)$ and $V(values)$ to improve attention score calculation and feature extraction efficiency. These include:

Top-k Operation. We use the Top-k operation, where at each position the top k elements with the highest attention weight are selected. By ignoring lower weights, the model can focus on the most important parts for processing. To normalize and preserve the effectiveness of the chosen attention weights, we normalize the highest weight values such that their sum is equal to or close to 1. This normalization ensures that the attention probabilities have the same properties. Using the filtered attention weights, we perform a weighted sum over the corresponding value vector for each location. This operation directs the model’s attention to values related to key information. Thus, it helps to create a more accurate representation.

Average Pooling Operation. We also perform an average pooling operation before calculating the attention matrix, we perform the average pooling operation on K and V to alleviate computational overhead and focus on higher attention weights. This operation involves a weighted average between attention weights and value vectors, thereby allowing the model to emphasize key features and disregard irrelevant or noisy influences. Moreover, by summarizing multiple value vectors while preserving important information, the average pooling operation helps capture global information and generate more comprehensive and representative representations.

Experimental results show that KFT-Net achieves good performance in point cloud classification and segmentation tasks. The main contributions of this paper are summarized as follows:

- 1) We propose a novel transformer-based point cloud learning framework called KFT-Net, which is designed to better handle the irregularity and disorder of point cloud data.
- 2) We perform a Top-k operation on the attention matrix, which allows us to extract the most crucial point cloud features and improve learning efficiency compared to the original self-attention module.
- 3) We apply the average pooling operation on K and V , which can reduce the number of points considered during attention computation. By aggregating multiple points into a representative point, it helps to capture global information and summarize the overall context of the entire point cloud.

2 Related Work

2.1 Point-Based Deep Learning

Deep learning has shown remarkable performance in point cloud classification and segmentation. PointNet [13] is introduced as a novel framework for directly processing unordered point clouds without pre-processing or transformations. In the years following PointNet, many techniques such as Pointnet++ [14], PointCNN [10], and PointConv [23] are developed to improve the accuracy of point cloud classification and segmentation. PointWeb [28] addresses the challenges of arbitrary point order and complex feature extraction through innovative

techniques. PointGrid [9] converts point cloud data into a structured grid representation to achieve efficient feature extraction for various tasks. PointNeXt [16] improves the PointNet++ framework by introducing Inverted Residual Bottleneck and separable MLPs, achieving superior results in point cloud classification and segmentation tasks.

2.2 Transformer for Point Cloud

Point cloud processing faces challenges in preserving spatial relationship and capturing long-range dependencies. To address these challenges, attention mechanisms are proposed. Self-attention [20] is designed to capture global and local features simultaneously. Transformers have shown impressive performance in both the field of NLP (Natural Language Processing) [11, 20] and CV (Computer Vision) [2, 7, 21]. PointASNL [27] use an adaptive sampling strategy to exploit higher-order local structures in graphs. PCT [4] hierarchically extracts local features using point-to-cluster operations. PT [15] aggregates contextual information using prism-based pooling methods. Pointformer [29] encodes point relationships using multi-head attention mechanisms after translating point clouds to continuous tensors.

3 Point Cloud Processing with KFT-Net

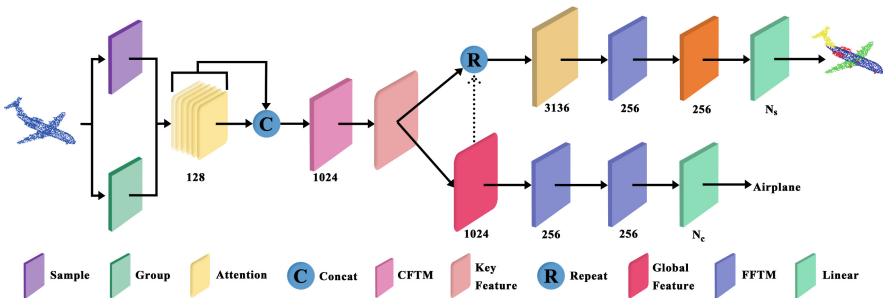


Fig. 1. KFT-Net architecture. The encoder consists of a Group module, a Sample module, and Contextual-Aware Attention Module (CAAM). The decoder is mainly composed of a Coarse Feature Transfer Module (CFTM) and Fine Feature Transfer Module (FFTM), which outputs segmentation and classification results. And the number of output channels is marked below each module. CFTM combines Linear, BatchNorm and ReLU layers. FFTM combines Linear, BatchNorm, ReLU and dropout layers.

3.1 Overall Architecture

The overall architecture of KFT-Net is shown in Fig. 1. KFT-Net is an encode-decoder architecture. The encoder consists of a Group module, a Sample module, and Contextual-Aware Attention Module (CAAM). The decoder is mainly composed of a Coarse Feature Transfer Module (CFTM) and Fine Feature Transfer Module (FFTM), which produces the final feature representation for segmentation and classification results.

Given the input point cloud $P \in \mathbb{R}^{N \times d}$, where N points have d -dimensional features, the encoder learns d_c -dimensional reduction embedding feature $F_c \in \mathbb{R}^{N \times d_c}$ through the Group and Sample modules [4]. The per-point feature representation $F_o \in \mathbb{R}^{N \times d_o}$ is obtained by concatenating the CAAM outputs.

$$\begin{aligned} F_1 &= CAAM_1(F_c), \\ F_i &= CAAM_i(F_{i-1}), i = 2, 3, 4, 5, 6, \\ F_o &= \text{concat}(F_1, F_2, F_3, F_4, F_5, F_6) \cdot W_o, \end{aligned} \quad (1)$$

where $CAAM_i$ represents the i -th CAAM layer, each layer has the same output dimension and W_o is the learnable parameters. A detailed explanation of the CAAM layer will be elaborated in Sect. 3.2. Various implementations of input embedding and attention mechanisms will be described in detail below. We concatenate CAAM outputs to learn per-point feature representation. Thus, global feature representation F_g is extracted based on the learned per-point features.

We classify the point cloud P into N_c specific categories (e.g. airplane, car, chair) using the extracted global features for classification tasks. The global features F_g are fed into the decoder, which consists of CFTM and FFTM. Finally, we predict the classification score $S \in \mathbb{R}^{N_c}$ (where N_c is the number of object categories), and the label of the point cloud is assigned to the class with the maximal score.

In the segmentation task, we find the corresponding category for each point. As it involves categorizing points, we extract global and per-point features from the point cloud. Then, we merge the global feature F_g and the per-point feature F_o together to further segment the point cloud into N_c classes (e.g. airplane head, airplane tail, airplane wings). In the decoder, after fusing the global features and per-point features obtained from the encoder, we use a CFTM and a FFTM. Finally, we apply a linear layer for score prediction. Specifically, after the decoding process, we can obtain the predicted scores $S \in \mathbb{R}^{N \times N_c}$ for each point, and we determine the label of the point based on the highest score.

3.2 Contextual-Aware Attention Module

When dealing with point clouds, the extracted point is regarded as a word, and the entire point cloud is treated as a sentence. In this case, the point cloud P is embedded into a d_c -dimensional space represented by $F_c \in \mathbb{R}^{N \times d_c}$, where N is the number of points and d_c is set to 128 for computational efficiency. In our KFT-Net, we propose Contextual-Aware Attention Module (CAAM) to

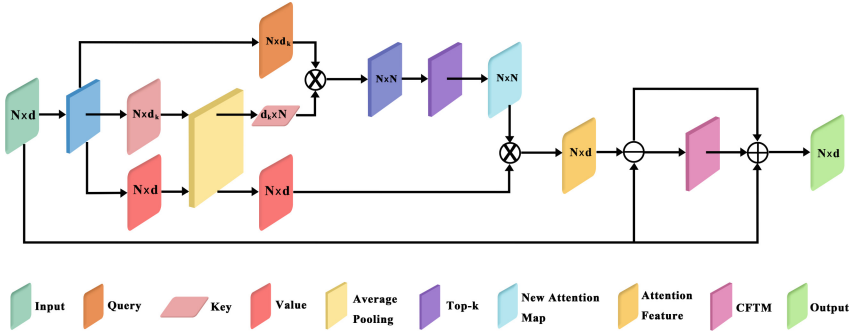


Fig. 2. Architecture of Contextual-Aware Attention Module.

enhance point cloud processing, where CAAM is decomposed of self-attention mechanism, Top-K operation and average pooling operation. The structure of CAAM is illustrated in Fig. 2.

Self-Attention Mechanism. The self-attention mechanism involves transforming each element in the input sequence into three vectors: *query*, *key*, and *value*. The similarity between the *query* vector and *key* vector is then calculated to obtain attention weights. Finally, the attention weights are multiplied with the *value* vector and summed up. Referring to literature [20], let Q, K, V be the query matrix, key matrix, and value matrix generated by linear transformation of the input features $F_{in} \in \mathbb{R}^{N \times d_c}$ as follows:

$$\begin{aligned} \text{Attn}(Q, K, V) &= F_{in} \cdot (W_q, W_k, W_v), \\ Q, K &\in \mathbb{R}^{N \times d_k}, V \in \mathbb{R}^{N \times d_c}, \\ W_q, W_k &\in \mathbb{R}^{d_c \times d_k}, W_v \in \mathbb{R}^{d_c \times d_c}, \end{aligned} \quad (2)$$

where Attn is the vanilla self-attention mechanism, W_q, W_k and W_v are shared learnable linear transformation operations, d_k is the dimension of the *query* vector and *key* vector, and d_c is the dimension of the *value* vector.

Average Pooling Operation. To obtain features F_{sa} , we perform an average pooling [7] operation on K' and V' . The specific operation is as follows:

$$F_{sa}(Q', K', V') = \text{Attn}(F_{in} \cdot W_q, P_t(F_{in} \cdot W_k), P_t(F_{in} \cdot W_v)), \quad (3)$$

where Attn is the vanilla self-attention mechanism. P_t is an average pooling operation with stride t for transforming the spatial dimension to reduce computational cost. Then, we can use the query and key matrices to calculate the attention weights through matrix dot-product:

$$\bar{A}_{sa} = \bar{S}_{i,j} = Q' \cdot K'^T. \quad (4)$$

These weights are then normalized to obtain $A_{sa} = S_{i,j}$. Here, we depart from the conventional attention mechanism and instead employ the l_1 -norm used in the [4] network for regularization:

$$\begin{aligned} \tilde{A}_{sa} = \tilde{S}_{i,j} &= \frac{\tilde{S}_{i,j}}{\sqrt{d'_k}}, \\ A_{sa} = S_{ij} &= \frac{\exp(\tilde{S}_{i,j})}{\sum_k \exp(\tilde{S}_{i,k})}. \end{aligned} \quad (5)$$

Average pooling operation in Fig. 2 is performed by taking the weighted average of the value vectors using the corresponding attention weights to obtain the transformed output. After the operation of weighted average pooling operation, we obtain the pooled K' and V' . The K' summarizes the feature representations of keys with higher attention weights, while the V' summarizes the feature representations of values with higher attention weights. In this way, during the calculation process of the attention matrix A and V' , each value vector V' is multiplied by its corresponding attention weight, and then all the weighted value vectors are summed up. The final result is a weighted average value vector that represents the aggregated information of the query vector Q' .

Top-K Operation. After implementing the average pooling operation on self-attention mechanism, an attention weight matrix A_{sa} is obtained. Different from the random dropout strategy that randomly drops scores, an adaptive selection of the Top-k operation contributing scores on matrix A_{sa} is implemented here, aiming to retain the most important components and discard irrelevant ones. Here, k is a tunable parameter that dynamically controls the sparsity level, which is formally obtained through the weighted average of appropriate scores, we set $k = N/2$, where N is the number of elements in the input sequences. Therefore, for softmax computation, only the top k values within the range $[\delta 1, \delta 2]$ are normalized for each row of matrix A_{sa} . For other elements below the top k scores, their probabilities are replaced with 0 using a scatter function at the given indices. This dynamic selection shifts attention from dense to sparse, which is written by the following equation:

$$A_{topk} = [T_k(S)]_{i,j} = \begin{cases} S_{i,j}, & S_{i,j} \in Top - k(row, j), \\ 0, & otherwise. \end{cases} \quad (6)$$

where T_k represents the Top-k operation.

After adding the Top-k operation in Fig. 2, we can select specific elements with the highest attention weights. By dropping lower attention weights, noise or irrelevant information could be restrained, and it is of benefit to improve the model's robustness and accuracy. For large-scale point cloud data, we can reduce computational costs and storage requirements by retaining more important information. The output feature F_r is the weighted sum of the value vectors

using the refined attention weights:

$$F_r = A_{topk} \cdot V'. \quad (7)$$

3.3 Output

In the final output, we refer to the methods of [24] and [4]. As shown in Fig. 2, during the decoding process, we perform element-wise subtraction between encoded features F_r and the input features F_{in} . The subtracted result is then passed into the CFTM and FFTM.

We shift offsets and add them to query features, which boosts sensitivity to other points and helps in calculating similarities. This operation is performed on all query features and yields attention distributions for each one. Offset vectors enhance positional information and make the attention mechanism more flexible and accurate, ultimately improving performance in point cloud classification and segmentation tasks.

4 Experiments

We conduct extensive experiments on three public benchmarks to evaluate the effectiveness of the KFT-Net. We use the same softmax cross-entropy loss function as [4]. KFT-Net is trained with the Adam (Adaptive Moment Estimation) optimizer using a learning rate of 10^{-4} and weight decay of 10^{-3} .

Table 1. ModelNet40 shape classification. We compare our model with other networks. ‘P’ denotes the number of input points, and ‘N’ denotes the corresponding normals.

Method	Input	Accuracy
PointNet [13]	P, N	89.4
PointMLP [12]	P	91.1
Kd-Net [5]	P	91.5
PointNet++ [14]	P, N	91.9
PointGrid [9]	P	92.0
PointWeb [28]	P	92.2
PointCNN [10]	P	92.4
PointConv [23]	P, N	92.5
KPConv [18]	P	92.7
DGCNN [24]	P	92.9
PointASNL [27]	P	92.9
PCT [4]	P	92.9
Ours	P	93.4

4.1 Shape Classification on ModelNet40

We use ModelNet40 [25] to evaluate the performance on point cloud classification tasks. The dataset is divided into a training set and a test set, following a standard evaluation protocol for a fair comparison. Each object is uniformly sampled to 1,024 points using the same sampling strategy as [13]. Our model is trained for 250 epochs with an initial learning rate of 10^{-4} , employing cosine decay per epoch and a batch size of 16. The experimental results are shown in Table 1. Compared to [13] and [4], KFT-Net makes a 4.0% and 0.5% improvement respectively. KFT-Net achieves the best result of 93.4% overall accuracy.

Table 2. Comparison on the ShaperNet part segmentation dataset. ‘pIoU’ means part-average Intersection-over-Union.

Method	pIoU	airplane	bag	cap	car	chair	earphone	guitar	knife	lamp	laptop	motorbike	mug	pistol	rocket	skateboard	table
Kd-Net [5]	82.3	80.1	74.6	74.3	70.3	88.6	73.5	90.2	87.2	81.0	94.9	57.4	86.7	78.1	51.8	69.9	80.3
PointNet [13]	83.7	83.4	78.7	82.5	74.9	89.6	73.0	91.5	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
PointNet++ [14]	85.1	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6
DGCNN [24]	85.2	84.0	83.4	86.7	77.8	90.6	74.7	91.2	87.5	82.8	95.7	66.3	94.9	81.1	63.5	74.5	82.6
SFCNN [17]	85.4	83.0	83.4	87.0	80.2	90.1	75.9	91.1	86.2	84.2	96.7	69.5	94.8	82.5	59.9	75.1	82.9
PointConv [23]	85.5	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SGPN [22]	85.5	80.4	78.6	78.8	71.5	88.6	78.0	90.9	83.0	78.8	95.8	77.8	93.8	87.4	60.1	92.3	89.4
PT [15]	85.7	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
PCT [4]	85.8	84.8	81.3	88.5	81.2	90.9	71.5	91.2	87.9	86.3	95.6	63.3	95.8	83.2	61.3	77.6	83.7
PointCNN [10]	85.8	84.1	86.5	86.0	80.8	90.6	79.7	92.3	88.4	85.3	96.1	77.2	95.2	84.2	64.2	80.0	83.0
PointASNL [27]	86.0	84.1	84.7	87.9	79.7	92.2	73.7	91.0	87.2	84.2	95.8	74.4	95.2	81.0	63.0	76.3	83.2
Ours	86.2	85.0	82.4	88.4	77.8	92.5	72.0	91.4	88.6	84.3	95.6	71.9	95.9	82.0	64.8	80.7	82.9

4.2 Object Part Segmentation on ShapeNet

We conduct point cloud segmentation experiments using the ShapeNet [26] dataset. According to [13], all models are downsampled to 2048 points, preserving point-to-part annotations. Table 2 shows the class-wise segmentation results. The evaluation metric used is partial mean pIoU, which provides overall and per-object category performance metrics. The results indicate that our KFT-Net outperforms [13] and [4] by 2.5% and 0.4%, respectively. KFT-Net achieves the best performance with pIoU of 86.2%. Figure 3 illustrates segmentation examples using KFT-Net.

Table 3. Comparison on the S3DIS semantic segmentation dataset tested on Area 5. ‘OA’ means overall point accuracy. ‘mIoU’ means Intersection-over-Union.

Method	OA	mIoU	ceiling	floor	wall	beam	column	window	door	chair	table	bookcase	sofa	board	clutter
PointNet [13]	78.5	49.6	88.8	97.3	69.8	0.1	3.9	46.3	10.8	59.0	52.6	5.9	40.3	26.4	33.2
PointNet++ [14]	–	53.2	90.8	96.5	74.1	0.0	5.8	43.6	25.4	69.2	76.9	21.5	55.6	49.3	41.9
DGCNN [24]	84.1	56.1	–	–	–	–	–	–	–	–	–	–	–	–	–
SPG [8]	85.5	62.1	89.4	96.9	78.1	0.0	42.8	48.9	61.6	84.7	75.4	69.8	52.6	2.1	52.2
PointCNN [10]	88.1	65.4	92.3	98.2	79.4	0.0	17.6	22.8	62.1	74.4	80.6	31.7	66.7	62.1	56.7
PointWeb [28]	87.3	66.7	92.0	98.5	79.4	0.0	21.1	59.7	34.8	76.3	88.3	46.9	69.3	64.9	52.5
PCT [4]	87.6	67.1	92.5	98.4	80.6	0.0	19.4	61.6	48.0	76.6	85.2	46.2	67.7	67.9	52.3
Ours	88.3	67.9	92.8	98.5	80.0	0.0	20.6	63.5	51.2	75.8	84.1	47.1	68.3	66.4	51.6

4.3 Semantic Segmentation on S3DIS

The S3DIS [1] dataset is a dataset used for 3D segmentation and understanding of indoor scenes. We used the S3DIS dataset for the task of semantic segmentation of indoor scenes. We use the same method as [13] for semantic segmentation of the S3DIS dataset, but we made improvements in the network model. Our KFT-Net outperforms [13] in terms of mIoU, and OA by 18.3% and 9.8% respectively. At the same time, it is faster in terms of speed. The speed improvement is due to the use of Top-k operation in the SA block of KFT-Net, which reduces computation. As the model expands, the performance of KFT-Net can gradually improve. As shown in Table 3, our KFT-Net achieves the best results with mIoU and OA reaching 67.9% and 88.3%.

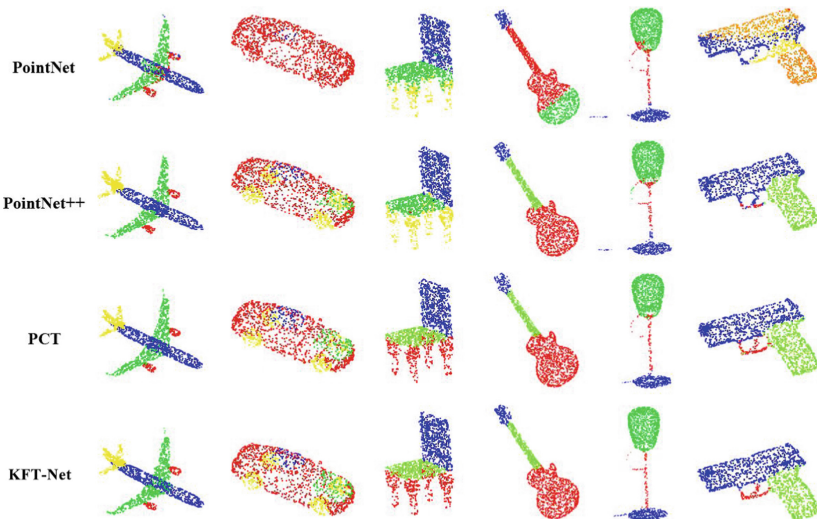


Fig. 3. Qualitative results of PointNet, PointNet++, PCT, KFT-Net.

4.4 Ablation Studies

Effectiveness of Top-k Operation. With the Top-k operation, we focus the attention mechanism on the most relevant elements in the input sequence while discarding irrelevant elements. This reduces complexity and improves accuracy. As shown in Table 4, the Top-k operation achieves accuracy improvements on all three tasks, which verifies the advantages of the proposed Top-k operation. For example, 0.7% Acc has been improved on ModelNet40, and 0.6% pIoU has been increased on ShapeNet.

Table 4. Ablation study on ModelNet40, ShapeNet and S3DIS benchmarks with different modules.

Datasets	Baseline	+Top-k	+Average pooling	KFT-Net
ModelNet40 Acc(%)	92.9	93.6	92.2	93.4
ShapeNet pIoU(%)	85.8	86.4	85.4	86.2
S3DIS mIoU(%)	67.1	68.2	66.5	67.9

Effectiveness of Average Pooling Operation. We leverage the average pooling operation to narrow the focus to only the most relevant key-value pairs for the computation. This increases speed and maintains high-quality attention. As shown in Table 4, after adding average pooling operation, the accuracy rate has a small drop, but the overall operation speed has been greatly improved, and the benefits brought by the operation efficiency are higher than the accuracy rate benefits with a small drop.

5 Conclusion

In this paper, KFT-Net is proposed to point cloud classification and segmentation, which has shown effective and robust performance. By utilizing the attention mechanism to capture contextual information between points, we can extract key features more effectively. To improve the computation of attention scores and key feature extraction, we incorporate Top-k operation into the attention matrix and average pooling operation on K and V . Experimental results demonstrate that the KFT-Net is capable of learning excellent semantic features and achieving high performance in point cloud classification and segmentation tasks.

Acknowledgment. This work is supported in part by the National Natural Science Foundation of China (Grant No. 42106193, 41927805).

References

1. Armeni, I., et al.: 3D semantic parsing of large-scale indoor spaces. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2016)
2. Chen, X., et al.: Learning a sparse transformer network for effective image deraining. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2023)
3. Dai, Z., et al.: Transformer-xl: attentive language models beyond a fixed-length context. arXiv preprint [arXiv:1901.02860](https://arxiv.org/abs/1901.02860) (2019)
4. Guo, M.-H., et al.: Pct: point cloud transformer. *Comput. Visual Media* **7**, 187–199 (2021)
5. Klokov, R., Lempitsky, V.: Escape from cells: deep kd-networks for the recognition of 3d point cloud models. In: Proceedings of the IEEE International Conference on Computer Vision (2017)

6. Lee, J., et al.: BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* **36**(4), 1234–1240 (2020)
7. Li, J., et al.: Next-vit: next generation vision transformer for efficient deployment in realistic industrial scenarios. arXiv preprint [arXiv:2207.05501](https://arxiv.org/abs/2207.05501) (2022)
8. Landrieu, L., Simonovsky, M.: Large-scale point cloud semantic segmentation with superpoint graphs. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018)
9. Le, T., Duan, Y.: Pointgrid: a deep network for 3D shape understanding. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018)
10. Li, Y., et al.: Pointcnn: convolution on x-transformed points. *Adv. Neural Inf. Process. Syst.* **31** (2018)
11. Lin, Z., et al.: A structured self-attentive sentence embedding. arXiv preprint [arXiv:1703.03130](https://arxiv.org/abs/1703.03130) (2017)
12. Ma, X., et al.: Rethinking network design and local geometry in point cloud: a simple residual MLP framework. arXiv preprint [arXiv:2202.07123](https://arxiv.org/abs/2202.07123) (2022)
13. Qi, C.R., et al.: Pointnet: deep learning on point sets for 3D classification and segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017)
14. Qi, C.R., et al.: Pointnet++: deep hierarchical feature learning on point sets in a metric space. *Adv. Neural Inf. Process. Syst.* **30** (2017)
15. Pan, X., et al.: 3D object detection with pointformer. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021)
16. Qian, G., et al.: Pointnext: revisiting pointnet++ with improved training and scaling strategies. *Adv. Neural Inf. Process. Syst.* **35**, 23192–23204 (2022)
17. Rao, Y., Lu, J., Zhou, J.: Spherical fractal convolutional neural networks for point cloud recognition. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019)
18. Thomas, H., et al.: Kpconv: flexible and deformable convolution for point clouds. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019)
19. Tchapmi, L., et al.: Segcloud: semantic segmentation of 3D point clouds. In: *2017 International Conference on 3D vision (3DV)*. IEEE (2017)
20. Vaswani, A., et al.: Attention is all you need. *Adv. Neural Inf. Process. Syst.* **30** (2017)
21. Wu, B., et al.: Visual transformers: token-based image representation and processing for computer vision. arXiv preprint [arXiv:2006.03677](https://arxiv.org/abs/2006.03677) (2020)
22. Wang, W., et al.: SGPN: similarity group proposal network for 3d point cloud instance segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018)
23. Wu, W., Qi, Z., Li, F.: Pointconv: deep convolutional networks on 3d point clouds. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019)
24. Wang, Y., et al.: Dynamic graph cnn for learning on point clouds. *ACM Trans. Graph. (tog)* **38**(5), 1–12 (2019)
25. Wu, Z., et al.: 3D shapenets: a deep representation for volumetric shapes. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015)
26. Yi, L., et al.: A scalable active framework for region annotation in 3d shape collections. *ACM Trans. Graph. (ToG)* **35**(6), 1–12 (2016)

27. Yan, X., et al.: Pointasnl: robust point clouds processing using nonlocal neural networks with adaptive sampling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2020)
28. Zhao, H., et al.: Pointweb: enhancing local neighborhood features for point cloud processing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2019)
29. Zhao, H., et al.: Point transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (2021)