Attention-Based Point Cloud Edge Sampling

Chengzhi Wu[1] Junwei Zheng[1] Julius Pfrommer[2,3] Jürgen Beyerer[1,2,3]
[1]Karlsruhe Institute of Technology, Germany [2]Fraunhofer IOSB, Germany
[3]Fraunhofer Center for Machine Learning, Germany
{chengzhi.wu, junwei.zheng}@kit.edu, {julius.pfrommer, juergen.beyerer}@iosb.fraunhofer.de

Abstract

Point cloud sampling is a less explored research topic for this data representation. The most commonly used sampling methods are still classical random sampling and farthest point sampling. With the development of neural networks, various methods have been proposed to sample point clouds in a task-based learning manner. However, these methods are mostly generative-based, rather than selecting points directly using mathematical statistics. Inspired by the Canny edge detection algorithm for images and with the help of the attention mechanism, this paper proposes a non-generative Attention-based Point cloud Edge Sampling method (APES), which captures salient points in the point cloud outline. Both qualitative and quantitative experimental results show the superior performance of our sampling method on common benchmark tasks.

1. Introduction

Point clouds are a widely used data representation in various domains including autonomous driving, augmented reality, and robotics. Due to the typically large amount of data, the sampling of a representative subset of points is a fundamental and important task in 3D computer vision.

Apart from random sampling (RS), other classical point sampling methods including grid sampling, uniform sampling, and geometric sampling have been well-established. Grid sampling samples points with regular grids and thus cannot control the number of sampled points exactly. Uniform sampling takes the points in the point cloud evenly and is more popular due to its robustness. Farthest point sampling (FPS) [9, 29] is the most famous of them and has been widely used in many current methods when downsampling operations are required [19, 34, 47, 52, 56]. Geometric sampling samples points based on local geometry, such as the curvature of the underlying shape. Another example of Inverse Density Importance Sampling (IDIS) [11] samples points whose distance sum values with neighbors are smaller. But this method requires the point cloud to have a high density throughout, and it performs even worse when the raw point cloud has an uneven distribution.

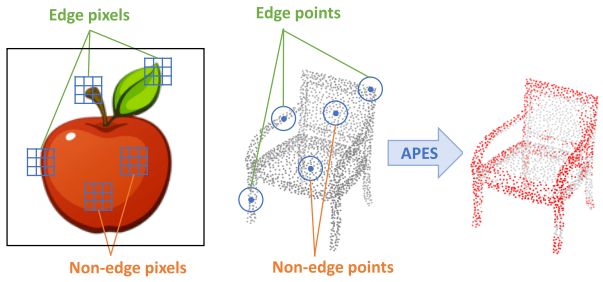

Figure 1. Similar to the Canny edge detection algorithm that detects edge pixels in images, our proposed APES algorithm samples edge points which indicate the outline of the input point clouds. The blue grids/spheres represent the local patches for given center pixels/points.

In addition to the above mathematical statistics-based methods, with the development of deep learning techniques, several neural network-based methods have been proposed for task-oriented sampling, including S-Net [8], SampleNet [16], DA-Net [21], etc. They use simple multi-layer perceptrons (MLPs) to generate new point cloud sets of desired sizes as resampled results, supplemented by different post-processing operations. MOPS-Net [36] learns a sampling transformation matrix first, and then generates the sampled point cloud by multiplying it with the original point cloud. However, all these methods are generative-based, rather than selecting points directly. On the other hand, there is an increasing body of work designing neural network-based local feature aggregation operators for point clouds. Although some of them (e.g., PointCNN [19], PointASNL [52], GSS [53]) decrease the point number while learning latent features, they can hardly be considered as sampling methods in the true sense as no real spatial points exist during the processing. Moreover, none of the above methods consider shape outlines as special features.

In this paper, we propose *a point cloud edge sampling method that combines neural network-based learning and mathematical statistics-based direct point selecting.* One key to the success of 2D image processing with neural networks is that they can detect primary edges and use them to form shape contours implicitly in the latent space [55].

Inspired by that insight, we pursue the idea of focusing on salient outline points (edge points) for the sampling of point cloud subsets for downstream tasks. Broadly speaking, edge detection may be considered a special sampling strategy. Hence, by revisiting the Canny edge detection algorithm [4] which is a widely-recognized classical edge detection method for images, we propose our attention-based point cloud edge sampling (APES) method for point clouds. It uses the attention mechanism [42] to compute correlation maps and sample edge points whose properties are reflected in these correlation maps. We propose two kinds of APES with two different attention modes. Based on neighbor-to-point (N2P) attention which computes correlation maps between each point and its neighbors, local-based APES is proposed. Based on point-to-point (P2P) attention which computes a correlation map between all points, global-based APES is proposed. Our proposed method selects sampled points directly, and the intermediate result preserves the point index meaning so they can be visualized easily. Moreover, our method can downsample the input point cloud to any desired size.

We summarize our contributions as follows:

- A point cloud edge sampling method termed APES that combines neural network-based learning and mathematical statistics-based direct point selecting.
- Two variants of local-based APES and global-based APES, by using two different attention modes.
- Good qualitative and quantitative results on common point cloud benchmarks, demonstrating the effectiveness of the proposed sampling method.

2. Related Work

2.1. Point Cloud Sampling

In the past decades, non-learning-based sampling methods are mostly used for point cloud sampling. FPS [9, 29] is the most widely used sampling method, which selects the farthest points iteratively. FPS is easy to implement and has been frequently used in neural networks that aggregate local features, e.g., PointNet++ [34], PointCNN [19], PointConv [47], and RS-CNN [24]. Besides, RS has also been adopted to process large-scale point clouds with great computational efficiency in lots of works, including VoxelNet [60], RandLA-Net [13] and P2B [35]. A more recently proposed method of IDIS [11] defines the inverse density importance of a point by simply adding up all distances between the center point and its neighbors, and samples points whose sum values are smaller.

Recently, learning-based sampling methods show better performances on point cloud sampling when trained in a task-oriented manner. The pioneering work of S-Net [8] generates new point coordinates directly from the global representation. Its subsequent work of SampleNet [16] further introduces a soft projection operation for better point approximation in the post-processing step. Alternatively, DA-Net [21] extends S-Net with a density-adaptive sampling strategy, which decreases the influence of noisy points. By learning a sampling transformation matrix, MOPS-Net [36] multiplies it with the original point cloud to generate a new one as the sampled point cloud. CPL [31] samples points by investigating the output in the max-pooling layer. Replacing the MLP layers in S-Net with several self-attention layers, PST-NET [43] reports better performances on trained tasks. Its subsequent work of LighTN [44] proposes a lightweight Transformer framework for resource-limited cases.

2.2. Deep Learning on Point Clouds

Prior to the emergence of PointNet [33], deep learning-based methods for point cloud analysis are usually multi-view-based [1, 3, 17, 40] or volumetric-based [14, 18, 28]. PointNet [33] is the first DL-based method that learns directly on points and it uses point-wise MLP to extract global features. Its subsequent work of PointNet++ [34] further considers local information. Convolution-based methods [19, 22, 41, 46, 47, 49, 58] bring the convolution operation into point cloud feature learning. For example, PointConv [47] and KPConv [41] propose point-wise convolution operators with which points are convoluted with neighbor points. Graph-based methods [5, 20, 23, 24, 45, 50, 57] analyze point clouds by using graph structure. For example, Simonovsky et al. [38] treat each point as a graph vertex and apply graph convolution. In DGCNN [45], EdgeConv blocks update the neighbor information dynamically based on dynamic graphs. More recently, Attention-based methods [2,7,10,12,13,26,27,32] are starting to trend. PCT [12] pioneers this direction by replacing the encoder layers in the PointNet framework with self-attention layers, while PT [59] is based on U-Net [37]. 3DCTN [27] uses offset attention blocks, while a deformable self-attention module is proposed in SA-Det3D [2], and a dual self-attention module is proposed in 3DPCT [26]. Stratified Transformer [15] additionally samples distant points as the key input to capture long-range contexts.

3. Methodology

3.1. Revisiting Canny Edge Detection on Images

The Canny edge detector uses a multi-stage algorithm to detect edges in images. It consists of five steps: (i) Apply Gaussian filter to smooth the image; (ii) Find the intensity gradients of the image; (iii) Apply gradient magnitude thresholding or lower bound cut-off suppression; (iv) Apply double threshold to determine potential edges; (v) Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

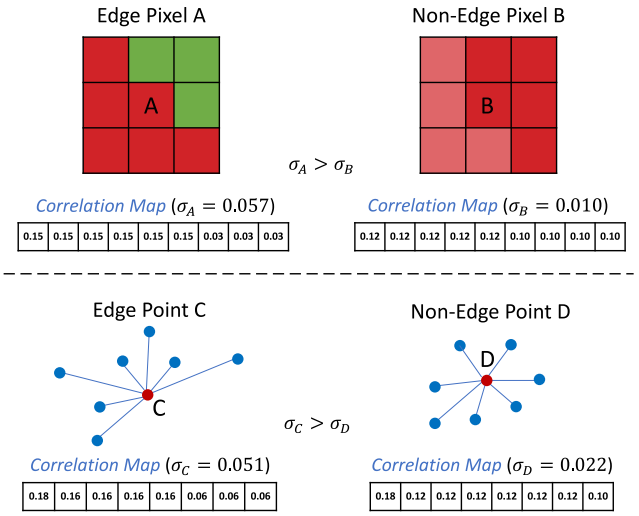

Figure 2. Illustration of using standard deviation to select edge pixels/points. A normalized correlation map is computed between the center pixel/point and its neighbors. The center pixel/point is self-contained as a neighbor. A larger standard deviation in the normalized correlation map means a higher possibility that it is an edge pixel/point.

The key to the effectiveness of the Canny edge detector is how edge pixels are defined. The intensity gradient of each pixel i is computed in comparison to its neighbors in a patch set $\mathcal{S}_i$, which is typically a 3×3 or 5×5 patch. Pixels with larger intensity gradients are defined as edge pixels. We make the following observation: *If there are large differences between the pixels from a patch set $\mathcal{S}_i$, then the standard deviation σ_i of the intensities in the patch is also high.* Hence, an alternative method for edge detection is to select pixels whose patch sets have larger σ_i.

We further generalize beyond pixel intensities to any (latent) per-pixel features $\mathbf{p}_i$ with a "measure of feature correlation" $h(\mathbf{p}_i, \mathbf{p}_{ij})$ defined between the center pixel i and its neighbor pixel j. In each patch $\mathcal{S}_i$, we call the vector $\mathbf{m}_i = \mathrm{softmax}\big(h(\mathbf{p}_i, \mathbf{p}_{ij})_{j \in \mathcal{S}_i}\big)$ the normalized correlation map between the center pixel and its neighbors. Then the standard deviation σ_i is computed over the elements of $\mathbf{m}_i$, and *pixels with larger σ_i are selected as edge pixels.* An illustration is given in the top row of Figure 2. When the neighbor number k is fixed (e.g., $k = 9$ for the top row, the center pixel is self-contained as a neighbor), for each patch, the mean value of its normalized correlation map is always $1/k$. However, for edge pixels, the standard deviations of their normalized correlation maps are larger.

For images, the proposed alternative edge detection algorithm, and in particular using the standard deviation for the normalized correlation map, is computationally much more expensive compared to the Canny edge detector. However, it provides the starting point to transfer the idea to point cloud edge sampling. Unlike images where pixels are well-aligned and patch operators can be easily defined and applied, point clouds are usually irregular, unordered, and potentially sparse. Voxel-based 3D convolution kernels are not applicable. Moreover, image pixels come with a color value (e.g. RGB or grayscale). For many point clouds, however, the point coordinates are the only available feature.

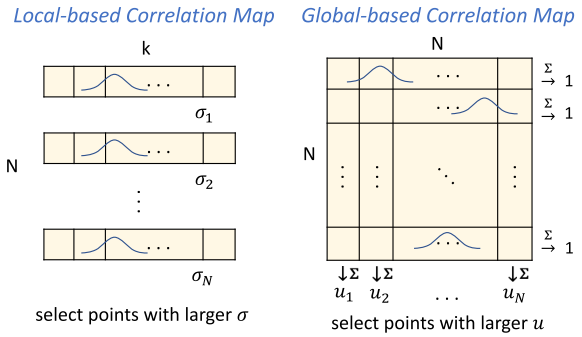

Figure 3. The key idea of proposed methods. N denotes the total number of points, while k denotes the number of neighbors used for local-based sampling method.

3.2. Local-based Point Cloud Edge Sampling

To adopt the previously introduced alternative edge detection algorithm to a point cloud set with $|\mathcal{S}| = N$ points, we use k-nearest neighbor to define a local patch $\mathcal{S}_i \subseteq \mathcal{S}$ for each point i to compute normalized correlation maps. As illustrated in the bottom row of Figure 2, when the neighbor number k is fixed (e.g., $k = 8$ for the bottom row, the center point is self-contained as a neighbor), for each patch, the mean value of its normalized correlation map is again always $1/k$. However, for edge points, the standard deviations of their normalized correlation maps are larger.

On the other hand, the attention mechanism is an ideal option to serve as the "measure of correlation" between point features within each patch, i.e., *the attention map serves as the normalized correlation map* directly. The local-based correlation measure $h^l(\cdot)$ is defined as

$$h^l(\mathbf{p}_i, \mathbf{p}_{ij}) = Q(\mathbf{p}_i)^\top K(\mathbf{p}_{ij} - \mathbf{p}_i) \tag{1}$$

where Q and K stand for the linear layers applied on the query input and the key input, respectively. Here we use the (latent) features of the center point $\mathbf{p}_i$ as the query input, and the feature difference between the neighbor point and the center point $\mathbf{p}_{ij} - \mathbf{p}_i$ as the key input. As in the original Transformer model [42], the square root of the feature dimension count $\sqrt{d}$ serves as a scaling factor. The final normalized correlation map $\mathbf{m}_i^l$ is given as

$$\mathbf{m}_i^l = \mathrm{softmax}\left(h^l(\mathbf{p}_i, \mathbf{p}_{ij})_{j \in \mathcal{S}_i} / \sqrt{d}\right) . \tag{2}$$

Again, a standard deviation σ_i is computed for each normalized correlation map. *The edge points are sampled by selecting the points with higher σ_i.*

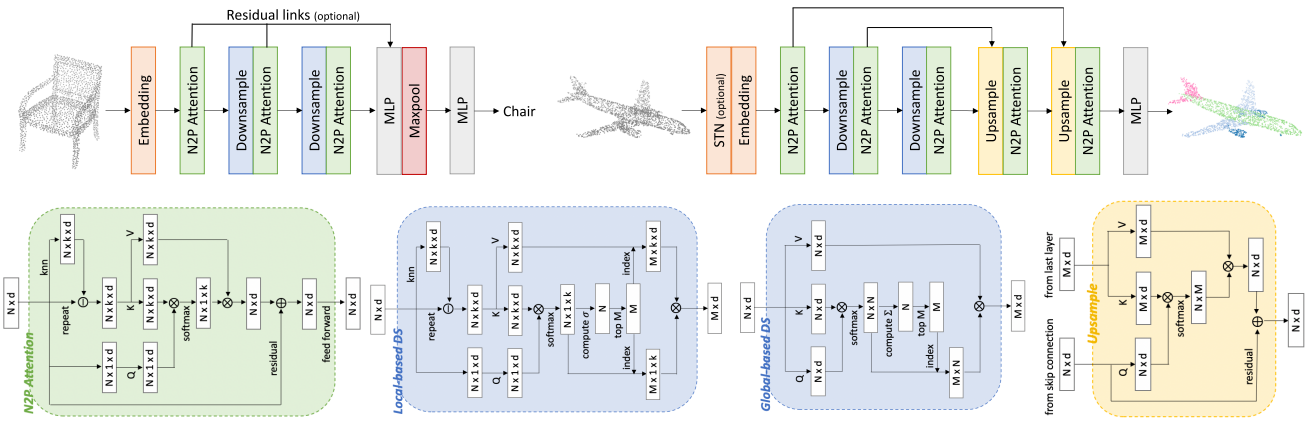

Figure 4. Network architectures for classification (top left) and segmentation (top right). The structures of N2P attention feature learning layer (bottom left), two alternative downsample layers (bottom middle), and upsample layer (bottom right) are also given. Both kinds of downsample layers downsample a point cloud from N points to M points, while upsample layer upsamples it from M points to N points.

3.3. Global-based Point Cloud Edge Sampling

We term the above-applied attention as neighbor-to-point (N2P) attention, which is specifically designed to capture local information using patches. For sampling problems, global information is also crucial. Considering the special case where all points are included in the local patch (i.e., $k = N$), a new global correlation map M^g of size $N \times N$ is obtained with the linear layers Q and K being shared for all points. Now the N2P attention simplifies into the common self-attention. We term it point-to-point (P2P) attention in this paper. In this case, the correlation measure $h^g(\cdot)$ and the normalized correlation map are defined as:

$$h^g(\mathbf{p}_i, \mathbf{p}_j) = Q(\mathbf{p}_i)^\top K(\mathbf{p}_j) \tag{3}$$

$$\mathbf{m}_i^g = \text{softmax}\left(h^g(\mathbf{p}_i, \mathbf{p}_j)_{j \in \mathcal{S}} / \sqrt{d}\right) \tag{4}$$

Note that all $\mathbf{m}_i^g$ now have the same point order, but the attention result for each point pair is not affected by the order.

The global correlation map M^g regroups the point-wise normalized correlation maps into a $N \times N$ matrix:

$$M^g = \begin{pmatrix} — & \mathbf{m}_1^{g\top} & — \\ — & \mathbf{m}_2^{g\top} & — \\ & \vdots & \\ — & \mathbf{m}_N^{g\top} & — \end{pmatrix} \tag{5}$$

In the context of the global correlation map M^g, instead of computing row-wise standard deviations for selecting points, we propose an alternative approach. Denote m_{ij} as the value of ith row and jth column in M^g. For point i, if it is an edge point, $\mathbf{m}_i^g$ has a larger standard deviation. In this case, considering its neighboring area, if a point j is close (based on 3d spatial space or latent space) to point i, m_{ij} is larger and point j is also likely to be an edge point. Given this property, now consider M^g column-wise. For a point j, in order to qualify it as an edge point, it needs to rank a higher value of m_{ij} in M^g more often compared to other points. Hence *instead of computing row-wise standard deviations, we compute column-wise sums*. Denote $u_j = \sum_i m_{ij}$, we then *sample the points with higher* u_j. Overall, we can consider it as follows: if a point contributes more in the self-attention correlation map, it is more likely to be an "important" point. An illustrative figure of the two proposed methods is given as Figure 3.

4. Experimental Results

4.1. Classification

Dataset. ModelNet40 [48] contains 12311 manufactured 3D CAD models in 40 common object categories. For a fair comparison, we use the official train-test split, in which 9843 models are used for training and 2468 models for testing. From each model mesh surface, points are uniformly sampled and normalized to the unit sphere. Only 3D coordinates are used as point cloud input. For data augmentation, we randomly scale, rotate, and shift each object point cloud in the 3D space. During the test, no data augmentation or voting methods were used.

Network Architecture Design. The classification network architecture is given in Figure 4. The embedding layer converts the input 3D coordinates into a higher-dimensional feature with multiple EdgeConv blocks. For feature learning layers, it is possible to use the layers designed for a similar purpose in other papers. Alternatively, the aforementioned N2P attention or P2P attention can also be used as feature learning layers. We use $k = 32$ neighbor points as default in local-based APES downsample layers. For an

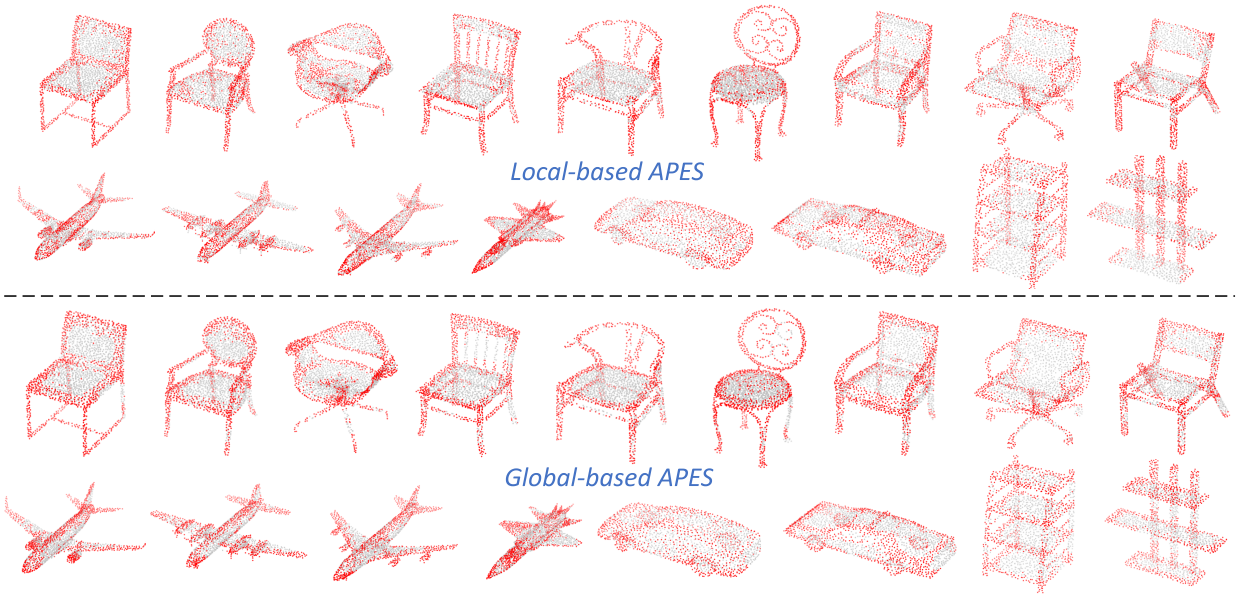

Figure 5. Visualized sampling results of local-based APES and global-based APES on different shapes. All shapes are from the test set.

input point cloud of N points from the previous layer, each downsample layer samples it to $N/2$ points. Note that our method can actually sample the point cloud to any desired number of points. The optional residual links are used for better feature transmission.

Training Details. To train the model, we use AdamW optimizer with an initial learning rate 1×10^{-4} and decay it to 1×10^{-8} with a cosine annealing schedule. The weight decay hyperparameter for network weights is set as 1. Dropout with a probability of 0.5 is used in the last two fully connected layers. We train the network with a batch size of 8 for 200 epochs.

Quantitative and Qualitative Results. The quantitative comparison with the SOTA methods is summarized in Table 1, where our proposed APES is among the best-performing methods. Qualitative results are presented in Figure 5. From it, we can observe that both local-based APES and global-based APES achieve good edge sampling results. On the other hand, local-based APES focuses more strictly on edge points, while global-based APES ignores some edge points and leverages a bit more on other non-edge points that are close to the edges. For example, in chair shapes, global-based APES discards some chair leg points and selects more points for chair seat edges to make the edges "thicker". We contribute its slightly better quantitative results to this. Overall, sampling more edge points improves the performance of downstream tasks. However, this can be overdone, and selecting only edge points can be detrimental. APES uses end-to-end training that includes the downstream task to make a good trade-off in the sample selection. Local-based APES imposes stronger mathematical statistics constraints during the task loss minimization, while global-based APES pursues better performance by allowing sampling the points that are less belong to the edge yet more important globally.

Method	Overall Accuracy
PointNet [33]	89.2%
PointNet++ [34]	91.9%
SpiderCNN [51]	92.4%
DGCNN [45]	92.9%
PointCNN [19]	92.2%
PointConv [47]	92.5%
PVCNN [25]	92.4%
KPConv [41]	92.9%
PointASNL [52]	93.2%
PT[1] [10]	92.8%
PT[2] [59]	93.7%
PCT [12]	93.2%
PRA-Net [6]	93.7%
PAConv [49]	93.6%
CurveNet [30]	**93.8%**
DeltaConv [46]	**93.8%**
APES (local-based)	93.5%
APES (global-based)	**93.8%**

Table 1. Classification results on ModelNet40. In comparison with other SOTA methods that also only use raw point clouds as input. Note that our reported results did not consider the voting strategy.

4.2. Part Segmentation

Dataset. The ShapeNetPart dataset [54] is annotated for 3D object part segmentation. It consists of 16,880 models from 16 shape categories, with 14,006 3D models for train-

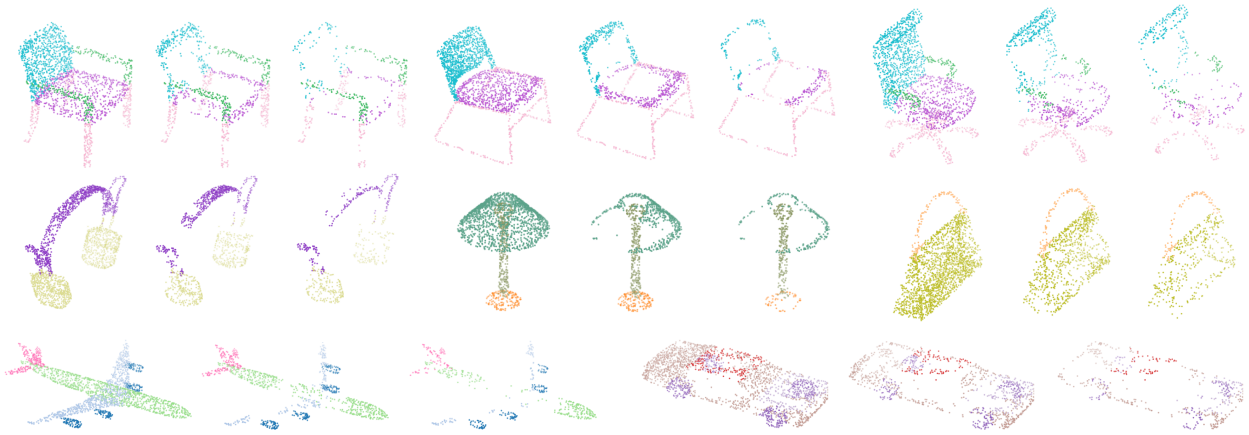

Figure 6. Visualized segmentation results as shape point clouds are downsampled. All shapes are from the test set.

ing and 2,874 for testing. The number of parts for each category is between 2 and 6, with 50 different parts in total. We use the sampled point sets produced in [34] for a fair comparison with prior work. For evaluation metrics, we report category mIoU and instance mIoU.

Network Architecture Design. The segmentation network architecture is given in Figure 4. Most network layers are identical to the layers in the classification model, except for the spatial transform network (STN) and the upsample layer. The optional STN layer learns a spatial transformation matrix to transform the input cloud for better initial alignment [33, 45]. The upsample layer is a cross-attention-based layer to map the input point cloud to an upsampled size. Its key and value input is the feature from the last layer, while the query input is the corresponding residual feature within the downsample process.

Training Details. To train the model, we use AdamW optimizer with an initial learning rate 1×10^{-4} and decay it to 1×10^{-8} with a cosine annealing schedule. The weight decay hyperparameter for network weights is set as 1×10^{-4}. The dropout with a probability of 0.5 is used in the last two fully connected layers. We train the network with a batch size of 16 for 200 epochs.

Quantitative and Qualitative Results. The segmentation quantitative results are given in Table 2. Our method achieves decent performance but is not on par with the best ones. However, as we compute the same metrics on the intermediate downsampled point clouds in Table 3, we surprisingly find that their performances are extremely good, even far better than the SOTA methods. This indicates the downsampled edge points contribute more to the performance, while the features of the discarded non-edge points are not well reconstructed by the upsample layer. Most other neural network papers use FPS for downsampling and FPS preserves a similar data distribution compared to the original point cloud. When upsampling, simple interpolation operations [15, 34, 59] are used to create new points. However, our method focuses on edge points and the sampled result has a quite different data distribution than the original point cloud. For non-edge points, especially those far from edges, neighbor-based interpolation methods are no longer applicable. We have designed a cross attention-based layer for upsampling, but it is still hard to get the features of the former discarded points back, even with residual links. Note that in this case, the upsampling problem

Method	Cat. mIoU	Ins. mIoU
PointNet [33]	80.4%	83.7%
PointNet++ [34]	81.9%	85.1%
SpiderCNN [51]	82.4%	85.3%
DGCNN [45]	82.3%	85.2%
SPLATNet [39]	83.7%	85.4%
PointCNN [19]	84.6%	86.1%
PointConv [47]	82.8%	85.7%
KPConv [41]	85.0%	86.2%
PT[1] [10]	-	85.9%
PT[2] [59]	83.7%	**86.6%**
PCT [12]	-	86.4%
PRA-Net [6]	83.7%	86.3%
PAConv [49]	84.6%	86.1%
CurveNet [30]	-	**86.6%**
StratifiedTransformer [15]	**85.1%**	**86.6%**
APES (local-based)	83.1%	85.6%
APES (global-based)	83.7%	85.8%

Table 2. Segmentation results on ShapeNet Part.

Method \ Points	Cat. mIoU (%)			Ins. mIoU (%)		
	2048	1024	512	2048	1024	512
APES (local)	83.11	85.56	**86.17**	85.58	87.27	87.41
APES (global)	83.67	84.86	85.44	85.81	87.78	**88.06**

Table 3. Segmentation results of the full point clouds and intermediate downsampled point clouds of different sizes.

Method	Feature Learning Layer	OA (%)
DGCNN	EdgeConv	92.90
APES (local-based)	EdgeConv	93.02
	P2P Attention	93.30
	N2P Attention	**93.47**
APES (global-based)	EdgeConv	93.18
	P2P Attention	93.46
	N2P Attention	**93.81**

Table 4. Ablation study of using different feature learning layers in the classification network.

Method	Embedding Dimension	OA (%)
APES (local-based)	64	93.10
	128	93.47
	192	**93.54**
APES (global-based)	64	93.34
	128	93.81
	192	**93.83**

Table 5. Ablation study of using a different number of embedding dimensions for the classification task.

actually turns into a point cloud completion or reconstruction task, which is another advanced topic for point cloud analysis. We would like to leave this for future work. The qualitative segmentation results are given in Figure 6, intermediate visualization results are also provided.

4.3. Ablation study

In this subsection, multiple ablation studies are conducted regarding the design choices of neural network architectures. All following experiments are performed on the classification benchmark of ModelNet40.

Feature Learning Layer. The feature learning layer we used in the above experiments is the N2P attention layer. However, as discussed in Section 4.1, it is possible to replace it with other feature layers. We additionally report the results of using EdgeConv or P2P attention as the feature learning layer in Table 4. From it, we can observe that N2P attention achieves the best performance. Meanwhile, the results of using EdgeConv are improved when using our proposed sampling methods.

Embedding Dimension. In most network-based methods, it is often reported that better performances are achieved when a larger embedding dimension is used. In our experiments, we use an embedding dimension of 128 as the default. We additionally report the results of using embedding dimensions of 64 and 192 in Table 5.

Choice of k in local-based APES. When local-based APES is used, the parameter of neighbor number k is a very important parameter since it decides the perception area size of local patches. We additionally report the results of using different k in Table 6.

k	8	16	32	64	128	256	512
OA (%)	93.14	93.26	93.47	93.52	93.54	93.59	**93.63**

Table 6. Ablation study of using a different number of neighbors for local-based edge point sampling.

Edge Supervision	None	Pre-trained and Fixed	Joint Training
APES (local-based)	93.47%	93.45%	93.46%
APES (global-based)	**93.81%**	93.47%	93.51%

Table 7. Ablation study of considering the edge supervision. Results of using it for pre-training or joint training are both presented.

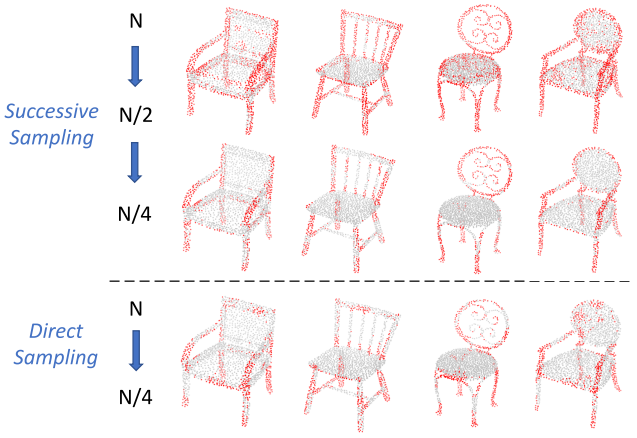

Figure 7. Sampling results of successively sampling to a fourth of the original size and directly sampling by a factor of four.

Successive sampling vs. Direct sampling. An advantage of our proposed method is that we can sample any desired number of points with it. We further provide qualitative comparison results of successively sampling the raw point cloud to a quarter ($N \rightarrow N/2 \rightarrow N/4$) and directly sampling it to a quarter ($N \rightarrow N/4$) in Figure 7. We observe that the sampled results are mostly similar. With the exception of a few extreme edge points which are better captured by successive sampling.

Additional edge point supervision. Since it is possible to compute "ground-truth" edge points from the shapes using local curvatures, we further study the cases where an edge supervision loss term is introduced. Experiments of not using the edge supervision, using it for pre-training (and fixing it during the downstream task training), and using it for joint training are conducted. Numerical results are given in Table 7. The results are consistent with our conclusion in subsection 4.1. For local-based APES which already focuses on edge point sampling, edge supervision has no significant impact. However, for global-based APES, edge supervision decreases performance slightly.

M	Voxel	RS	FPS [9]	S-NET [8]	PST-NET [43]	SampleNet [16]	MOPS-Net [36]	DA-Net [21]	LighTN [44]	APES (local)	APES (global)
512	73.82	87.52	88.34	87.80	87.94	88.16	86.67	89.01	89.91	90.79	**90.81**
256	73.50	77.09	83.64	82.38	83.15	84.27	86.63	86.24	88.21	90.38	**90.40**
128	68.15	56.44	70.34	77.53	80.11	80.75	86.06	85.67	86.26	89.73	**89.77**
64	58.31	31.69	46.42	70.45	76.06	79.86	85.25	85.55	86.51	88.68	**89.57**
32	20.02	16.35	26.58	60.70	63.92	77.31	84.28	85.11	86.18	86.49	**88.56**

Table 8. Comparison with other sampling methods. Evaluated on the ModelNet40 classification benchmark with multiple sampling sizes.

Method	S-NET	PST-NET	SampleNet	MOPS-Net	LighTN	APES (local)	APES (global)
Params	0.33M	0.42M	0.46M	0.44M	0.37M	0.35M	0.35M
FLOPs	152M	122M	167M	149M	115M	142M	114M

Table 9. Computation complexity of different sampling methods. Here "M" stands for million.

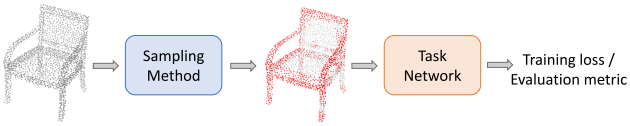

Figure 8. Framework for sampling methods evaluation.

5. Sampling Methods Comparison

5.1. Experiment Setting

We additionally compare our sampling method to previous work including RS, FPS, and the more recent learning-based S-Net, SampleNet, LighTN, etc. The same evaluation framework from [8, 16, 44] is used, as illustrated in Figure 8. The task here is the ModelNet40 Classification, and the task network is PointNet. Sampling methods are evaluated with multiple sampling sizes.

As discussed in the results part of Section 4.2, edge point sampling changes the data distribution compared to the original point cloud, especially when a large downsampling ratio (defined as N/M) is used. Hence for a fair comparison, in order to achieve a downsampled point cloud size of M, we first sample the input point cloud to a size of $2M$ with FPS, then sample it to the desired size M with our method APES. The computation complexity of different sampling models is given in Table 9. For a fair comparison, we use the same sampling size $M = 512$ and the same point embedding dimension of 128 in this table.

5.2. Quantitative and Qualitative Results

Quantitative results are given in Table 8, from which we can observe that both local-based and global-based APES achieve good classification results with the task network under different sampling ratios. Additional qualitative results are provided in Figure 9. Although other learning-based methods achieve decent numerical results, it is difficult to identify their sampling patterns from the visualization results. Their results look quite similar to random sampling. On the other hand, our proposed method shows a comprehensive sampling pattern of sampling point cloud outlines.

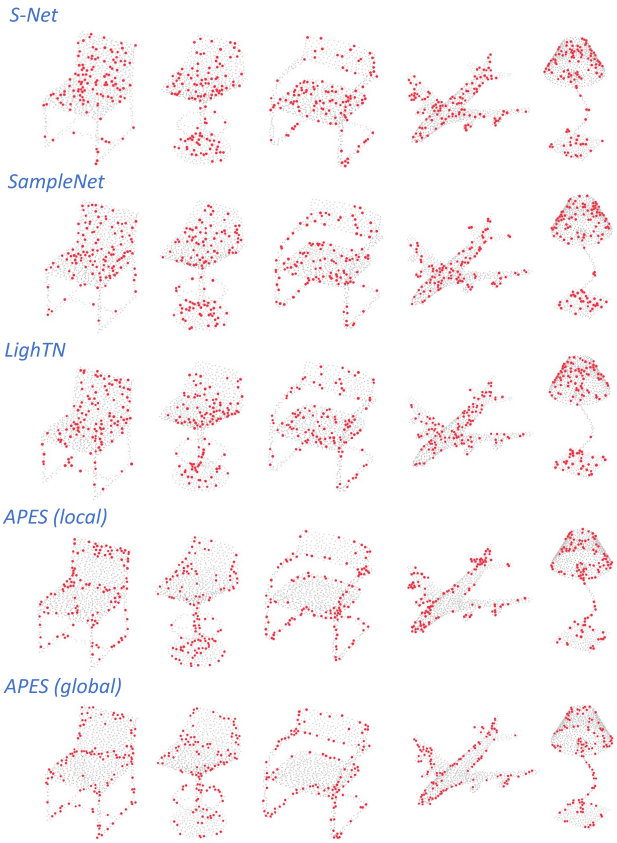

Figure 9. Qualitative comparison for the sampling of 128 points from input point clouds with 1024 points.

6. Conclusion

In this paper, an attention-based point cloud edge sampling (APES) method is proposed. It uses the attention mechanism to compute correlation maps and sample edge points accordingly. Two variations of local-based APES and global-based APES are proposed based on different attention modes. Qualitative and quantitative results show that our method achieves excellent performance on common point cloud benchmark tasks.

For future work, it is possible to design other supplementary losses for the training. Moreover, we noticed that the different point distribution by edge point sampling hinders later upsampling operations and segmentation performance. It would be interesting to design upsampling methods that can better cope with edge point sampling.

References

[1] Nicolas Audebert, Bertrand Le Saux, and Sébastien Lefèvre. Semantic segmentation of earth observation data using multimodal and multi-scale deep networks. In *Asian conference on computer vision*, pages 180–196. Springer, 2016. 2

[2] Prarthana Bhattacharyya, Chengjie Huang, and K. Czarnecki. Sa-det3d: Self-attention based context-aware 3d object detection. *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 3022–3031, 2021. 2

[3] Alexandre Boulch, Bertrand Le Saux, and Nicolas Audebert. Unstructured point cloud semantic labeling using deep segmentation networks. *3DOR@ Eurographics*, 3, 2017. 2

[4] John F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8:679–698, 1986. 2

[5] Can Chen, Luca Zanotti Fragonara, and Antonios Tsourdos. Gapnet: Graph attention based point neural network for exploiting local feature of point cloud. *Neurocomputing*, 438:122–132, 2021. 2

[6] Silin Cheng, Xiwu Chen, Xinwei He, Zhe Liu, and Xiang Bai. Pra-net: Point relation-aware network for 3d point cloud analysis. *IEEE Transactions on Image Processing*, 30:4436–4448, 2021. 5, 6

[7] Zhang Cheng, Haocheng Wan, Xinyi Shen, and Zizhao Wu. Patchformer: A versatile 3d transformer based on patch attention. *ArXiv*, abs/2111.00207, 2021. 2

[8] Oren Dovrat, Itai Lang, and Shai Avidan. Learning to sample. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2755–2764, 2019. 1, 2, 8

[9] Yuval Eldar, Michael Lindenbaum, Moshe Porat, and Yehoshua Y. Zeevi. The farthest point strategy for progressive image sampling. *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 2 - Conference B: Computer Vision & Image Processing. (Cat. No.94CH3440-5)*, pages 93–97 vol.3, 1994. 1, 2, 8

[10] Nico Engel, Vasileios Belagiannis, and Klaus C. J. Dietmayer. Point transformer. *IEEE Access*, 9:134826–134840, 2021. 2, 5, 6

[11] Fabian Groh, Patrick Wieschollek, and Hendrik P. A. Lensch. Flex-convolution - million-scale point-cloud learning beyond grid-worlds. In *ACCV*, 2018. 1, 2

[12] Meng-Hao Guo, Junxiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph Robert Martin, and Shimin Hu. Pct: Point cloud transformer. *Comput. Vis. Media*, 7:187–199, 2021. 2, 5, 6

[13] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Agathoniki Trigoni, and A. Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11105–11114, 2020. 2

[14] Mingyang Jiang, Yiran Wu, Tianqi Zhao, Zelin Zhao, and Cewu Lu. Pointsift: A sift-like network module for 3d point cloud semantic segmentation. *arXiv preprint arXiv:1807.00652*, 2018. 2

[15] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified transformer for 3d point cloud segmentation. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8490–8499, 2022. 2, 6

[16] Itai Lang, Asaf Manor, and Shai Avidan. Samplenet: Differentiable point cloud sampling. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7575–7585, 2020. 1, 2, 8

[17] Felix Järemo Lawin, Martin Danelljan, Patrik Tosteberg, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Deep projective 3d semantic segmentation. In *International Conference on Computer Analysis of Images and Patterns*, pages 95–107. Springer, 2017. 2

[18] Truc Le and Ye Duan. Pointgrid: A deep network for 3d shape understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9204–9214, 2018. 2

[19] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *NeurIPS*, 2018. 1, 2, 5, 6

[20] Zhidong Liang, Ming Yang, Hao Li, and Chunxiang Wang. 3d instance embedding learning with a structure-aware loss function for point cloud segmentation. *IEEE Robotics and Automation Letters*, 5:4915–4922, 2020. 2

[21] Yanan Lin, Yan Huang, Shihao Zhou, Mengxi Jiang, Tianlong Wang, and Yunqi Lei. Da-net: Density-adaptive downsampling network for point cloud classification via end-to-end learning. *2021 4th International Conference on Pattern Recognition and Artificial Intelligence (PRAI)*, pages 13–18, 2021. 1, 2, 8

[22] Yiqun Lin, Zizheng Yan, Haibin Huang, Dong Du, Ligang Liu, Shuguang Cui, and Xiaoguang Han. Fpconv: Learning local flattening for point convolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4293–4302, 2020. 2

[23] Zhi-Hao Lin, Sheng-Yu Huang, and Yu-Chiang Frank Wang. Convolution in the cloud: Learning deformable kernels in 3d graph convolution networks for point cloud analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1800–1809, 2020. 2

[24] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8887–8896, 2019. 2

[25] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. *ArXiv*, abs/1907.03739, 2019. 5

[26] Dening Lu, Kyle Gao, Qian Xie, Linlin Xu, and Jonathan Li. 3dpct: 3d point cloud transformer with dual self-attention. *ArXiv*, abs/2209.11255, 2022. 2

[27] Dening Lu, Qian Xie, Linlin Xu, and Jonathan Li. 3dctn: 3d convolution-transformer network for point cloud classification. *ArXiv*, abs/2203.00828, 2022. 2

[28] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition.

In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015. 2

[29] Carsten Moenning and Neil A. Dodgson. Fast marching farthest point sampling. In *Eurographics*, 2003. 1, 2

[30] A. A. M. Muzahid, Wanggen Wan, Ferdous Sohel, Lianyao Wu, and Li Hou. Curvenet: Curvature-based multitask learning deep networks for 3d object recognition. *IEEE/CAA Journal of Automatica Sinica*, 8:1177–1187, 2021. 5, 6

[31] Ehsan Nezhadarya, Ehsan Moeen Taghavi, Bingbing Liu, and Jun Luo. Adaptive hierarchical down-sampling for point cloud classification. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12953–12961, 2020. 2

[32] Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3d object detection with pointformer. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7459–7468, 2021. 2

[33] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017. 2, 5, 6

[34] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017. 1, 2, 5, 6

[35] Haozhe Qi, Chen Feng, ZHIGUO CAO, Feng Zhao, and Yang Xiao. P2b: Point-to-box network for 3d object tracking in point clouds. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6328–6337, 2020. 2

[36] Yu Qian, Junhui Hou, Yiming Zeng, Qijian Zhang, Sam Tak Wu Kwong, and Ying He. Mops-net: A matrix optimization-driven network fortask-oriented 3d point cloud downsampling. *ArXiv*, abs/2005.00383, 2020. 1, 2, 8

[37] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *ArXiv*, abs/1505.04597, 2015. 2

[38] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3693–3702, 2017. 2

[39] Hang Su, V. Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2530–2539, 2018. 6

[40] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3887–3896, 2018. 2

[41] Hugues Thomas, C. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6410–6419, 2019. 2, 5, 6

[42] 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. 2, 3

[43] Xu Wang, Yi Jin, Yigang Cen, Congyan Lang, and Yidong Li. Pst-net: Point cloud sampling via point-based transformer. In *ICIG*, 2021. 2, 8

[44] Xu Wang, Yi Jin, Yigang Cen, Tao Wang, Bowen Tang, and Yidong Li. Lightn: Light-weight transformer network for performance-overhead tradeoff in point cloud downsampling. *ArXiv*, abs/2202.06263, 2022. 2, 8

[45] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38:1 – 12, 2019. 2, 5, 6

[46] Ruben Wiersma, Ahmad Nasikun, Elmar Eisemann, and Klaus Hildebrandt. Deltaconv: Anisotropic point cloud learning with exterior calculus. *ArXiv*, abs/2111.08799, 2021. 2, 5

[47] Wenxuan Wu, Zhongang Qi, and Fuxin Li. Pointconv: Deep convolutional networks on 3d point clouds. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9613–9622, 2019. 1, 2, 5, 6

[48] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015. 4

[49] Mutian Xu, Runyu Ding, Hengshuang Zhao, and Xiaojuan Qi. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3172–3181, 2021. 2, 5, 6

[50] Qiangeng Xu, Xudong Sun, Cho-Ying Wu, Panqu Wang, and Ulrich Neumann. Grid-gcn for fast and scalable point cloud learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5661–5670, 2020. 2

[51] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. *ArXiv*, abs/1803.11527, 2018. 5, 6

[52] Xu Yan, Chaoda Zheng, Zhuguo Li, Sheng Wang, and Shuguang Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5588–5597, 2020. 1, 5

[53] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds with self-attention and gumbel subset sampling. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3318–3327, 2019. 1

[54] L. Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas J. Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (TOG)*, 35:1 – 12, 2016. 5

[55] Jason Yosinski, Jeff Clune, Anh M Nguyen, Thomas J. Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *ArXiv*, abs/1506.06579, 2015. 1

[56] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-net: Point cloud upsampling network. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2790–2799, 2018. 1

[57] Kuangen Zhang, Ming Hao, Jing Wang, Xinxing Chen, Yuquan Leng, Clarence W. de Silva, and Chenglong Fu. Linked dynamic graph cnn: Learning through point cloud by linking hierarchical features. *2021 27th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, pages 7–12, 2021. 2

[58] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5565–5573, 2019. 2

[59] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H. S. Torr, and Vladlen Koltun. Point transformer. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16239–16248, 2021. 2, 5, 6

[60] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018. 2