SWPT: Spherical Window-based Point Cloud Transformer

Xindong Guo[1,2], Yu Sun[2], Rong Zhao[1], Liqun Kuang[1], and Xie Han[1]✉

[1] North University of China, Taiyuan, China
gxd@sxau.edu.cn, hanxie@nuc.edu.cn
[2] Shanxi Agricultural University, Jinzhong, China

Abstract. While the Transformer architecture has become the de-facto standard for natural language processing tasks and has shown promising prospects in image analysis domains, applying it to the 3D point cloud directly is still a challenge due to the irregularity and lack of order. Most current approaches adopt the farthest point searching as a downsampling method and construct local areas with the k-nearest neighbor strategy to extract features hierarchically. However, this scheme inevitably consumes lots of time and memory, which impedes its application to near-real-time systems and large-scale point cloud. This research designs a novel transformer-based network called Spherical Window-based Point Transformer (SWPT) for point cloud learning, which consists of a Spherical Projection module, a Spherical Window Transformer module and a crossing self-attention module. Specifically, we project the points on a spherical surface, then a window-based local self-attention is adopted to calculate the relationship between the points within a window. To obtain connections between different windows, the crossing self-attention is introduced, which rotates all the windows as a whole along the spherical surface and then aggregates the crossing features. It is inherently permutation invariant because of using simple and symmetric functions, making it suitable for point cloud processing. Extensive experiments demonstrate that SWPT can achieve the state-of-the-art performance with about 3-8 times faster than previous transformer-based methods on shape classification tasks, and achieve competitive results on part segmentation and the more difficult real-world classification tasks.

Keywords: Point cloud · Spherical projection · Transformer.

1 Introduction

3D point clouds have been attracting more and more attention from both industry and academia due to their broad applications including autonomous driving, augmented reality, robotics, etc. Unlike images having regular pixel grids, 3D point clouds are irregular and unordered sets of points corresponding to object surfaces. This difference makes it challenging to apply traditional network architectures used widely in 2D computer vision directly into 3D point cloud.

Researchers propose a variety of models for deep learning on 3D point clouds which are classified into three categories according to data representations, i.e., projection-based, voxel-based and point-based models. The projection-based models [25, 3, 13, 36] generally project 3D shapes into regular 2D representations, so that many classical convolutional models can be employed. However, these methods can only obtain limited receptive field from one or a few perspectives, which may induce losses of spatial relations between different parts. The voxel-based models[18, 33, 23] generally rasterize 3D point clouds onto regular grids and apply 3D convolutions for feature learning, which induces massive computational and memory costs due to the cubic growth in number of voxels related to the resolution. Although sparse convolutional models[23, 29, 2] alleviate this problem by performing only on volexs that are not empty, it may still not capture the spatial relations. The point-based models[20, 22] operate directly on points and propagate features via pooling operators. Specifically, some works transform point cloud to a graph for message passing[24, 14]. These models achieve more competitive results over previous methods. However, the neighborhood searching strategy which is a core component of these methods has a high computational complexity as the iterative number increasing or points scale getting larger.

In recent years, Transformer[1, 28] has been dominating the natural language processing field, and has been applied to image vision tasks, also achieving encouraging performance[31, 6]. As the core component of Transformer, the self-attention module computes the refined weighted features based on global context by considering the connections between any two words. So the output feature of each word is related to all input features, which make it capable of obtaining the global feature. Specifically, all operations of Transformer are parallelizable and order invariance, so it is naturally suitable for point cloud processing. Transformer-based model [8] is the pioneering work that introduces the self-attention in points processing, which employ the farthest point searching (FPS) as downsampling strategy and k-nearest neighbor (KNN) as local region searching strategy to perform local self-attention operation. It lacks spatial connections between different regions, and consumes a large amount of time when constructing local regoins due to the high complexity of FPS and KNN.

To solve the problems mentioned above, we propose a noval Spherical Window based Point Cloud Transformer. Firstly, we project point cloud to a spherical surface to reduce the dimension of points, which makes the processing of points as simple as images. The Spherical Window (SW) module makes the points projected on the spherical surface more regular ressembling pixels in an image, but with more spatial structue information. Then, we partition the points on the spherical surface into spherical windows and apply local self-attention hierarchically on each window. With this spherical window-based mechanism, local self-attention are performed in all windows parallelly, which would facilitate point cloud processing and is beneficial to its scalability for large point clouds. Finally, we introduce a Cross-Window operation to achieve connections between the neighboring windows, which is an efficient operation with computational complexity as low as $O(1)$. We use the SWPT as the backbone to a variety of

point cloud recognition tasks, and extensive experiments demonstrate that it's an effective and efficient framework.

The main contributions of this paper are summarized as following:

- We propose a Spherical Projection (SP) Layerwhich projects the points to a spherical surface, followed by a point-wise transformation to maintain the information between points. This layer is permutition-invariant, high-efficiency and thus is inherently suitable for point cloud processing.
- Based on the Spherical Projection Layer, we construct a Spherical Windows (SW) Transformer which partitions the spherical surface into windows comprising certain patches. Then we perform local self-attention on each window hierarchically.
- We introduce a cross window operation, which rotates the spherical surface as a whole by half of the window size to exploit the spatial connections between different windows. Extension experiments demonstrate that our network achieves the state-of-the-art performance on shape classification, part segmentation and semantic segmentation.

2 Related Works

2.1 Projection-based networks

As a pioneering work, MVCNN[25] simply use max-pooling to aggregate multi-view features into a global descriptor. But max-pooling only retains the top elements from a specific view, which result in loss of details. These methods[3, 13] project 3D point clouds into various image planes, and then employ 2D CNNs to extract local features in these image planes followed by a multi-view features fusion module to obtain the global feature. This approach[36] propose a relation network to exploit the inter-relationships over a group of views which are aggregated to form a discriminative representation indicating the 3D object. Different from the previous methods, [30] construct a directed graph by regarding multiple views as graph nodes, and then design a GCNN over the view-graph to hierarchically achieve global shape descriptor.

2.2 Voxel-based networks

VoxNet [18] integrates volumetric Occupancy Grid representation with a supervised 3D CNN to utilize 3D information and deal with large amounts of point cloud. 3D ShapeNets [33, 23] is another work using a Convolutional Deep Belief Network to represent a geometric 3D shape as a probability distribution of binary variables on a 3D voxel grid, which can learn the distribution of complex 3D shapes across different categories. However, these methods are unable to scale well to dense 3D data since the computation and memory consumption of such methods increases cubically with respect to the resolution of voxel.

2.3 Point-based networks

PointNet[20] as a pioneer work firstly utilizes permutation-invariant operators such as MLP and max-pooling to aggregate a global feature from a point cloud. As PointNet lacks local connections between points, the author propose PointNet++[22] which applies tiny PointNet within a hierarchical spatial structure to extract local features with increasing contextual scales. Inspired by PointNet, many a recent works[11, 32, 9] are propoesd with more sophisticated architecture achieving encouraging performance. Afterwards, a graph convolutional neural network(GCNN) is applied to extract features. DGCNN[24] is the first performing graph convolutions on KNN graphs. As the core component of EdgeConv in DGCNN, MLP is used to learn the features for each edge. Deepgcns[14] presents a new way to train a very deep GCNs to solve the vanishing gradient problem and shows a 56-layer GCN achieving positive effect. PointCNN[15] propose a X-transformation learnt from the point cloud, which weight the input features associated with the points and reorder them into latent potentially canonical order simultaneously. PointConv[32] extend the dynamic filter to a new convolution and take the local coordinates of 3D points as input to compute weight and density functions.

2.4 Transformer in NLP and Vision

[1] propose a neural mechine translation with an attention mechanism allowing a model to automatically search for parts of a source sentence that are relevant to predicting a target word. [16] further propose a self-attention mechanism to extract an interpretable sentence embedding. Subsequent works employed self-attention layers to replace some or all of the spatial convolution layers. [28] propose Transformer based solely on self-attention mechanism without recurrence and convolutions entirely. [5] propose a new Bidirectional Transformers to pre-train deep bidirectional representations and it obtains competitive results.

Transformer and self-attention models have revolutionized natural language processing and inspired researchers to apply them to vision tasks. [31] proposed visual transformers that apply Transformer to token-based images from feature maps. [6] propose a pure transformer-based network partitioning a image to patches, and results show with sufficient training data, Transformer provides better performance than a traditional convolutional neural network. [17] propose a new vision Transformer, called Swin-Transformer, to incorporate inductive bias for spatial locality, as well as for hierarchy and translation invariance.

3 Spherical-Window Point Transformer

3.1 Overview

In this section we detail the design of SWPT. We first show how the spherical projection (SP) can provide an efficient representation for point clouds. Then we present the detail of the gridding module which splits the points on the spherical

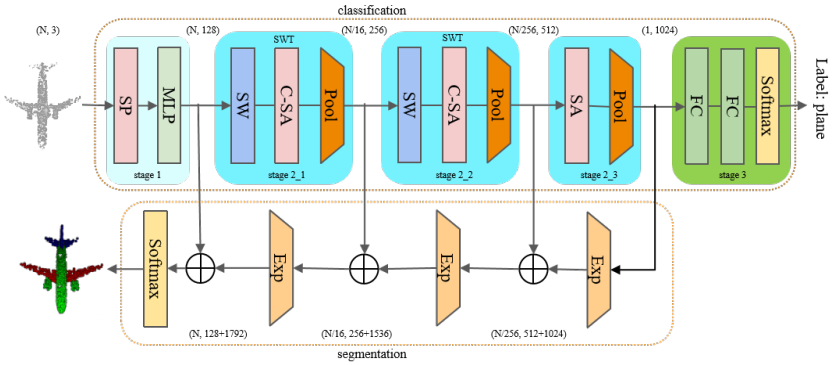

Fig. 1. Model Architecture. The above is the classification network, and the below is the segmentation network. SP stands for the spherical projection. SW is the spherical window layer used to split points. C-SA is the crossing self-attention applied to calculate the cross window attentions. Exp means expanding high-dimension features to local windows.

surface into non-overlapping patches. Lastly, we introduce a high-performance self-attention module to extract local features, and with a rotation to enhance the cross-window connections.

The model architecture is illustrated in Fig 1. The above of Fig 1 illustrates the classification network which takes as input the raw point cloud with N points. Firstly, the network applies a Spherical Projection (SP) to all points, followed by a point-wise transformation, e.g. multi-layers perceptron (MLP). Then the outputs of the previous layer are fed into two stacked Spherical Window Transformer (SWT) which consists of a Spherical Window (SW) module, a Crossing self-attention (C-SA) module and a pooling module to learn hierarchical features. It is worth noting that the network achieves competitive results with only two iterations of the SWT module thanks to the high efficiency of the Spherical Window based neighborhood searching strategy. Finally, a self-attention (SA) and a pooling operation are employed to produce a global feature which is fed into a fully connected (FC) layer followed by a softmax function to predict the label. The below of Fig 1 shows the segmentation network which expands the high-dimension features to the corresponding local windows in the previous layer and then concatenates them with features from the last layer. After three expanding and concatenation operations, the network produces the dense points features that a lightweight fully connected network use to generate labels.

3.2 Spherical Projection

Spherical Projection Layer plays a role of data pre-processing in the framework, which projects the points from the point cloud to a spherical surface, as illustrated in Fig 2. For projecting the points on the spherical surface evenly, we

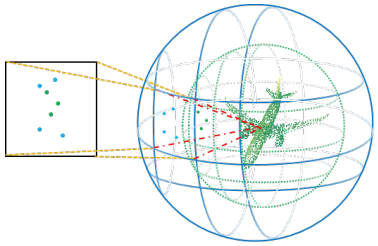

Fig. 2. Visualization of Spherical projection. For simplification, here we just show two spheres for projection. The points on different sections in the same frustum are gathered into a spherical window. The blue points and the green points in the window are from the blue spherical surface and green spherical surface respectively.

normalize the points by the following formulation:

$$p' = p - p_c \qquad p_c = \frac{1}{N}(\sum_{i=1}^{N} x_i,\ \sum_{i=1}^{N} y_i,\ \sum_{i=1}^{N} z_i) \tag{1}$$

where p_c is the centroid of point cloud and p' is the point translated from point $p = (x, y, z)$. N stands for the number of points. By doing so, we translate the centroid of point cloud to the origin of coordinates.

Then, we define a spherical projection method which can be formulated as follows:

$$\begin{aligned} r &= \sqrt{x^2 + y^2 + z^2} \\ \theta &= \arctan \frac{y}{x} \\ \varphi &= \arccos \frac{z}{r} \end{aligned} \tag{2}$$

Here (x, y, z) is the coordinate of point in the Cartesian system, θ and φ are the angle of azimuth and angle of pitch, respectively.

Although the spherical projection preserves most prominent features, the information of connection between points on the spherical surface may change compared to the original point cloud due to the reduction of dimension, causing structure information loss. To solve this problem, we apply a transformation to the points through a shared MLP, projecting the features of the points to a latent space, while maintaining the spatial relations between points:

$$f = T(p) \oplus P(p) \quad P(p) = r \oplus \theta \oplus \varphi\ , \tag{3}$$

where $p \in R^D$ is the feature of a point such as x-y-z coordinate, in our case D is 3. In other cases, D may be different when the point includes features like normal vector, etc. T is a pointwise transformation such as MLP. $\oplus$ is a vector concatenation operator. P indicates the spherical projection mentioned before.

Eventually, the output of the Spherical Projection module is a new set of features:

$$F = \{f_i \mid f_i = P(p_i) \oplus T(p_i),\ i \in N\}, \tag{4}$$

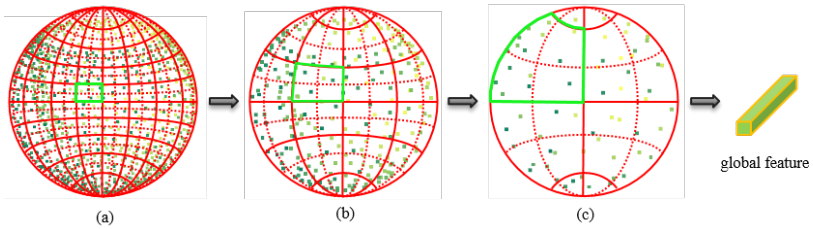

Fig. 3. The proposed Spherical Window Transformer obtains the global feature by merging features within a 2×2 spherical window hierarchically. The points features within a solid window are adopted to perform local self-attention and then aggregated to a point for the next stage denoted by the dotted box. The green box indicates a local spherical window to perform self-attention and pooling operation. Please note that here we simplify the figure structure to illustrate the principle of a module clearly.

among which the angle of azimuth θ and angle of pitch φ are utilized to partition features into windows.

3.3 Spherical Window Transformer

The simplest way to apply Transformer on point clouds is to treat the entire points as a sequence and each point as an element, but it incurs massive computation and memory costs and local spatial relation loss. In addition, it lacks hierarchies which are significant in learning features proved by [22]. On the other hand, the farthest point sampling (FPS) and K-nearest neighbors (KNN) searching strategies adopted by most previous models introduce large computation and memory costs, making it unsuitable for large scale point clouds [7, 8].

To this end, we propose the Spherical Window Transformer layer to compute local self-attention and to construct a hierarchical architecture, which has linear computational complexity to patch size. The Spherical Window Transformer layer aims to provide an efficient local area partitioner which splits the points on the spherical surface into lots of patches, a patch involve points from a neighborhood. To this end, we adopt a very simple and efficient approach formally defined as follow:

$$F' = R_{\varphi}(S_{\varphi}(R_{\theta}(S_{\theta}(F)))), \tag{5}$$

where $F \in R^{N \times C}$ is a set of point features from the last layer, S_{θ} is a sort function by θ, R_{θ} is capable of reshaping the set shape from $N \times C$ to $\Theta \times N_{\theta} \times C$ where $N = N_{\theta} \times \Theta$. S_{φ} and R_{φ} perform the similar thing. After then we obtain a new set with the shape $\Phi \times \Theta \times N_{\phi} \times N_{\theta} \times C$ meaning that the point cloud is partitioned into $\Phi \times \Theta$ spherical windows with $N_{\theta} \times N_{\phi}$ points in each window. Because of using sorting and reshaping only, this layer maintains the permutation-invariance of point cloud.

Then a local self-attention is employed in each window, followed by a patch merging block to gather the local features with low-dimension. Afterwards, local

points are gathered to a high-dimension space with fewer points. The windows are arranged to enenly partition the points into non-overlapping areas which produce a higher layer local neighborhood, as shown in Fig 3.

Given a window size of (θ, φ), which means it has a number of $\theta \times \varphi$ points, the computational complexity of global self-attention and window-based local self-attention on a spherical surface of $\Theta \times \Phi$ patches are:

$$\Omega(SA) = 4\Theta\Phi C^2 + 2(\Theta\Phi)^2 C, \tag{6}$$

$$\Omega(W - SA) = 4\Theta\Phi C^2 + 2(\Theta\Phi)(\theta\varphi)C, \tag{7}$$

where the former is quadratic to points number $\Theta \times \Phi$, and the latter is linear when the window size is fixed. The global self-attention computation cost is generally unaffordable for a large scale point cloud, while window-based self-attention is scalable. Another function of the Spherical Window module is to reduce the cardinality of patches as required, for example, from N points to N/W through a window of W size.

The strategy of choosing local neighborhoods of previous works is KNN with Euclidean distancewhich has a linear computational complexity with respect to the number of neighborhoods, total points and the dimension of data. So it's still a challenge to extend a model using this to large scale point clouds. Our method using the Spherical Window to split points only needs to make a spherical surface grid by sorting points along the angle of azimuth and pitch, which only need to do once after the projection finished and is invariant to permutation of points. Extensive experiments demonstrates that the Spherical Window is a novel representation of point cloud on which local self-attention can be applied effectively and efficiently.

3.4 Crossing Windows Self-Attention

Spherical window-based local self-attention performs well to obtain local features. However, it lacks the capacity of acquiring relationships across neighboring windows, which limits its modeling power. To capture cross window relationships while maintaining the efficiency of computation and memory cost, we propose a rotating spherical surface approach which rotates all the windows as a whole sphere along the angle of azimuth and the angle of pitch by half of window size respectively. As illustrated in Fig 4, the module on the left calculates a regular self-attention within each window. Then the module on the right rotates windows along the angle of azimuth and pitch by $\lfloor \theta/2, \varphi/2 \rfloor$, followed by a self-attention computation same as before, where the θ and φ are window size. As the windows form a spherical surface and data points are distributed discretely over it, the rotation of windows does not need a masked operation resembling that in [17]. Results show that our Crossing Self-Attention approach introduces relationships between neighboring windows in previous layers and performs effectively and efficiently in several point cloud recognition tasks, as shown in table 5.

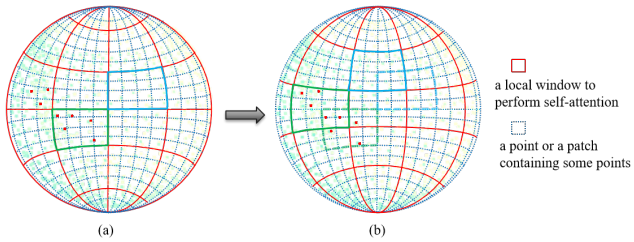

Fig. 4. Illustration of Crossing Windows Self-Attention considering the relationship between differenet windows. In this example, the window size is 4×4 and the rotating size is 2×2. After calculating the window-based self-attention showed by (a), the whole spherical window rotate along azimuth and pitch by 2×2 patches respectively, illustrated by (b). The green and blue dotted boxes in (b) indicate the spherical windows before rotation corresponding to windows in (a), and the solid boxes in (b) denote the spherical windows after rotation.

4 Experiments

We evaluate the performance of SWPT on three recognization tasks: 3D shape classification, 3D part segmentation and real-world object classification, giving comprehensive analyses and comparing with other approaches. For 3D shape classification, we use the widely adopted ModelNet40 [34] dataset. For 3D part segmentation, we use the ShapeNet55 [37] dataset. For real-world object classification, we use the ScanObjectNN [27] dataset which is a recent point cloud object dataset constructed from the real-world indoor datasets such as SceneNN [10] and ScanNet [4].

We use PyTorch [19] as the framework to implement SWPT. We employ the stochastic gradient descent (SGD) optimizer with momentum 0.9 and weight decay 0.0001 for training. Other training parameters including number of patches, window size, learning rate and batch size are given later in each related section.

4.1 Shape Classification

Data and metric. The ModelNet40 [34] dataset contains 12,311 CAD models in 40 categories, which is widely used in shape classification. For a fair comparison, we use the official split with 9,843 models for training and 2,468 models for testing, and use the same strategy as PointNet [20] to sample 1,024 points uniformly from each CAD model. For evaluation metrics, we adopt the mean accuracy within each category (mAcc) and the overall accuracy (OA) over all classes. In addition, we retrain the point-based methods to evaluate their efficiency.

Experiment results. During training we used a random translation in [0.2, 0.2], a random anisotropic scaling in [0.67, 1.5] and a random input dropout as the data augmentation strategy, while no data augmentation was used in testing. We set the batch size to 32, epochs to 200 and initial learning rate to 0.01 with a

Table 1. Comparison of state-of-the-art models on ModelNet40 dataset. The mAcc and OA results are quoted from the cited papers, while the latencies are obtained by training and testing the official code in the same environment.

Method	input	points	mAcc(%)	OA(%)	Latency(s)
3DShapeNets [33]	voxel	-	77.3	84.7	-
VoxNet [18]	voxel	-	83.0	85.9	-
Subvolumn [21]	voxel	-	86.0	89.2	-
MVCNN [25]	image	-	-	90.1	-
Kd-Net [12]	point	-	-	91.8	-
PointNet [20]	point	1k	86.2	89.2	11
PointNet++ [22]	point	1k	-	91.9	21
PointCNN [15]	point	1k	88.1	92.2	45
DGCNN [24]	point	1k	**90.2**	92.2	10
PointConv [32]	point	1k	-	92.5	58
KPConv [26]	point	1k	-	92.9	25
PCT1 [7]	point	1k	-	92.8	34
PCT2 [8]	point	1k	-	93.2	18
SWPT (ours)	point	1k	90.1	**93.5**	5

step schedule to adjust it at every 40 epochs. For a fair comparison of efficiency, we use the same hardware environments and only the official code from github. We experiment in the same test datasets and report the average latency time.

The results are presented in Table 1. The SWPT outperforms all prior methods in overall accuracy with 93.5%. As for Transformer-based methods, the SWPT achieves a better result with less time.

4.2 Part Segmentation

Data and metric. Part Segmentation is a challenging task which aims to divide a 3D point cloud into multiple meaningful parts. We evaluate the models on the ShapeNet [37] dataset which contains 16,880 3D models consisting of 14,006 models for training and 2,874 models for testing. It has 16 categories and 50 parts, where each category has a number of parts between 2 to 6. Following PointNet [20], we downsample each model to 2,048 points with point-wise labels, which is widely used in prior works. For metrics, we evaluate the mean Intersection-over-Union (IoU) and IoU for each object category.

Experiment results. During training we used a random translation in [0.2, 0.2], a random anisotropic scaling in [0.5, 1.5] and a random input dropout as the data augmentation strategy, while no data augmentation was used in testing. We set the batch size to 32, epochs to 200 and initial learning rate to 0.001 with a step schedule to adjust it at every 40 epochs by 0.5. Table 2 shows the mean part IoU and category-wise IoU. The results show that our SWPT improves by 1.7% over PointNet and is competitive with most SOTA methods. The reason why SWPT does not achieve the best result in mIoU is that projecting overlapped and shaded points (of objects like motorbikes and cars) on a spherical

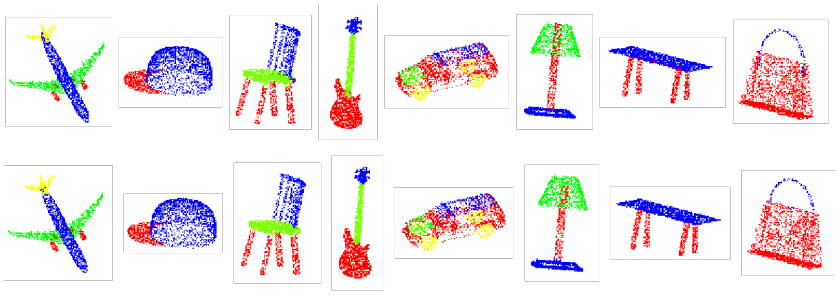

Fig. 5. Visualization of part segmentation results on the ShapeNet dataset. The top row shows the ground truth, results of SWPT are on the bottom row.

surface may cause semantic ambiguity issues.

Visualization. Fig 5 shows the results of object part segmentation on a number of models in the ShapeNet dataset. The predictions of SWPT on the bottom row are precision and close to the ground truth, from which we can see some structural details captured by SWPT, such as the engines of the plane and wheels of the automobile.

4.3 Real-World Object Classification

Data and metric. While ModelNet40 is the most popular benchmark for point cloud classification, it may lack a practical scenario due to its synthetic nature (i.e. complete, well-segmented and noisy-free). To evaluate the performance of our model on real-world objects, we conduct experiments on the ScanObjectNN benchmark which contains 15,000 objects categorized into 15 classes with 2,902 unique instances in read world. We adopt the following variants of ScanObjectNN in our experiments: (1)**OBJ_ONLY** which has only ground truth objects segmented from the scene datasets, and (2)**PB_T50_RS** which is the hardest variant for training and testing our model. For metrics, we report the overall accuracy (OA) over all classes. We use

Experiment results. We use the same batch size, training epochs and other

Table 2. Comparison of part segmentation models on the ShapeNet. mIoU means part-average Intersection-over-Union. All results are quoted from the cited papers.

Method	mIoU	aero	bag	cap	car	chair	ear phone	guitar	knife	lamp	laptop	motor	mug	pistol	rocket	skate board	table
Yi[38]	81.4	81.0	78.4	77.7	75.7	87.6	61.9	92.0	85.4	82.5	95.7	70.6	91.9	**85.9**	53.1	69.8	75.3
Kd-Net [12]	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 [20]	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++ [22]	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
PointCNN [15]	86.1	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
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
PointConv [32]	85.7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PCT1 [7]	85.9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PCT2 [8]	**86.4**	**85.0**	82.4	**89.0**	**81.2**	**91.9**	71.5	91.3	88.1	**86.3**	95.8	64.6	95.8	83.6	62.2	77.6	83.7
SWPT (ours)	85.4	82.2	79.2	83.1	74.3	91.7	74.7	91.7	86.0	80.6	**97.7**	55.8	**96.6**	83.3	53.0	74.5	**83.9**

settings as in experiments on ModelNet40. As presented in Table 3, our SWPT achieves competitive results on both OBJ_ONLY and PB_T50_RS datasets compared with previous methods. There is still a small gap between SWPT and the SOTA method, mostly because of the dimensionality reduction caused by projection, however which significantly accelerates the speed of model inference.
Visualization. Fig 6 illustrates the confusion matrices of the mainstream methods on the PB_T50_RS dataset which is the hardest variant. It can be seen that SWPT only has ambiguity issues between box and pillow, while PointNet and PointNet++ have ambiguity issues between table and desk besides. Even the PCT which obtains a good experimental result on the synthetic dataset has some ambiguity such as pillow vs bag and table vs desk. DGCNN has less ambiguity, which might benefit from the EdgeConv module recomputing graphs dynamically in each layer.

4.4 Ablation Study

We construct a number of ablation studies to analyze the performance of different components in SW-PCT. All studies are performed on the shape classification.
Spherical Window size. We first evaluate the setting of window size (θ, φ), which determines the local areas to performe self-attention. As presented in Table 4, SWPT achieves the best performance when the window size is set to $(4, 4)$. When the windows size gets smaller, the model may not have sufficient context to compute self-attention, leading to some local detail loss. When the window size gets larger, the local area may have more data points with fewer relationships, which introduces extreme noise into self-attention computation, reducing the model's accuracy.
Cross-Attention. We conducted an ablation study on the Cross-Attention layer. As shown in Table 5, we investigate global self-attention, local self-attention and cross-attention with different times in each layer. The results indicate adopting cross-attention twice in each layer achieves the best performance, while con-

Table 3. Comparision of state-of-the-art methods on ScanObjectNN. All results are quoted from [27] except for the PCT.

Method	OBJ_ONLY(%)	PB_T50_RS(%)
3DmFV [2]	73.8	63
PointNet [20]	79.2	68.2
SpiderCNN [35]	79.5	73.7
PointNet++ [22]	84.3	77.9
DGCNN [24]	**86.2**	78.1
PointCNN [15]	85.5	78.5
BGA-DGCNN [27]	-	79.7
BGA-PN++ [27]	-	**80.2**
PCT [8]	80.7	71.4
SWPT (ours)	85.1	77.2

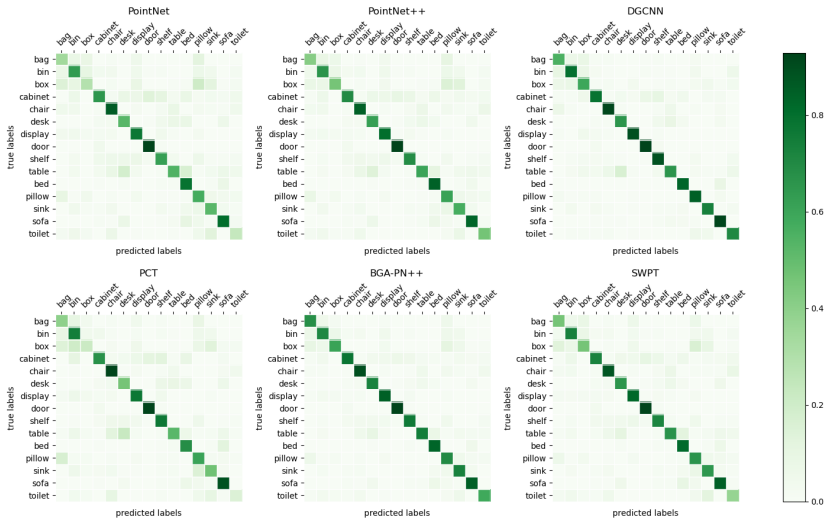

Fig. 6. Confusion matrices of some previous methods on the hardest real-world dataset PB_T50_RS.

tinuously increasing cross-attention within a layer reduces the performance. This suggests that relationships between windows are critical in learning local features.

Attention type. We now study the attention type in self-attention layers. The results are shown in Table 6. It shows that scalar attention is more expressive than vector attention in our model. In addition, vector attention consumes more memory and time.

Pooling type. Finally, we investigate the type of pooling used for gathering local features from a window. As shown in Table 7, Max-pooling outperforms other methods even the 'Max-Ave' pooling which is a concatenation of the results of Max-pooling and Average-pooling. 'Con-Pool' stands for the operation which concatenates all the features in a spherical window. This indicates that Max-pooling captures the most expressive features in the local area.

Table 4. Ablition study: window size to perform self-attention

size	layers	mAcc	OA
(2,2)	5	88.7	90.8
(3,3)	4	89.6	92.4
(4,4)	3	90.1	93.5
(8,8)	2	89.4	92.3

Table 5. Ablition study: Cross-Attention

type	mAcc	OA
Local-AT	88.0	90.1
Cross-AT	89.5	92.2
Cross-AT × 2	90.1	93.5
Cross-AT × 3	89.7	92.7

Table 6. Ablition study: Cross-Attention

type	mAcc	OA
scalar attention	90.1	93.5
vector attention	89.9	93.2

Table 7. Ablation study: type of pooling to gather local features, Con-Pool refers to the concatenation of features in a window.

Pooling	mAcc	OA
Max-Pool	90.1	93.5
Avg-Pool	87.9	90.2
Max-Avg	88.5	91.4
Con-Pool	88.6	92.3

5 Conclusion

In this paper, we present a Transformer-based architecture for 3D point cloud recognition which achieves competitive results with more efficiency compared with previous methods. To this end, we project the points on a spherical surface to reduce the dimension of each point, followed by a spherical window layer to perform local self-attention. We also introduce crossing window self-attention module to capture hierarchical features, which is proved that the relationships between different windows are effective to improve the accuracy of recognition tasks. In the future, we expect to study the semantic properties extracting, which would promote the network for other tasks such as 3D point cloud generation and 3D object detection.

Acknowledgements This work was supported by the National Natural Science Foundation of China (NSFC, No. 62272426) and the Research Project by Shanxi Scholarship Council of China (No. 2020-113).

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)

2. Ben-Shabat, Y., Lindenbaum, M., Fischer, A.: 3d point cloud classification and segmentation using 3d modified fisher vector representation for convolutional neural networks. arXiv preprint arXiv:1711.08241 (2017)
3. Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3d object detection network for autonomous driving. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 1907–1915 (2017)
4. Dai, A.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
5. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
6. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
7. Engel, N., Belagiannis, V., Dietmayer, K.: Point transformer. IEEE Access **9**, 134826–134840 (2021)
8. Guo, M.H., Cai, J.X., Liu, Z.N., Mu, T.J., Martin, R.R., Hu, S.M.: Pct: Point cloud transformer. Computational Visual Media **7**(2), 187–199 (2021)
9. Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., Markham, A.: Randla-net: Efficient semantic segmentation of large-scale point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11108–11117 (2020)
10. Hua, B.S., Pham, Q.H., Nguyen, D.T., Tran, M.K., Yeung, S.K.: Scenenn: A scene meshes dataset with annotations. In: Fourth International Conference on 3d Vision (2016)
11. Joseph-Rivlin, M., Zvirin, A., Kimmel, R.: Momen(e)t: Flavor the moments in learning to classify shapes (2018)
12. 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. pp. 863–872 (2017)
13. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12697–12705 (2019)
14. Li, G., Muller, M., Thabet, A., Ghanem, B.: Deepgcns: Can gcns go as deep as cnns? In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 9267–9276 (2019)
15. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: Pointcnn: Convolution on x-transformed points. Advances in neural information processing systems **31** (2018)
16. Lin, Z., Feng, M., Santos, C.N.d., Yu, M., Xiang, B., Zhou, B., Bengio, Y.: A structured self-attentive sentence embedding. arXiv preprint arXiv:1703.03130 (2017)
17. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10012–10022 (2021)
18. Maturana, D., Scherer, S.: Voxnet: A 3d convolutional neural network for real-time object recognition. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 922–928. IEEE (2015)
19. Paszke, A., Gross, S., Massa, F., Lerer, A., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library (2019)

20. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017)
21. Qi, C.R., Su, H., Nießner, M., Dai, A., Yan, M., Guibas, L.J.: Volumetric and multi-view cnns for object classification on 3d data. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5648–5656 (2016)
22. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in neural information processing systems **30** (2017)
23. Riegler, G., Osman Ulusoy, A., Geiger, A.: Octnet: Learning deep 3d representations at high resolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3577–3586 (2017)
24. Simonovsky, M., Komodakis, N.: Dynamic edge-conditioned filters in convolutional neural networks on graphs. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3693–3702 (2017)
25. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view convolutional neural networks for 3d shape recognition. IEEE (December 2015)
26. Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: Kpconv: Flexible and deformable convolution for point clouds. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 6411–6420 (2019)
27. Uy, M.A., Pham, Q.H., Hua, B.S., Nguyen, T., Yeung, S.K.: Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. IEEE (2020)
28. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
29. Wang, P.S., Liu, Y., Guo, Y.X., Sun, C.Y., Tong, X.: O-cnn: Octree-based convolutional neural networks for 3d shape analysis. ACM Transactions On Graphics (TOG) **36**(4), 1–11 (2017)
30. Wei, X., Yu, R., Sun, J.: View-gcn: View-based graph convolutional network for 3d shape analysis. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1847–1856. IEEE (2020)
31. Wu, B., Xu, C., Dai, X., Wan, A., Zhang, P., Yan, Z., Tomizuka, M., Gonzalez, J., Keutzer, K., Vajda, P.: Visual transformers: Token-based image representation and processing for computer vision. arXiv preprint arXiv:2006.03677 (2020)
32. Wu, W., Qi, Z., Fuxin, L.: Pointconv: Deep convolutional networks on 3d point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
33. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1912–1920 (2015)
34. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1912–1920 (2015)
35. Xu, Y., Fan, T., Xu, M., Long, Z., Yu, Q.: Spidercnn: Deep learning on point sets with parameterized convolutional filters (2018)
36. Yang, Z., Wang, L.: Learning relationships for multi-view 3d object recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7505–7514 (2019)

37. Yi, L., Kim, V.G., Ceylan, D., Shen, I.C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., Guibas, L.: A scalable active framework for region annotation in 3d shape collections. ACM Transactions on Graphics (ToG) **35**(6), 1–12 (2016)
38. Yi, L., Kim, V.G., Ceylan, D., Shen, I.C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., Guibas, L.: A scalable active framework for region annotation in 3d shape collections. ACM Transactions on Graphics (ToG) **35**(6), 1–12 (2016)