PatchFormer: An Efficient Point Transformer with Patch Attention

Cheng Zhang[1]*, Haocheng Wan[1]*, Xinyi Shen[2], Zizhao Wu[1]†
[1]Hangzhou Dianzi University, Hangzhou China
[2]University College London, London UK
{zhangcheng828,wuzizhao}@hdu.edu.cn {wanhaocheng2022,xinyishen2018}@163.com

Abstract

*The point cloud learning community witnesses a modeling shift from CNNs to Transformers, where pure Transformer architectures have achieved top accuracy on the major learning benchmarks. However, existing point Transformers are computationally expensive since they need to generate a large attention map, which has quadratic complexity (both in space and time) with respect to input size. To solve this shortcoming, we introduce **P**atch **AT**tention (PAT) to adaptively learn a much smaller set of bases upon which the attention maps are computed. By a weighted summation upon these bases, PAT not only captures the global shape context but also achieves linear complexity to input size. In addition, we propose a lightweight **M**ulti-**S**cale a**T**tention (MST) block to build attentions among features of different scales, providing the model with multi-scale features. Equipped with the PAT and MST, we construct our neural architecture called PatchFormer that integrates both modules into a joint framework for point cloud learning. Extensive experiments demonstrate that our network achieves comparable accuracy on general point cloud learning tasks with 9.2× speed-up than previous point Transformers.*

1. Introduction

Transformer has recently drawn great attention in natural language processing [7, 34] and 2D vision [8, 21, 33, 38] because of its superior capability in capturing long-range dependencies. Self-Attention (SA), the core of Transformer, obtains an attention map by computing the affinities between self $queries$ and self $keys$, generating a new feature map by weighting the self $values$ with this attention map. Benefitting from SA module, Transformer is capable of modeling the relationship of tokens in a sequence, which is also important to many point cloud learning tasks. Hence, plenty of researches have been done to explore Transformer-based point cloud learning architectures.

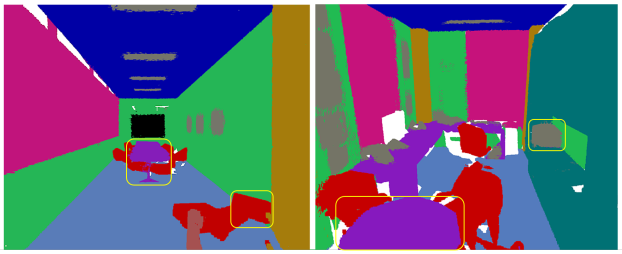

Figure 1. A large indoor scene often consists of small instances (e.g., chair and typewriter) and large objects (e.g., table and blackboard), building the relationships among them requires a multiscale attention mechanism.

Recently, Nico et al. proposed PT[1] [9] to extract global features by introducing the standard SA mechanism, which aims to capture spatial point relations and shape information. Guo et al. proposed offset-attention (PCT [10]) to calculate the offset difference between the SA features and the input features by element-wise subtraction. Lately, more and more researchers have applied SA module to various point cloud learning tasks and achieved significant performance such as [24, 56]. However, existing point Transformers are computationally expensive because the original SA module needs to generate a large attention map, which has high computational complexity and occupies a huge number of GPU memory. This bottleneck lies in that both the generation of attention map and its usage require the computation with respect to all points.

Towards this issue, we propose a novel lightweight attention mechanism, namely PAT which calculates the attention map via low-rank approximation [17, 52]. Our key observation is that a 3D shape is composed of its local parts and thus the features of points in the same part should have similar semantics. Based on this observation, we first exploit the intrinsic geometry similarity, cluster local points on a 3D shape as one patch and estimate a base by aggregating the features of all points in the same patch. Then we use a product of self $queries$ and self bases to approximate the

*These authors contributed equally.
†Corresponding author: wuzizhao@hdu.edu.cn.

global attention map, which can be obtained by computing self $queries$ and self $keys$. Notably, the representation of such product is low-rank and discards noisy information from the input.

In addition, to aggregate local neighborhood information, Zhao et al. [56] proposed PT^2 to build local vector attention in neighborhood point sets, Guo et al. (PCT [10]) proposed to use a neighbor embedding strategy to improve point embedding. Though PT^2 and PCT have achieved significant progress, there exist problems that restrict their efficiency and performance. First, they wastes a high percentage of the total time on structuring the irregular data, which becomes the efficiency bottleneck [23]. Second, they fails to build the attentions among features of different scales which is very important to 3D visual tasks. As shown in Fig 1, a large indoor scene often contains small instances (e.g., chair and lamp) and large objects (e.g., table), building the relationships among them required a multi-scale attention mechanism. However, the input sequence of PT^2 and PCT is generated from equal-sized points, so only one single scale feature will be preserved in the same layer.

To solve these issues, we present a lightweight **M**ulti-**S**cale a**T**tention (MST) block for point cloud learning, which consists of two steps. In the first step, our MST block transforms point cloud into voxel grids, sampling boxes with multiple convolution kernels of different scales and then concatenates these grids as one embedding (see Fig 4). Specifically, we propose to use the depth-width convolution (DWConv [11]) on boxes sampling beacuse of both few parameters and FLOPs. In the second step, we incorporate 3D relative position bias and build attentions to non-overlapping local 3D window, providing our model with strong multi-scale features at a low computational cost.

Based on these proposed blocks, we construct our neural architecture called PatchFormer (see Fig 2). Specifically, we perform the classification task on the ModelNet40 and achieve the strong accuracy of 93.5% (no voting) with 9.2× faster than previous point Transformers. On ShapeNet and S3DIS datasets, our model also obtains strong performance with 86.5% and 68.1% mIoU, respectively.

The main contributions are summarized as following:

- We present PatchFormer for efficient point cloud learning. Experiments show that our network achieves strong performance with 9.2× speed-up than prior point Transformers.

- We propose PAT, the first linear attention mechanism in the point cloud analysis paradigm.

- We present a lightweight voxel-based MST block, which compensates for previous architectures' disability of building multi-scale relationship.

2. Related works

2.1. Transformer for 2D Vision

Motivated by the success of Transformers in NLP [6, 14, 29, 34, 48], researchers designed visual Transformers for vision tasks to take advantage of their great attention mechanism. In particular, Vision Transformer (ViT) [8] is the first such example of a Transformer-based approach to match or even surpass convolution neural networks (CNNs) for image classification. Later, Wang et al. [37] proposed pyramid structure into transformers, named PVT, greatly decreasing the number of patches in the later layers of the model. Liu et al. [21] proposed Swin Transformer whose representation is computed with non-overlapping local windows. Subsequently, Wang et al. [38] and Chen et al. [2] proposed CrossFormer and CrossViT to study how to learn multi-scale features in Transformers.

Inspired by the cross-scale attention used in CrossFormer and CrossViT for image analysis, we present a voxel-based MST block for point cloud learning that combines voxel grids of different sizes to learn stronger local features.

2.2. Point Cloud Learning

Most existing point cloud learning methods could be classified into two categories in terms of data representations: the *voxel*-based models and the *point*-based models. The *voxel*-based models generally rasterize point clouds onto regular grids and apply 3D convolution for feature learning [13, 15, 27, 40, 57]. These models are computationally efficient due to their excellent memory locality, but suffer from the inevitable information degrades on the fine-grained localization accuracy [23, 31, 51]. Instead of voxelization, developing a neutral network that consumes directly on point clouds is possible [12, 20, 26, 30, 35, 39, 44, 47, 50, 53, 54]. Although these *point*-based models naturally preserve accuracy of point location, they are usually computationally intensive.

Generally, the voxel-based models have regular data locality and can efficiently encode coarse-grained features, while the point-based networks preserve accuracy of location information and can effectively aggregate fine-grained features. In this paper, we propose PatchFormer to incorporate both the advantages from the two models mentioned above.

2.3. Point Transformers

Powered by Transformer [34] and its variants [8, 21], the point-based models have recently applied SA to extract features from point clouds and improve performance significantly [9, 10, 16, 25, 49, 51]. In particular, PT^1 is the first such example of a Transformer-based approach for point cloud learning. Later, Guo et al. and Zhao et al. pro-

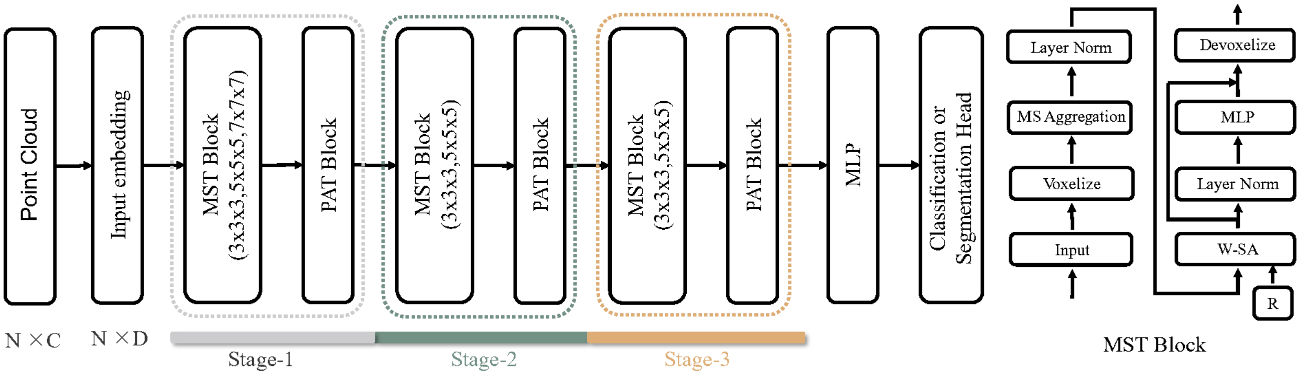

Figure 2. **The architecture of PatchFormer**: PatchFormer is comprised of three stages and each stage contains two blocks: MST block and PAT bolck. A specialized head (e.g., the classification head) is followed by the final stage for the specific task. **MST block:** It first voxelizes point cloud into voxel grids, aggregating multi-scale features, and then conducts 3D window-based SA (W-SA) to capture local information. Finally, MST block transforms voxel grids to points and feed them into PAT block. Numbers in MST represent the size of kernels used DWConv, R denotes the relative position bias and W-SA refers to 3D window-based self-attention.

posed PCT and PT^2 to construct SA networks for general 3D recognition tasks.

However, they suffer from the fact that as the size of the feature map increases, the computing and memory overheads of the original SA increase quadratically. To address this issue, we propose PAT to compute the relation between self $queries$ and a much smaller bases, yet captures the global context of a point cloud as well.

3. Overview

An overview of the PatchFormer architecture is presented in Fig 2. Our method first embeds a point cloud $\mathcal{P}$ into a D dimensional space $F \in \mathbb{R}^{N \times D}$ using a shared MLP, where N is the number of points. We empirically set D = 128, a relatively small value for computational efficiency. Late, we split our model into three stages and each stage is comprised of two blocks: **M**ulti-**S**cale a**T**tention (MST) block and **P**atch-**AT**tention (PAT) block.

As illustrated in Fig 2, the MST block first voxelizes a point cloud into regular voxel grids and then feeds them into a multi-scale aggregating module. In this module, we sample boxes using three DWConv kernels of different size and concatenate them as one embedding. After that, we limit SA computation to non-overlapping local boxes in order to alleviate the quadratic complexity of the original SA. Note that a LayerNorm (LN) layer is applied before W-SA module and MLP module, and a residual connection is applied after each module. Eventually, we leverage the trilinear interpolation to transform the voxel grids to points.

Like ViT [8], the PAT block treats each point as a "token" and aggregates global feature by using patch attention. It receives MST block's output as input, estimates a much more compact bases and generates global attention map upon these bases. Note that, our method reduces the complexity (both in space and time) from $\mathcal{O}(N^2)$ of the original SA to $\mathcal{O}(MN)$ $(M << N)$ where M is the number of bases. As a result, the proposed patch attention can conveniently replace the backbone networks in existing point Transformers for various point cloud learning tasks.

Throughout the next sections we use the following notations: the original point cloud with N points is denoted by $\mathcal{P} = \{p_i\}_{i=1}^{N} \subseteq \mathbb{R}^C$. In the simplest setting of C = 3, each point contains 3D coordinates. $F = \{f_i\}_{i=1}^{N} \subseteq \mathbb{R}^D$ is the input embedding feature.

4. Method

In this section, we first analysis the original SA mechanism, then we detail our novel way to define attention: patch attention. And finally, we discuss the design of MST block in detail.

4.1. Self-Attention

We first revisit the self-attention (SA) mechanism. The standard SA, also called scalar dot-product attention, is a mechanism that calculates semantic affinities among different elements within a sequence of data. Following the terminology in [34], let Q, K, V be the $query, key$ and $value$ matrices, respectively, generated by linear transformations of the input features $F \in \mathbb{R}^{N \times D}$ as follows

$$(Q, K, V) = (W_q, W_k, W_v)F, \tag{1}$$

$$Q, K, V \in \mathbb{R}^{N \times D}, \tag{2}$$

where W_q, W_k and W_v are the shared learnable linear transformation as illustrated in Fig 3.

Using the pairwise dot product $QK^T \in \mathbb{R}^{N \times N}$, then SA

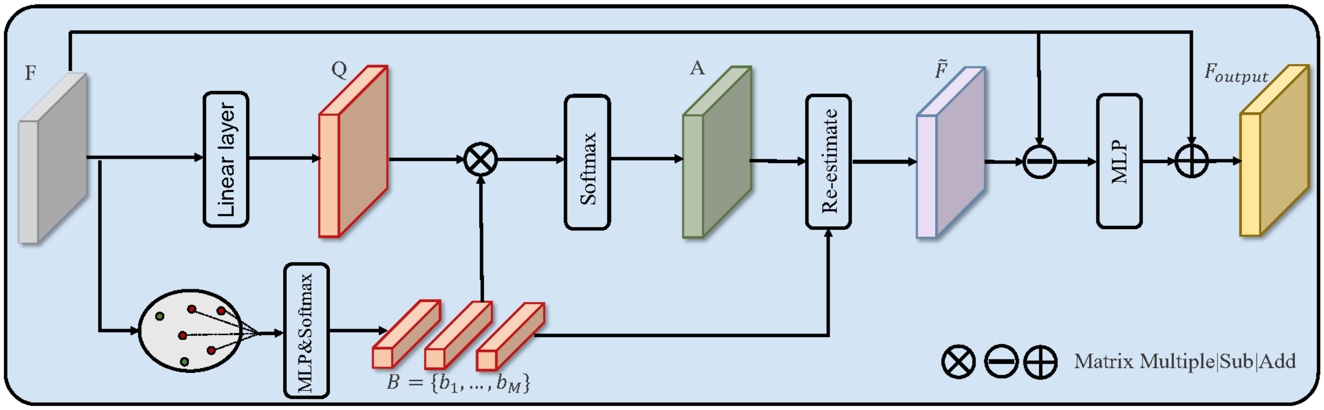

Figure 3. Architecture of PAT block (patch attention). PAT can be seen as an variant to the original self-attention to approximate global map at a lower computational cost.

can be formulated as:

$$A = (\alpha_{i,j}) = softmax(QK^T), \tag{3}$$

$$F_{out} = AV, \tag{4}$$

where $A \in \mathbb{R}^{N \times N}$ is the attention map and $\alpha_{i,j}$ is the pairwise affinity between (similarity of) the i-th and j-th elements. It is apparent that the output F_{output} is a weighted sum of V, where a value gets more weight if the similarity between the keys and values yields a higher attention weighting score.

However, the high computational complexity of $\mathcal{O}(N^2 D)$ presents a significant drawback to use of SA. The quadratic complexity in the number of input points makes it infeasible to apply SA to point cloud directly.

4.2. Patch Attention

In view of the high computational complexity of the attention mechanism and limitations, we first propose the PAT, which is an augmented version of SA. Unlike prior point Transformers obtain an attention map by computing affinities between self $queries$ and self $keys$, our patch attention (PAT) computes the relation between self $queries$ and a much smaller bases, yet captures the global context of a point cloud.

For simplicity, we consider an input point cloud $\mathcal{P}$ and its corresponding feature map F of size $N \times D$, our proposed PAT is illustrated in Fig 3 which consists of two steps, including *base estimation* and *data re-estimation*.

Base Estimation. In this step, we estimate a compact basis set $B \in \mathbb{R}^{M \times D}$ where M is the number of bases. In particular, we introduce the concept of patch-instance base. For each point cloud $\mathcal{P}$ in the dataset, we over-segment it into M patches ($M << N$) and based on which, we create M patch-instance bases. In this way, the global shape can be approximated by the set of each patch-instance base, which have a less total number. For simplicity, we use the K-Means algorithm to segment $\mathcal{P}$ into M patches $\{S_1, S_2, ..., S_M\}$, M=96, by default in classification task. We define each base as b_m by aggregating the representations of all the points in the S_m, it can be described as:

$$b_m = \sum_{f_i \in S_m} w_i(\varphi(f_i)), \tag{5}$$

$$B = \{b_m\}_{m=1}^{M} \subseteq \mathbb{R}^D. \tag{6}$$

Here, f_i is the representation of point p_i, the transformation function $\varphi(\cdot)$ is an MLP with one linear layer and one ReLU nonlinearity, w_i is the normalized degree for f_i belonging to the S_m. We use spatial softmax to normalize each patch.

Generally, our base estimation method can adaptively adjust the contribution of all points in the same patch to the base via a data-driven way. Such adaptive adjusting facilitates to fit the intrinsic geometry submanifold.

Data Re-estimation. After estimating the bases B, we can replace K matrices with B and re-formulate Eq 3 as:

$$A = softmax(QB^T), \tag{7}$$

where $A \in \mathbb{R}^{N \times M}$ is the attention map constructed from a compact basis set. After that, the final bases B and attention map A are used to re-estimate the inputs F. We formulate a new equation to re-estimate the F using $\tilde{F}$ as follows:

$$\tilde{f}_i = \sum_{m=1}^{M} A_i^m b_m, \tag{8}$$

$$\tilde{F} = \{\tilde{f}_i\}_{i=1}^{N} \subseteq \mathbb{R}^D. \tag{9}$$

As $\tilde{F} \in \mathbb{R}^{N \times D}$ is constructed from a compact basis set B, it has the low-rank property compared with the input F.

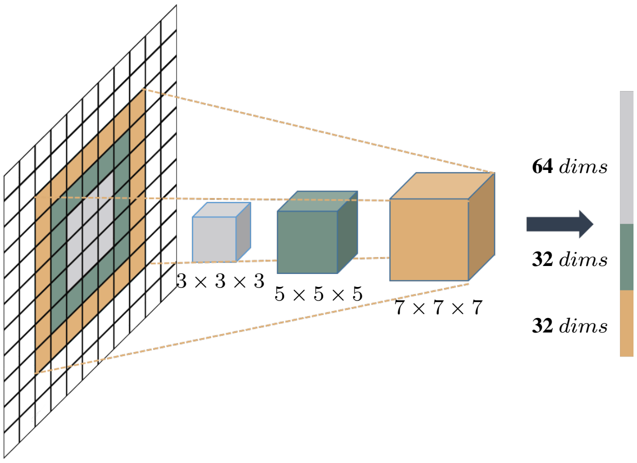

Figure 4. Illustration of multi-scale feature aggregating in MST block on Stage-1. We note that this is a 2D example and can be easily extended to 3D cases. The input voxel grids is sampled by three DWConv kernels (i.e., $3 \times 3 \times 3$, $5 \times 5 \times 5$, $7 \times 7 \times 7$) with stride $1 \times 1 \times 1$. Each embedding is constructed by projecting and concatenating the three 3D boxes.

Inspired by PCT [10], we calculate the difference between the estimated features $\tilde{F}$ and the input features F by element-wise subtraction. Finally, we feed the difference into MLP layer and adopt residual connection strategy to help propagate information to higher layers. This step can be formulated as:

$$F_{output} = \phi(\tilde{F} - F) + F, \tag{10}$$

where $F_{ouput} \in \mathbb{R}^{N \times D}$ is the outputs of our PAT block and $\phi(\cdot)$ is an MLP with one linear layer and one ReLU nonlinearity.

Complexity Analysis. Compared with the standard SA module, our PAT finds a representative set of bases for points of a point cloud, which reduces the complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(MN)$ ($M << N$), where M and N are the number of bases and points, respectively. Moreover, we only need to calculate K-Means algorithm once on the original point cloud $\mathcal{P}$ that can be accelerated in parallel by CUDA. Although the K-Means optimization has an asymptotic complexity $\mathcal{O}(NMC)$, it can be ignored in our network because M is fixed and $C = 3$.

4.3. Multi-Scale Attention

In this subsection, we detail how our MST block learns multi-scale feature representations in attention models. This block consists of two steps, including *Multi-scale feature aggregating* and *Attention building*.

Multi-scale Feature Aggregating. This step is used to generate multi-scale features for each stage. Fig 4 illustrates the first MST block, which is ahead of the Stage-1, as an example. we receive voxel grids as input, sampling boxes using three kernels of different size. The strides of three kernels are kept the same so that they generate the same number of embeddings. As can be seen in Fig 4, every three corresponding boxes own the same center but locate at different scales. These three boxes will be projected and concatenated as one embedding. In practice, the process of sampling and projecting can be implemented through three DWConv layers. Note that we use a lower dimension for large kernels while a higher dimension for small kernels. Fig 4 provides the specific allocation rule in its subtable, where a 128 dimensional example is given. Compared with allocating the dimension equally, our scheme reduces computational cost while maintaining the model's high performance. The MST blocks in other stages work in a similar way. As shown in Fig 2, MST blocks in Stage-2/3 use two kernels ($3 \times 3 \times 3$ and $5 \times 5 \times 5$). The strides are set as $1 \times 1 \times 1$. For computing efficiency, DWConv with kernel sizes larger than $5 \times 5 \times 5$ is implemented by stacking multiple convolutions with kernel size $3 \times 3 \times 3$ and $5 \times 5 \times 5$.

Attention Building. To build the attentions among features of different scales. we attempt to conduct the standard SA on multi-scale feature map. However, the computation complexity of the full SA mechanism is quadratic to feature map size. Therefore, it will suffer from huge computation cost for 3D vision tasks that take high resolution feature maps as input, such as semantic segmentation.

To solve this shortcoming, our MST block limits the SA computation to non-overlapping local 3D windows. In addition, we observe that numerous previous works [21, 22, 41] have shown that it can be advantageous to include relative position bias in SA computation. Thus we introduce 3D relative position bias $R \in \mathbb{R}^{V^3 \times V^3}$ as

$$F_{output} = softmax(QK^T + R)V, \tag{11}$$

where $Q, K, V \in \mathbb{R}^{V^3 \times D}$ are the *query*, *key* and *value* matrices, and V^3 is the number of voxel grids in a local 3D window. Since the relative position along each axis lies in the range $[-V+1, V-1]$, we parameterize a smaller-sized bias matrix $\hat{R} \in \mathbb{R}^{(2V-1)\times(2V-1)\times(2V-1)}$, and values in R are taken from $\hat{R}$.

For cross-window information interaction, existing works [21, 33, 37] suggested to apply halo or shifted window to enlarge the receptive filed. However, the elements within each Transformer block still has limited attention area and requires stacking more blocks to achieve large receptive filed. In our network, the local attention is building in multi-scale input features. Thus, we don't need to stack more attention layers for cross-window connection or larger receptive filed.

Model	Input	OA	Latency
OA<92.5			
PointNet [26]	16×1024	89.2	**13.6ms**
PointNet++ [28]	16×1024	91.9	35.3ms
SpiderCNN [45]	8×1024	92.4	82.6ms
PointCNN [18]	16×1024	92.2	221.2ms
PointWeb [55]	16×1024	92.3	–
PVCNN [23]	16×1024	92.4	24.2ms
OA>92.5			
KPConv [32]	16×6500	92.9	120.5ms
DGCNN [39]	16×1024	92.9	85.8ms
LDGCNN [53]	16×1024	92.7	–
PointASNL [46]	16×1024	93.2	923.6ms
PT^1 [9]	16×1024	92.8	320.6ms
PT^2 [56]	8×1024	**93.7**	530.2ms
PCT [10]	16×1024	93.2	92.4ms
PatchFormer	16×1024	93.5	**34.3ms**

Table 1. Results on ModelNet40 [43]. Compared with previous Transformer-based models, our PatchFormer achieves the promising accuracy with 9.2× measured speed-up on average.

Model	Params	FLOPs	SDA(%)	OA(%)
PointNet	3.47M	0.45G	0.0	89.2
PointNet++(SSG)	1.48M	1.68G	43.5	90.7
PointNet++(MSG)	1.74M	4.09G	47.6	91.9
DGCNN	1.81M	2.43G	57.2	92.9
PointASNL	3.98M	5.92G	39.8	93.1
PT^1	21.1M	5.05G	32.5	92.8
PT^2	9.14M	17.1G	65.4	**93.7**
PCT	2.88M	2.17G	24.6	93.2
PatchFormer	**2.45M**	**1.62G**	**6.3**	93.5

Table 2. Computational resource requirements. SDA means the rate of total runtime on structuring the sparse data.

5. Experiments

In this section, we evaluate the proposed PatchFormer for different tasks: classification, part segmentation, and scene semantic segmentation. Performance is quantitatively evaluated using four metrics: mean class accuracy, overall accuracy (OA), per-class intersection over union (IoU), and mean IoU (mIoU). For fair comparison, we report the measured latency and model size on a RTX 2080 GPU to reflect the efficiency but evaluate other indicators on a RTX 3090 GPU. Please refer to our appendices for more detailed network architecture and experimental results.

Implementation details. We implement the PatchFormer in PyTorch. We use the SGD optimizer with momentum 0.9 and weight decay 0.0001, respectively. For 3D shape classification on ModelNet40 and 3D object part segmentation on ShapeNetPart, we train for 250 epochs. The initial learning rate is set to 0.01 and is dropped until 0.0001 by using cosine annealing. For semantic segmentation on S3DIS, we train for 120 epochs with initial learning rate 0.5, dropped by 10× at 50 epochs and 80 epochs.

5.1. Shape Classification

Data. We evaluate our model on the ModelNet40 [42] dataset. This dataset contains 12,311 computer-aided design models from 40 man-made object categories, in which 9,843 models are used for training and 2,468 models are used for testing. We follow the experimental configuration of Qi et al. [26]: (1) we uniformly sample 1,024 points from the mesh faces for each model; (2) the point cloud is rescaled to fit the unit sphere; and (3) the (x,y,z) coordinates and the normal of the sampled points are used in the experiment. During the training process, randomly scaling, translating and perturbing the objects are adopted as the data augmentation strategy in our experiment.

Results. The results are presented in Table 1. The overall accuracy of PatchFormer on ModelNet40 is 93.5%. It outperforms strong graph-based models such as DGCNN, strong point-based models such as KPConv and excellent attention-based networks such as PointASNL. Remarkably, compared with existing Transformer-based models such as PT^1,PT^2 and PCT, our model is 9.2× faster while achieving comparable accuracy.

5.2. Computational requirements analysis

We now consider the computational requirements of PatchFormer and several other baselines by comparing the floating point operations required (FLOPs) and number of parameters (Params) in Table 2. We evaluate these indicators on ModelNet40 dataset. From Table 2, we can see that PatchFormer has the lowest memory requirements with only 2.45M parameters and also puts a low load on the processor of only 1.62G FLOPs, yet delivers comparable accurate results of 93.5%. Notably, we summarize Table 2 the PatchFormer only spend 6.3% of the total runtime on structuring the irregular data, which is much lower than previous point Transformers. Compare with its baselines, PatchFormer has not only strong performance but also the lowest computational and memory requirements. These characteristics make PatchFormer suitable for deployment on a edge devices.

5.3. Object Segmentation

Data. We use the large-scale 3D dataset ShapeNet Parts [19] as the experiment bed. ShapeNet Parts contains 16,880 models (14,006 models are used for training, and 2874 models are used for testing), each of which is annotated with two to six parts, and the entire dataset has 50 differ-

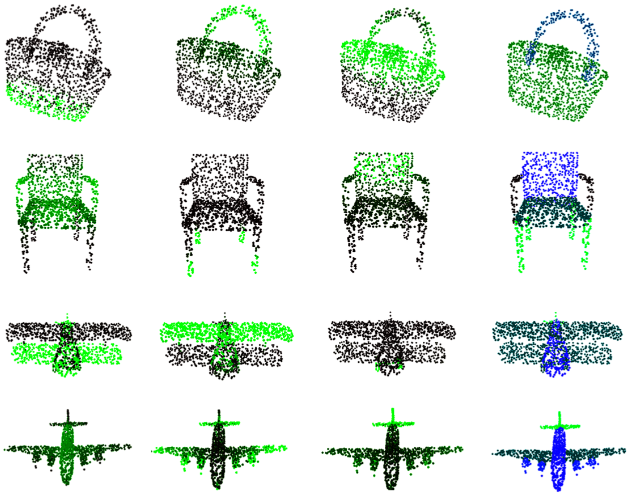

Figure 5. Attention map and segmentation results on ShapeNet. From left to right: attention maps w.r.t. three selected entries in the bases, segmentation results.

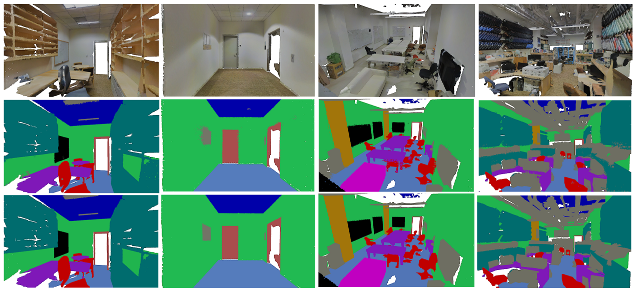

Figure 6. Visualization of semantic segmentation results on the S3DIS dataset. The input is in the top row, PatchFormer prediction is on the middle, the ground truth is on the bottom.

Model	Input	mIoU	Latency
DGCNN [39]	8×2048	85.4	96.2ms
PT^1 [9]	6×2048	85.9	360.4ms
PointCNN [18]	8×2048	86.1	145.6ms
PointASNL [46]	6×2048	86.1	1023.2ms
KPConv [32]	6×6500	86.4	127.8ms
PCT [10]	8×2048	86.4	101.1ms
PT^2 [56]	4×2048	**86.6**	560.2ms
PatchFormer	8×2048	86.5	**45.8ms**

Table 3. Results of part segmentation on ShapeNet Part.

ent part labels. We sample 2,048 points from each model as input, with a few point sets having six labeled parts. We directly adopt the same train–test split strategy similar to DGCNN [39] in our experiment.

Results and Visualization. From Table 3, we can see that with similar accuracy, our PatchFormer is 12.4× faster than PT^2 and 2.2× faster than PCT. Notably, with better accuracy, PatchFormer is 22.7× faster than PointASNL. In addition, we randomly select three entity from B in the last layer of our network and show their corresponding attention score of all points. As we can see, each basis corresponds to an abstract concept of the point cloud and the learned attention maps focus on meaningful parts for object segmentation as in Fig 5.

Model	Input	mIoU	Latency
DGCNN	8×4096	47.1	178.1ms
PointCNN [18]	4×4096	57.3	282.3ms
PointASNL [46]	4×4096	62.6	1895.2ms
PT^1 [9]	4×4096	63.1	1223.6ms
MinkowskiNet [5]	–	65.4	–
KPConv [32]	4×6500	67.1	267.5ms
PatchFormer	8×4096	68.1	**109.8ms**

Table 4. Indoor scene segmentation results on S3DIS, evaluated on Area5. From this table, we can see that PatchFormer outperforms most of previous models in accuracy and efficiency.

5.4. Indoor Scene Semantic Segmentation

Data. We evaluate our model on the S3DIS dataset [1], which contains 3D RGB point clouds from six indoor areas of three different buildings. Each point is marked with a semantic label from 13 categories (e.g., board, bookcase, chair, ceiling, and beam) plus clutter. Following a common protocol [26, 32], we divide and sample each room into 1 m × 1 m blocks, wherein each point is represented by a 9D vector (XYZ, RGB, and normalized spatial coordinates). In addition, the points in each block are sampled into a uniform number of 4,096 points during the training process, and all points are used in the test.

Results and Visualization. The results are presented in Tables 4. From this table we can see that our PatchFormer attains mIoU of 68.1%, which outperforms graph-based methods such as DGCNN [39], sparse convolutional networks such as MinkowskiNet [5], continuous convolutional networks such as KPConv [32], attention-based models such as PointASNL [46] and point Transformer such as PT^1. Remarkably, our PatchFormer also outperforms these powerful model by a large margin in latency.

Fig 6 shows the PatchFormer's predictions. We can see that the predictions are very close to the ground truth. PatchFormer captures detailed multi-scale features in complex 3D scenes, which is important in our network.

5.5. Ablation Studies

We now conduct a number of controlled experiments that examine specific decisions in the PatchFormer design.

Number of Bases. We first investigate the setting of the

M	ModelNet40(OA)	ShapeNet(mIoU)	Latency
32	91.54	84.92	**33.25ms**
64	92.94	85.82	33.82ms
96	**93.52**	86.52	34.32ms
128	93.50	**86.54**	35.56ms

Table 5. Ablation study: number of bases M in our network. We report latency on ModelNet40 dataset.

Ablation	ModelNet40(OA)	ShapeNet(mIoU)
w/o MS feature	92.85	85.22
MLP	92.62	85.32
EdgeConv	93.10	85.89
self-attention	93.29	86.22
no rel. pos	93.15	86.30
Ours	**93.52**	**86.52**

Table 6. Ablation study on the multi-scale feature aggregation, PAT and relative bias on two benchmarks. w/o MS feature: all MST block without aggregate multi-scale features. MLP: replace PAT with MLP layer in our architecture. EdgeConv: replace PAT with EdgeConv layer in our architecture. self-attention: replace PAT with self-attention layer in our architecture. rel. pos: the default settings with an additional relative position bias term.

Ablation	ModelNet40(OA)	Latency(ms)
A^2 Net [3]	92.89	36.89
EMA Net [17]	93.02	37.27
Linformer [36]	93.14	40.22
Performer [4]	93.22	35.46
Ours	**93.52**	**34.32**

Table 7. We replace our PAT with other linear attention mechanisms. We collect their public code and adapt them to 3D data.

number of bases. The results are shown in Table 5. The best performance of classification task is achieved when M is set to 96. On the one hand, when the bases is smaller (M = 32 or M = 64), the model may not have sufficient context for its predictions. On the other hand, increasing M doesn not give PatchFormer much accuracy benefit but incurs a raise on latency. This also demonstrates the efficiency and effectiveness of our PAT.

Effect of Multi-scale feature aggregating. We conduct an ablation study on the Multi-scale feature aggregating step. From Table 6, we can see the performance without this step on ModelNet40 and ShapeNet are 92.85%/85.22%, in terms of OA/mIoU. It is much lower than the performance with Multi-scale feature (93.52%/86.52%). This suggests that the Multi-scale feature is essential in this setting.

Impact of PAT. We investigate the impact of PAT used in the PAT block. From Table 6, we can see that PAT is more effective than the no-attention baseline (MLP). The performance gap between PAT and MLP baseline is significant: 93.52% vs. 92.62% and 86.52% vs. 85.32%, an improvement of 0.9 and 1.2 absolute percentage points. Compared with EdgeConv baseline, our PAT also achieves improvements of 0.42 and 0.63 absolute percentage points. Notably, our PAT outperforms the self-attention baseline with 0.23 and 0.30 absolute percentage points. We also compare PAT with other linear attention mechanisms in Table 7 and find it achieves the best accuracy and running speed. PAT has two obvious advantages. First, we only need to calculate K-Means once on the original point cloud due to the intrinsic geometry similarity, which means that the computational cost of the base estimation can be neglected. Second. PAT based on residual learning is more robust to any rigid transformation of objects.

Effect of 3D Relative position bias. Finally, we investigate the effect of 3D relative position bias used in the MAS block. Table 6 shows results. We can see that the PatchFormer with relative position bias yields +0.37% OA/+0.47% mIoU on ModelNet40 and ShapeNe in relation to those without position encoding respectively, indicating the effectiveness of the relative position bias.

6. Conclusion and Future Work

In this work, we propose a new type of attention mechanism, namely **P**atch **AT**tention (PAT) for point cloud learning, which computes a much smaller bases by exploiting the geometric similarity of nearby points. The reconstructed output of our PAT is low-rank and achieves linear time-space complexity to input size. Further, we propose a lightweight MST block, building attentions among features of different scales and providing our model with multi-scale features. Based on these modules, we construct PatchFormer for various point cloud learning task. Experiments show that our PatchFormer achieves comparable accuracy and better speed than other point Transformers.

We hope that our work will provide empirical guidelines for new method design and inspire further investigation of the properties of point Transformers. For example, performing K-Means in the points, extracting a patch feature for each cluster, directly reducing the number of tokens in the embedding stage.

7. Acknowledgements

This work was partially supported by the Zhejiang Provincial Natural Science Foundation of China (LGF21F20012), the National Natural Science Foundation of China (No.61602139), and the Graduate Scientific Research Foundation of Hangzhou Dianzi University (CXJJ2021082, CXJJ2021083).

References

[1] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017. 7

[2] C. Chen, Q. Fan, and R. Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 2

[3] Y. Chen, Y. Kalantidis, J. Li, S. Yan, and J. Feng. A^2-nets: Double attention networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 8

[4] K. M. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlós, P. Hawkins, J. Q. Davis, A. Mohiuddin, L. Kaiser, D. B. Belanger, L. J. Colwell, and A. Weller. Rethinking attention with performers. *International Conference on Learning Representations*, 2021. 8

[5] C. Choy, J. Y. Gwak, and S. Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. 7

[6] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context. *in ACL*, 2019. 2

[7] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *in NAACL-HLT*, 2018. 1

[8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations*, 2021. 1, 2, 3

[9] N. Engel, V. Belagiannis, and K. Dietmayer. Point transformer. *CoRR*, abs/2011.00931, 2020. 1, 2, 6, 7

[10] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, Apr 2021. 1, 2, 5, 6, 7

[11] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2

[12] Q. Huang, W. Wang, and U. a. Neumann. Recurrent slice networks for 3d segmentation of point clouds. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 2

[13] T. Le and D. Ye. Pointgrid: A deep network for 3d shape understanding. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 2

[14] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *in Bioinformatics*, pages 1234–1240, 2020. 2

[15] J. Li, B. M. Chen, and G. H. Lee. So-net: Self-organizing network for point cloud analysis. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018. 2

[16] R. Li, X. Li, P. Heng, and C. Fu. Point cloud upsampling via disentangled refinement. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 344–353, 2021. 2

[17] X. Li, Z. Zhong, J. Wu, Y. Yang, and H. Liu. Expectation-maximization attention networks for semantic segmentation. *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 1, 8

[18] Y. Li, R. Bu, M. Sun, and B. Chen. Pointcnn: Convolution on x-transformed points. *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 6, 7

[19] Y. Li, V. G. Kim, D. Ceylan, I. C. Shen, M. Yan, S. Hao, C. Lu, Q. Huang, A. Sheffer, and L. Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (TOG)*, 35(6cd):210.1–210.12, 2016. 6

[20] Y. Liu, B. Fan, S. Xiang, and C. Pan. Relation-shape convolutional neural network for point cloud analysis. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. 2

[21] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 1, 2, 5

[22] Z. Liu, J. Ning, Y. Cao, Y. Wei, Z. Zhang, S. Lin, and H. Hu. Video swin transformer. *CoRR*, abs/2106.13230, 2021. 5

[23] Z. Liu, H. Tang, Y. Lin, and S. Han. Point-voxel cnn for efficient 3d deep learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 2, 6

[24] K. Mazur and V. Lempitsky. Cloud transformers. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 1

[25] L. Pan, X. Chen, Z. Cai, J. Zhang, H. Zhao, S. Yi, and Z. Liu. Variational relational point completion network. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 8524–8533, 2021. 2

[26] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017. 2, 6, 7

[27] C. R. Qi, H. Su, M. Niebner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2016. 2

[28] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 6

[29] P. Shaw, J. Uszkoreit, and A. Vaswani. Self-attention with relative position representations. *NAACL*, 2018. 2

[30] Q. Shi, S. Anwar, and N. Barnes. Semantic segmentation for real point cloud scenes via bilateral augmentation and adaptive fusion. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 2

[31] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. 2

[32] H. Thomas, C. R. Qi, J. E. Deschaud, B. Marcotegui, and L. J. Guibas. Kpconv: Flexible and deformable convolution for point clouds. *ICCV*, 2019. 6, 7

[33] A. Vaswani, P. Ramachandran, A. Srinivas, N. Parmar, B. Hechtman, and J. Shlens. Scaling local self-attention for parameter efficient visual backbones. In *CVPR*, pages 12894–12904, 2021. 1, 5

[34] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 1, 2, 3

[35] L. Wang, Y. Huang, Y. Hou, S. Zhang, and J. Shan. Graph attention convolution for point cloud semantic segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. 2

[36] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma. Linformer: Self-attention with linear complexity, 2020. 8

[37] W. Wang, E. Xie, X. Li, D. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 2, 5

[38] W. Wang, L. Yao, L. Chen, D. Cai, X. He, and W. Liu. Crossformer: A versatile vision transformer based on cross-scale attention. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 1, 2

[39] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics*, 38(5), 2018. 2, 6, 7

[40] Z. Wang and F. Lu. Voxsegnet: Volumetric cnns for semantic part segmentation of 3d shapes. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–1, 2019. 2

[41] K. Wu, H. Peng, M. Chen, J. Fu, and H. Chao. Rethinking and improving relative position encoding for vision transformer. *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 5

[42] Z. Wu, S. Song, A. Khosla, X. Tang, and J. Xiao. 3d shapenets for 2.5d object recognition and next-best-view prediction. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2014. 6

[43] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 6

[44] T. Xiang, C. Zhang, Y. Song, J. Yu, and W. Cai. Walk in the cloud: Learning curves for point clouds shape analysis. *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 2

[45] Y. Xu, T. Fan, M. Xu, Z. Long, and Q. Yu. Spidercnn: Deep learning on point sets with parameterized convolutional filters. *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 6

[46] X. Yan, C. Zheng, Z. Li, S. Wang, and S. Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020. 6, 7

[47] B. Yang, J. Wang, R. Clark, Q. Hu, S. Wang, A. Markham, and N. Trigoni. Learning object bounding boxes for 3d instance segmentation on point clouds. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. 2

[48] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019. 2

[49] X. Yu, Y. Rao, Z. Wang, Z. Liu, J. Lu, and J. Zhou. Pointr: Diverse point cloud completion with geometry-aware transformers. *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 2

[50] C. Zhang, H. Chen, H. Wan, P. Yang, and Z. Wu. Graphpbn: Graph-based parallel branch network for efficient point cloud learning. *Graphical Models*, page 101120, 2021. 2

[51] C. Zhang, H. Wan, S. Liu, X. Shen, and Z. Wu. Pvt: Point-voxel transformer for 3d deep learning. *arXiv: 2108.06076 [cs]*, 2021. 2

[52] F. Zhang, Y. Chen, Z. Li, Z. Hong, J. Liu, F. Ma, J. Han, and E. Ding. Acfnet: Attentional class feature network for semantic segmentation. *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 1

[53] K. Zhang, M. Hao, J. Wang, C. D. Silva, and C. Fu. Linked dynamic graph cnn: Learning on point cloud via linking hierarchical features. *arXiv:1904.10014 [cs]*, 2019. 2, 6

[54] H. Zhao, L. Jiang, C. W. Fu, and J. Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. 2

[55] H. Zhao, L. Jiang, C.-W. Fu, and J. Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5565–5573, 2019. 6

[56] H. Zhao, L. Jiang, J. Jia, P. Torr, and V. Koltun. Point transformer. *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 1, 2, 6, 7

[57] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4490–4499, 2018. 2