SCIENCE CHINA
Information Sciences

• **RESEARCH PAPER** •

On Learning the Right Attention Point for Feature Enhancement

Liqiang LIN[1], Pengdi HUANG[1], Chi-Wing FU[2], Kai XU[3], Hao ZHANG[4] & Hui HUANG[1*]

[1]*Shenzhen University, Shenzhen* 518060*, China;*
[2]*The Chinese University of Hong Kong, Hong Kong* 999077*, China;*
[3]*National University of Defense Technology, Changsha* 410073*, China;*
[4]*Simon Fraser University, Burnaby* V5A1S6*, Canada*

Abstract We present a novel attention-based mechanism to learn enhanced point features for point cloud processing tasks, e.g., classification and segmentation. Unlike prior works, which were trained to optimize the weights of a pre-selected set of attention points, our approach learns to *locate* the best attention points to maximize the performance of a specific task, e.g., point cloud classification. Importantly, we advocate the use of *single* attention point to facilitate semantic understanding in point feature learning. Specifically, we formulate a new and simple convolution, which combines convolutional features from an input point and its corresponding *learned attention point*, or LAP, for short. Our attention mechanism can be easily incorporated into state-of-the-art point cloud classification and segmentation networks. Extensive experiments on common benchmarks such as Model-Net40, ShapeNetPart, and S3DIS all demonstrate that our LAP-enabled networks consistently outperform the respective original networks, as well as other competitive alternatives, which employ multiple attention points, either pre-selected or learned under our LAP framework.

Keywords Point Convolution, Feature Enhancement, Attention Point, Deep Neural Network

Citation Liqiang Lin, Pengdi Huang, Chi-Wing Fu, Kai Xu, Hao Zhang, and Hui Huang. On Learning the Right Attention Point for Feature Enhancement. Sci China Inf Sci, for review

1 Introduction

Learning point features is one of the most fundamental problems in 3D vision and a key building block for tasks such as shape classification and segmentation. Conventional convolution employs fixed kernels of varying sizes to aggregate point features, while extensions to variable neighborhoods, which account for anisotropy [1] and other local shape properties, have also been studied. Another line of approaches follows the non-local means idea [2] by collecting features at points that are similar to one another.

Recently, the use of *selective attention* [3–7] has gained much success in computer vision. In a typical setting, an attentional network computes the feature of a point by pre-selecting [8–10] a set of nearest points in the point's neighborhood and learning the associated attention weights to capture additional contextual information through the weights to enrich the point feature. Another recent approach resorts to finding non-local neighbors [11] by considering points within a much larger neighborhood.

In this paper, we present a new model for attentional point feature learning, which learns to *locate a single attention point for feature enhancement to optimize the performance of a specific point cloud processing task such as classification and segmentation.* For example, to incorporate our model into a classification network, e.g., DGCNN [9], that is based on point-wise convolutional feature processing and aggregation, we can readily replace each convolution layer in the network (i.e., EdgeConv in the case of DGCNN) with a new convolution, which *combines* or *fuses* the convolutional feature at an input point with the convolutional feature at its corresponding *learned attention point* (LAP). Hence, while only one attention point is selected by our method, the feature fusion captures the information from two local point *neighborhoods.* The LAP is obtained by adding to the original point an offset vector, as illustrated

* Corresponding author (email: hhzhiyan@gmail.com)

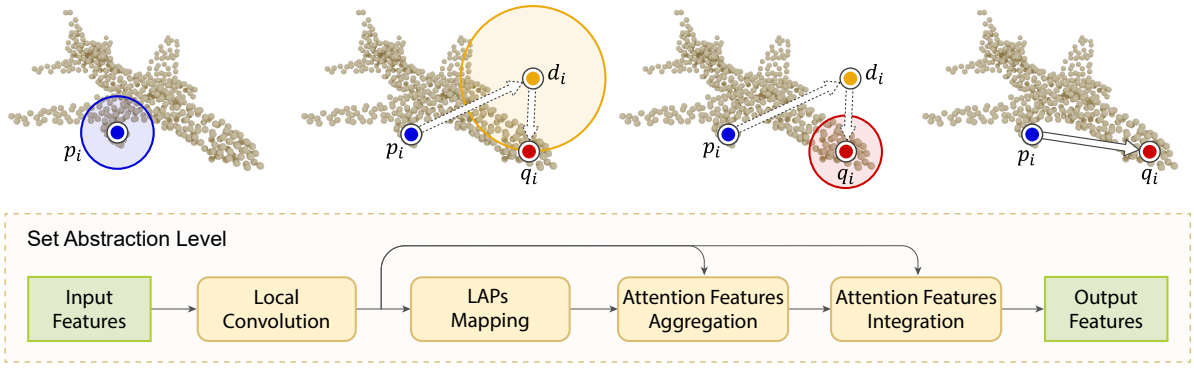
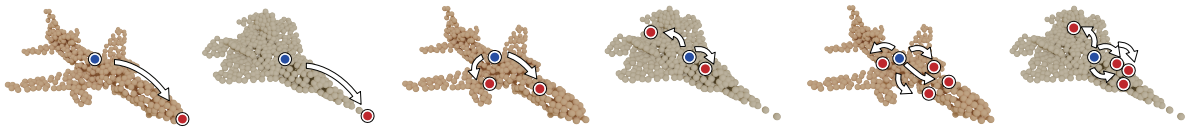


Figure 1 Learning to locate the best attention point q_i (red) for a given point p_i (blue). The feature f_i at p_i is updated using arbitrary local convolution. After that, we take f_i to learn an offset vector to obtain d_i (yellow), from which we locate the target attention point q_i. We then aggregate the neighboring features of q_i, which are finally fused into p_i's features.



(a) 1 LAP (b) 2 LAPs (c) 4 LAPs

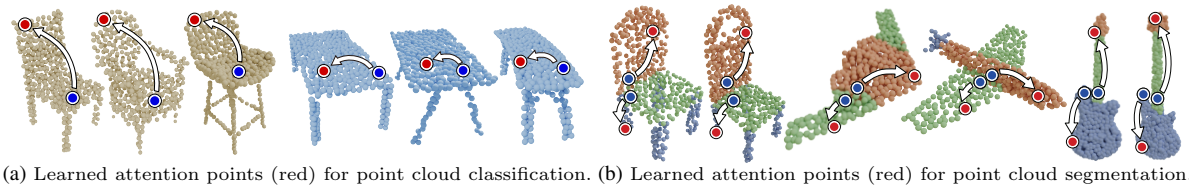
Figure 2 Visualizing *different number* of LAPs (red dots) selected by the same learning framework, for the same original points (blue dots) on two airplanes.More LAPs, two or four in (b) and (c), respectively, tend to exhibit less semantic consistency, compared to the single best LAP (a).

in Figure 1. These offset vectors are learned by minimizing the classification errors over the training point clouds.

Unlike prior attentional models, which were trained to optimize the weights of a pre-selected *set* of attention points [10, 11] or a *sequence* via a recurrent network [12], we focus on learning to *locate* one best LAP. A key premise of our work is that if the right point is found, then adding more attention may not be more effective. Our intuition is that as shape geometries and structures vary, e.g., within the same class of shapes for classification or within the same semantic part in the context of segmentation, it is easier to attain *semantic consistency* with one LAP than with more, as illustrated in Figure 2 for a comparison between one, two, and four LAPs. Such a consistency implies that matching points on two similar shapes (e.g., the two blue points on the two airplanes in each column of Figure 2) tend to be mapped to matching LAPs (the corresponding red points).

As shown in Figure 3, the attention points learned by our method do appear to exhibit a certain level of semantic understanding. Since our training is *task-dependent*,the LAPs tend to behave differently for different tasks. In particular, they may not possess similar features as the original points. The highlighted blue dots in Figure 3(a) are located in similar geometry: seat surface and desktop surface. For chairs, the LAPs (the highlighted red dots) are mapped to the chair backs, but there are no backs for desks. So the features of the blue dots can be complemented by the features of LAPs. This works similarly for a segmentation task. The highlighted blue dots in Figure 3(b) are located in the border between different semantic parts and thus share similar features. It is hard to discriminate them apart from a segmentation network. But with the different semantic features of the LAPs, they can be segmented in a more correct manner. A critical observation here is that for feature enhancement, points with dissimilar features can be equally useful since they can provide *complementary* information.

Our LAP-based attentional point feature learning mechanism can be easily incorporated into modern classification and segmentation networks, such as PointNet++ [13], RSCNN [14], DGCNN [9], and KPConv [15]. We show through comprehensive tests that, on common benchmarks ModelNet40, ShapeNetPart, and S3DIS, our LAP-enabled networks *consistently* improve the performance over the respective original networks, as well as other competitive alternatives, which employ multiple attention points, either pre-selected, such as PointASNL [11] and PointWeb [10], or learned under our LAP framework. The consistent improvements are verified over varying point neighborhood sizes, train-test splits, and ways for feature integration.



(a) Learned attention points (red) for point cloud classification. (b) Learned attention points (red) for point cloud segmentation.

Figure 3 *Learned attention points* (LAPs) by our method, shown in red, corresponding to several original (blue) points on two tasks. For classification (a), the LAPs exhibit *within-class semantic consistency*, while helping to discriminate between different classes: attention points for similar original points on chair seats and tabletops are located on different semantic parts. For segmentation (b), the two blue points on each shape belong to different semantic regions, but are spatially close, thus sharing similar spatial neighborhoods. However, their LAPs are located *far apart* on different semantic parts, effectively "pushing" the blue points into different segments by making their features *dissimilar*.

2 Related Work

Deep learning on 3D point clouds. Inspired by the seminal work of PointNet [16], many deep neural networks have been developed to directly operate on points [17]. To overcome the inadequacy of PointNet on capturing local structures, PointNet++ [13] adopts a deep hierarchical feature learning mechanism that recursively employs PointNet to process point neighborhoods of increasing sizes. Later, PCNN [18] adapts image-based CNNs to the point cloud setting, leading to a permutation-invariant point feature learning. PointCNN [19] learns a transformation matrix to weight the input features associated with the input points and rearrange the points into a canonical permutation. A recent survey on this topic is available in [20].

One recent work of relevance is S-NET [21], a learning-based point cloud downsampling network that is also optimized for a particular task (e.g., classification or retrieval), like our work. Also, both their sampling problem and our proposed approach to feature enhancement would come down to learning to locate a small number of points that contribute most to a task at hand. While the success of their work can be seen as validation to our task-specific training, the problem setup and model design for attention point selection are completely different from S-NET.

Point feature learning. Point features play an important role to task performance. So, a majority of deep models on 3D point clouds focus on designing methods to extract better features, typically by exploiting a point's local neighborhood. DGCNN [9] bases the neighboring relations of points on their distances in the feature space and aggregates pair-wise features to generate features for the center point. PointWeb [10] densely connects every pair of points in a local neighborhood, aiming at extracting point feature that better represents the local region around the center point. Ran et al. [22] explore the possibilities of local relation operators for point clouds.

Other methods focus on designing efficient convolution kernels [23,24] on points. PointConv [25] treats the convolution kernel as a Monte Carlo estimate of nonlinear functions of the local 3D coordinates of points. Point features are weighted by the estimated density. KPConv [15] uses a set of points in Euclidean space as the convolution kernel and aggregates input features based on the distances between kernel points and input points. PAConv [26] introduces a generic convolution operation for 3D point cloud processing. While prior works focus mostly on learning local features, some recent ones [11, 27] start to explore non-local 3D features with the attention mechanism [3].

Attentional point feature learning. Attention-based feature learning [3] has been introduced to learning point features soon after its applications to image features. Rather than being exhaustive, we discuss methods that focus on point feature learning rather than on specific vision tasks. Yang et al. [28] propose the point attention transformer, which leverages a parameter-efficient group shuffle attention to learn the point relations. Zhang et al. [29] propose Point Contextual Attention Network to predict the significance of each local point feature based on the point context. Chen et al. [30] learn local geometric representations by embedding a graph attention mechanism. Zhao et al. [31] design self-attention layers and adapt transformer for point clouds.

Further, attention can be used to learn long-range global features. Liu et al. [12] propose Point2Sequence, an RNN-based model that captures correlations between different areas in a point cloud. Lu et al. [32] design the spatial-channel attention module to capture multi-scale and global context features. Han et al. [33] create a global graph and weights features of distant points with attention. More recently, Cheng

et al. [27] propose global-level blocks to update the feature of a superpoint using weighted features of other superpoints, while Yan et al. [11] propose PointASNL that samples points over the whole point cloud to query similar ones for non-local point feature learning.

All the methods mentioned above *pre-select a set of points*, usually points in the local neighborhood or points with similar features, as the attentional points. In the 2D image domain, Zhang et al. [34] propose to learn an affinity matrix for each attention point to shift it to a better location. The features of the attention points are weighted based on feature similarity. On the other hand, Xue et al. [35] suggest that not all attention is needed, and only a small part of the inputs is related to the output targets, through their investigations into several natural language processing tasks. Our work is inspired by these works which employ learned attentional processing. Yet, we go beyond them and propose to learn to locate the point to attend to, over the entire point cloud, without relying on feature similarity.

3 Method

Recent deep learning approaches for 3D point clouds often focus on designing operators for better local feature extraction. Denoting $f_i \in \mathbb{R}^{C_1}$ as the input feature of point p_i in a certain network layer (C_1 is the channel number), this local operator takes the input features of p_i's neighbors and aggregates them to form the output feature $f_i' \in \mathbb{R}^{C_2}$ of p_i,

$$f_i' = LocalConv(\mathcal{N}(p_i)), \tag{1}$$

where $\mathcal{N}(p_i)$ is the set of neighboring points of p_i, which are often found by a ball query or k-nearest neighbor (KNN) search. Here, *LocalConv* denotes a local convolution. It can be a point-wise local operator [13, 16], a grid kernel local operator [15], or an attention-based local operator [10].

Some methods use attention to learn global features for long-range structures, in which $\mathcal{N}(p_i)$ is replaced by a set of pre-selected attention points. These points can be found by locating points with similar features as f_i in the feature space. Like Eq. (1), a basic mechanism is to weigh the input features of the attention points based on the feature similarity, then to combine them into the output feature of p_i.

In this work, we propose a new network module, called LAP-Conv, that *directly learns to find* attention points without relying on the feature similarity. In particular, we learn to find for each input point p_i *only one* attention point instead of finding multiple ones. The top part of Figure 1 illustrates how LAP-Conv works. Let P be the input cloud with n points $\{p_1, p_2, ..., p_n\}$. Taking as an example the blue point in Figure 1 as p_i, we learn to find its associated attention point q_i, the red point, through the assistance of the offset point d_i, the yellow point.

The bottom part of Figure 1 shows the overall feature learning pipeline. First, we update feature f_i of point p_i using a Local Convolution block (Eq. (1)), then map f_i to the corresponding attention point q_i using the LAPs Mapping block (Section 3.1). Next, the Attention Features Aggregation block (Section 3.2) aggregates features for the attention point q_i. Finally, the Attention Features Integration block (Section 3.3) integrates the feature of the attention point into the output feature of p_i. Also, we show the detailed structure of LAP-Conv in Figure 4, in which the three blocks are outlined.

3.1 Learning to Locate Attention Points

The features of points in the point cloud are first updated with Eq. (1). We learn a function D to map the input feature of point p_i to locate offset point d_i (the yellow point in Figure 1) with $d_i = D(f_i)$, and then find the target attention point q_i (the red point) with the assistance of point d_i. Inspired by the way how neighboring points are found in the feature space [9], we propose to learn to find the attention points either in the Euclidean space or in the feature space.

Case (i). For attention points in the Euclidean space, we use a small MLP to map the feature f_i to a three-dimensional offset vector in the Euclidean space. We then add this vector back to the 3D coordinates x_i of the original point p_i to locate the offset point

$$d_i = MLP(f_i) + x_i. \tag{2}$$

This MLP is shared among all the points in the input point cloud P. Also, point d_i is not necessarily a point in P. Its location is arbitrary in the Euclidean space. Hence, we map the feature of point p_i

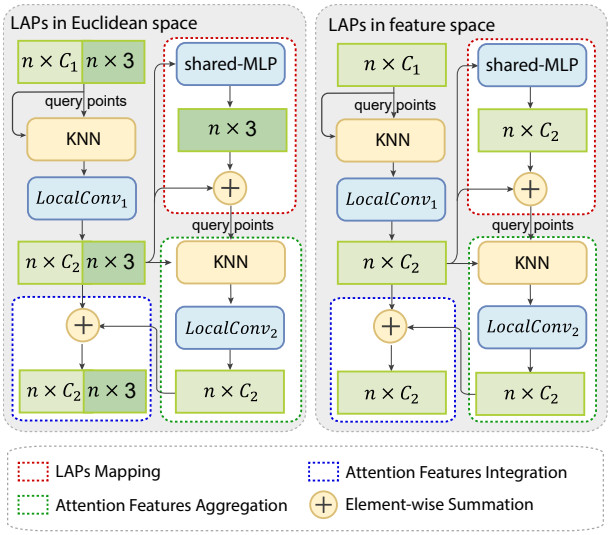


Figure 4 The structure of LAP-Conv. For each attention point in the Euclidean space, we learn a 3-dimensional offset vector and add it back. Each resulting point is used as a query point to find the k nearest points in the point cloud. The features of these points are aggregated into the learned attention point and the feature of each attention point is integrated into the original feature. For attention points in the feature space, the offset vectors are in the same space and the neighboring points are also found in the feature space.

to an offset vector, which acts as an attention direction for further pinpointing d_i. Taking point d_i as guidance, we then find the nearest point q_i to point d_i among all the points in the entire point cloud as the learned attention point q_i associated with p_i.

Case (ii). For attention points in the feature space, the feature f_i of point p_i is mapped via a small shared MLP to an offset vector (with same channel number as f_i) in the feature space. This vector is then added to f_i to produce f_{d_i}, which is an offset feature in the feature space:

$$f_{d_i} = MLP(f_i) + f_i. \tag{3}$$

Similar to d_i in case (i), offset feature f_{d_i} is not necessarily a feature vector of the points in the original point cloud. Also, it can have arbitrary values. With f_{d_i}, we then search the entire point cloud for point q_i with the nearest feature vector f_{q_i} to f_{d_i} in the feature space, and take q_i as the corresponding learned attention point for point p_i. The blocks with the red-dotted frames in Figure 4 illustrate the detailed procedure of this LAPs Mapping.

3.2 Attentional Features Aggregation

After finding the target attention point q_i for point p_i, we then aggregate the features of the neighborhood of q_i and update the feature of point q_i with

$$f_{q_i} = LocalConv(\mathcal{N}(q_i)). \tag{4}$$

For attention points in the Euclidean space, these neighboring points are found in the Euclidean space. For attention points in the feature space, these neighboring points are also found in the feature space.

To find the closest point q_i to d_i, we need to calculate the distances between point d_i and all the other points. To further find the neighboring points of q_i, another distance calculation is required.

To reduce the cost of processing two sets of distance calculations, we directly find the neighboring points of point d_i instead of explicitly finding point q_i and then its neighbors.

The learned attention point q_i must be among the neighboring points of d_i, since q_i is the nearest one to d_i in the original point cloud.

So, the feature of the learned attention point can be updated using the features of the neighboring points of d_i as

$$f_{q_i} = LocalConv(\mathcal{N}(d_i)). \tag{5}$$

For both cases (i) and (ii), we prefer to find these neighboring points using KNN. In this situation, no matter how large the offset vectors are, there are always neighboring points of d_i. If the neighboring points are found with a ball query [13], an additional loss is needed to punish the point d_i from being shifted too far away. Otherwise, there could be no neighboring points of d_i within a certain radius.

Hence, we use KNN to finding the neighboring points for performing $LocalConv_1$ for consistency. After we learn the offset points, we use them as query points to find k neighboring points and update the feature of q_i, as shown in the green dotted frames in Figure 4.

It should be noticed that all these neighboring points are points from the original point cloud (case (i)) or the features of the points in the original point cloud (case (ii)), in which d_i or f_{d_i} is not one of them.

3.3 Integration of Attentional Features

After we aggregate a feature for the learned attention point q_i, we then integrate the feature into the feature of p_i with an integration function I ('I' for integration):

$$f_i' = I(LocalConv_1(\mathcal{N}(p_i)), LocalConv_2(\mathcal{N}(d_i))), \tag{6}$$

where we could use the same or different convolutional operators for $LocalConv_1$ and $LocalConv_2$.

In the previous attention-based networks for processing point clouds, e.g., [9–11, 27, 28, 33, 36, 37], the features of the attention points are weighted by a dot-product similarity between the features of the pre-selected attention points and p_i, or by the similarity in the spatial location between them. An attention point with a more similar feature is given a larger weight when added into the feature of p_i. However, we argue that *not only* points with similar features could be useful, *but also* points with *different features* could also be useful in feature learning. Particularly, points with different features could supply the center point p_i with *vital context for performing the target task.*

Here, one straightforward choice for I is to simply add the features of point q_i and point p_i together:

$$I = Add(f_{p_i}, f_{q_i}), \tag{7}$$

as illustrated in the blue dotted frames in Figure 4. Since $LocalConv_1$ and $LocalConv_2$ do not share parameters, the network can learn to update the feature of q_i for adding it into the feature of p_i. Another choice for I is to concatenate the two features then use an MLP to reduce the dimensions:

$$I = MLP(Concatenate(f_{p_i}, f_{q_i})). \tag{8}$$

A comparison result between these two forms of feature integration can be found in Table 5 of Section 4.

4 Results and Evaluation

Our proposed LAP-Conv can be easily adapted into conventional point-wise feature learning networks.

In existing networks, a *LocalConv* block is generally used in a set abstraction level to update the input feature f_i of a point and produce f_i', as described in Eq. (1).

Hence, by keeping the channel sizes of the input and output features to be the same, we can replace the *LocalConv* block in existing networks with our LAP-Conv to boost the network performance.

To demonstrate the effectiveness of our LAP-Conv, we adapt it into common networks, including PointNet++ [13], RSCNN [14], DGCNN [9], and KPConv [15], producing LAP-enabled networks, i.e., LAP-PointNet++, LAP-RSCNN, LAP-DGCNN, and LAP-KPConv.

Note that since DGCNN finds neighboring points in feature space, the attention points for LAP-DGCNN are learned in the feature space (Eq. (2)). For the other LAP-enabled networks, the LAPs are learned in Euclidean space (Eq. (3)).

We conduct experiments using these networks on various tasks, including shape classification (Section 4.1), part segmentation, and semantic segmentation (Section 4.2), showing both quantitative comparisons and qualitative results. At last, we evaluate various aspects of our LAP-Conv (Section 4.3), e.g., one vs. multiple learned attention points, neighborhood sizes, train/test split, etc.

Table 1 3D Shape classification results on ModelNet40. OA: overall accuracy (%); xyz: points only as input; xyz+nor: use points and normals.

Method	input	#points	OA
PointNet [13]	xyz	1k	89.2
PAT [28]	xyz+nor	1k	91.7
PointCNN [19]	xyz	1k	92.2
PointWeb [10]	xyz+nor	1k	92.3
PointConv [25]	xyz+nor	1k	92.5
A-CNN [24]	xyz	1k	92.6
Point2Node [33]	xyz	1k	93.0
PointASNL [11]	xyz	1k	92.9
PointASNL [11]	xyz+nor	1k	93.2
DensePoint [38]	xyz	1k	93.2
PointTransformer [31]	xyz	1k	93.7
GDANet [22]	xyz	1k	93.8
PAConv [26]	xyz	1k	93.9
PointNet++ [13]	xyz	1k	90.7
RSCNN [14]	xyz	1k	91.7
KPConv [15]	xyz	7k	91.9
DGCNN [9]	xyz	1k	92.9
SimpleView-PointNet++ [39]	xyz	1k	91.2
SimpleView-RSCNN [39]	xyz	1k	93.3
SimpleView-DGCNN [39]	xyz	1k	93.9
LAP-PointNet++	xyz	1k	92.9
LAP-RSCNN	xyz	1k	92.8
LAP-KPConv	xyz	7k	92.3
LAP-DGCNN	xyz	1k	**93.9**

4.1 Classification

First, we evaluate our method on the shape classification task using ModelNet40 [40], which provides a comprehensive collection of 12,311 CAD models from 40 categories. Here, we use the input point clouds sampled from these models by PointNet [16]. Typically, 1,024 points are uniformly sampled per model to serve as the network input.

Following the official train/test split, we use 9,843 models for training and 2,468 models for testing.

In this experiment, we evaluate the performance of LAP-PointNet++, LAP-RSCNN, LAP-KPConv, and LAP-DGCNN, where data preprocessing, augmentation, and the choice of hyper-parameters all follow the original network settings. Also, we employ only 3D point coordinates as network inputs.

For LAP-PointNet++, the training epochs are 500, the batch size is 24, and the learning rate is 0.001. For LAP-RSCNN, since only the single-scale version of the RSCNN code is officially released, we develop our LAP-RSCNN based on the released code and report 91.7 for the single-scale RSCNN, following [41]. The training epochs are 120, the batch size is 32, and the learning rate is 0.001.

In this paper, we report 91.9 for KPConv as the result reproduced by the officially released code. The training epochs are 400, the batch size is 16, and the learning rate is 0.001 for LAP-KPConv.

For LAP-DGCNN, the training epochs are 500, the batch size is 32, and the learning rate is 0.001.

Table 1 reports the results of using different methods for the shape classification task. Comparing the results of the three LAP-enabled networks with those of the original counterparts, we can see that all of them improve the performance over the original networks, showing that our LAP-Conv can help improve the quality of the features for the shape classification task.

Overall, LAP-DGCNN attains a higher performance with an overall accuracy of 93.9 on the classification task compared with other attention-based methods such as PointWeb [10], Point2Node [33], PointASNL [11], and PointTransformer [31].

4.2 Segmentation

Part segmentation. The ShapeNet Part dataset [43] has 16,881 shapes over 16 categories and with 50 part labels. We sample 2,048 points per shape from this dataset for our experiments and follow the official

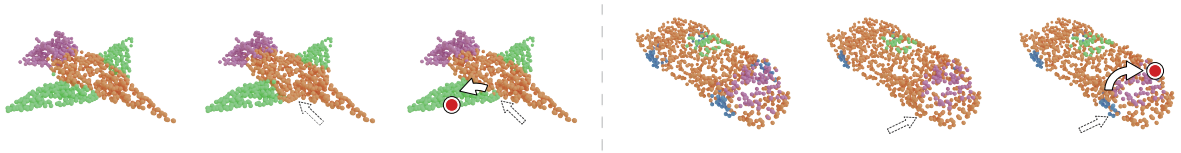


Figure 5 Comparing part segmentation on ShapeNet Part between PointNet++ and LAP-PointNet++. In each case, we present input point cloud with ground true segmentation labels (left), segmentation results predicted by PointNet++ (middle), and segmentation results predicted by LAP-PointNet++ (right), respectively. We highlight an incorrectly labeled point near a segmentation boundary (*hollow* arrow) by PointNet++ (middle) and how the use of LAP-Conv (right) corrected the label with feature enhancement by an attention point (large red dot).

Table 2 Part segmentation results on ShapeNet Part.

Method	mIoU
Kd-Net [42]	82.3
PointNet [16]	83.7
PCNN [18]	85.1
PointTransformer [31]	86.6
PointNet++ [13]	85.1
DGCNN [9]	85.2
RSCNN [14]	86.2
KPConv [15]	86.4
LAP-PointNet++	85.4
LAP-DGCNN	85.8
LAP-RSCNN	86.4
LAP-KPConv	**86.9**

Table 3 Semantic segmentation results on the S3DIS dataset (evaluated on Area 5).

Method	mAcc	mIoU
PointNet [16]	49.0	41.1
TangentConv [45]	62.2	52.6
PointCNN [19]	63.9	57.3
SPGraph [46]	66.5	58.0
PointWeb [10]	66.6	60.3
HPEIN [47]	68.3	61.9
Point2Node [33]	70.0	63.0
MinkowskiNet [48]	71.7	65.4
BAAF-Net [49]	-	65.4
PAConv [26]	73.0	66.6
JSENet [50]	-	67.7
PointTransformer [31]	76.5	**70.4**
DGCNN [9]	59.1	53.1
KPConv [15]	72.8	67.1
LAP-DGCNN	62.3	53.6
LAP-KPConv	73.4	68.2

data split setup [44] adopted in DGCNN [9] and PointNet++ [13]. The batch size for LAP-PointNet++ is 16, the training epochs for LAP-RSCNN are 200, and the learning rate for LAP-KPConv is 0.01. Other parameters stay the same as on ModelNet40.

Table 2 reports the performance by various methods, in terms of the mean Intersection-over-Union (mIoU) metric, showing that LAP-Conv helps improve the part segmentation performance of PointNet++, RSCNN, DGCNN, and KPConv. Figure 5 visually contrasts PointNet++ and LAP-PointNet++ to show the differences made by LAP-Conv.

Indoor scene semantic segmentation. We use the Stanford 3D Large-Scale Indoor Spaces (S3DIS) dataset [51], which contains large-scale 3D-scanned point clouds for six indoor areas with 272 rooms from three different buildings. Altogether, the dataset has $\sim$ 273 million points, each belonging to one of 13 semantic categories. Area-5 is used as the test scene for evaluating the method's generalizability.

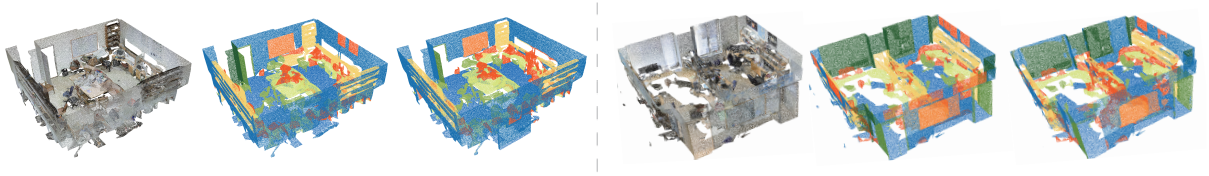


Figure 6 Visualization of two semantic segmentation results on S3DIS. In each case, we present input point cloud with RGB colors (left), points with ground-truth semantic labels (middle), and segmentation results predicted by LAP-DGCNN (right), respectively.

For LAP-DGCNN, each room is split into blocks of size $1m \times 1m$ as done in DGCNN. The input is point coordinates together with RGB colors and normalized spatial coordinates (a 9D vector per point), following the settings in DGCNN [9]. During the training, we sample 4,096 points from each room block. Then, all sampled points are used for testing. The training epochs are 500, the batch size is 10, and the learning rate is 0.01.

For LAP-KPConv, each 3D scene in the dataset is segmented into small sub-clouds contained in spheres. During the training, the spheres were picked randomly in the scenes, and during testing, they are picked regularly in the point clouds. The training epochs are 100, the batch size is 32, and the learning rate is 0.01.

Table 3 gives the results, showing that LAP-Conv helps improve semantic segmentation performance on S3DIS for both DGCNN and KPConv.

Also, we present two visual semantic segmentation results on S3DIS in Figure 6.

Note that since PointNet++ and RSCNN did not report their results on S3DIS, LAP-PointNet++ and LAP-RSCNN are not evaluated on S3DIS.

4.3 Ablation Study

Finally, we evaluate various aspects of our method, employing LAP-DGCNN to test on ModelNet40.

Different ways of choosing LAPs. In this experiment, we compare LAP-DGCNN with three variants of the network in terms of how to choose attention points:

- Random-LAP-DGCNN, in which we replace the MLP for learning to generate the offsets (Eq. (3)) with a randomly-generated vector, thus replacing the core of our LAP-Conv, which uses the learned offset vector to locate the learned attention point;
- LAP2-DGCNN and LAP4-DGCNN, which respectively learns two and four attention points simultaneously, instead of one, in one single LAP-Conv module.

As comparison results in Table 4 show, random-LAP-DGCNN hardly makes any improvement over the baseline, which is DGCNN. The use of random attention points makes the network converge much slower than that of the baseline DGCNN. However, after adequate training epochs, the weights for those random attention points turn out to be very small numbers (most of them are zeros). Under this circumstance, LAP-DGCNN with random attention points degenerates into the DGCNN and thus leads to similar results.

This result demonstrates the importance of the learned offset vectors for searching for the target attention point.

Yet, using just the right learned attention point, as in LAP-DGCNN, we can achieve a large performance improvement over the original DGCNN.

Next, comparing LAP2-DGCNN and LAP4-DGCNN with LAP-DGCNN, we can see that learning to find and use more than one attention point improves over DGCNN. However, the improvement is far below than using just one single learned attention point. Yet, an interesting observation is that the more learned attention points we use, resulting in larger network capacity, the worse the results appear to be, indicating that LAP's effectiveness is not merely brought by more convolution layers.

The visualization in Figure 2 on where the 1, 2, and 4 attention points were selected by the networks may hint at a possible reason: as more points are added, their consistency tends to drop.

As shown in Table 4, LAP4-DGCNN have 4.49M parameters and DGCNN have 4.21M parameters. Memory consumption has the same ratio as the amount of parameters. The runtime is 1.8 vs. 2.1 ($10^{-2}/batch$).

Different feature integration. We can have different ways of integrating the feature of point p_i with that of its LAP q_i; see Section 3.3.

Table 4 Comparing different ways of choosing LAPs.

	OA	mAcc	Parameters (M)
DGCNN (baseline)	92.9	90.2	4.21
random-LAP-DGCNN	93.0	89.9	4.25
LAP-DGCNN	**93.9**	**91.5**	4.28
LAP2-DGCNN	93.3	90.2	4.35
LAP4-DGCNN	93.1	90.3	4.49

Table 5 Comparing different ways of integrating features.

Different Function I	OA	mAcc
I_1 or Eq. (7)	93.9	91.5
I_2 or Eq. (8)	93.6	91.0

Table 6 Comparing different neighboring sizes (k) in KNN.

k	DGCNN	LAP-DGCNN
5	90.5	**93.5**
10	91.4	**93.4**
20	92.9	**93.9**
40	92.4	**93.3**

Table 7 Shape classification results (OA) on ModelNet40 with different train/test splits.

Train/all	DGCNN	LAP-DGCNN
80%	92.9	**93.9**
60%	91.5	**92.6**
40%	90.6	**92.1**
20%	88.9	**90.2**
10%	87.3	**88.0**
5%	82.7	**83.1**
1%	63.5	**64.9**

As shown in Table 5, option I_1 (Eq. (7)) gives a slightly better performance compared with I_2 (Eq. (8)), so we empirically choose I in our LAP-Conv. We have tested other ways of feature integration, e.g., point-wise multiply, but they do not lead to good results.

Different neighborhood sizes. Note that we built LAP-DGCNN by following the setting of DGCNN to use k=20 for the two KNN operations in LAP-Conv. In this experiment, we test LAP-DGCNN with different neighboring sizes (k), while keeping all the other settings unchanged.

As shown by results in Table 6, the performance of our LAP-DGCNN is quite robust against different choices of k, and it also consistently outperforms DGCNN. Even with a rather small neighborhood size k=5, our method can already attain a rather high performance, which exceeds that of DGCNN for all the neighborhood sizes shown in Table 6.

Different train/test splits. In a final experiment, we test the robustness of our method over different training/test splits. Note that the original setting uses 9,843 models, which is around 80% of the whole dataset, for training, and the remaining 20% for testing.

We test the performance of our LAP-DGCNN using smaller training sets, from 80% down to 1% of the whole dataset, and compare to DGCNN, with results shown in Table 7. Similar to [52], we select the training samples as follows, for all train/test splits: we randomly sample one object per class first; the remaining training data are randomly sampled from the original training set, regardless of their classes. As we can observe from Table 7, our method performs well even with a very small training set, and most importantly, it consistently outperforms DGCNN over all train/test splits. Also, the randomization during our setup of the multiple training sets can alleviate potential concerns of over-fitting.

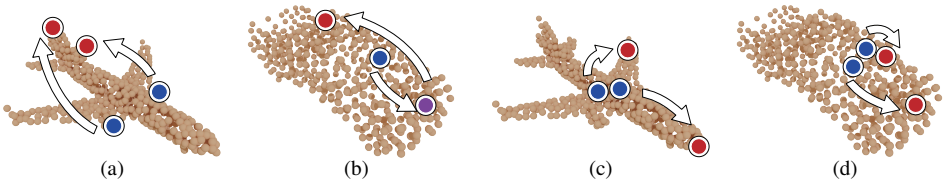


Figure 7 LAP selection learned by our network is neither symmetric (a) nor invertible (b), in general; and it is not always stable (c-d), as nearby points may be mapped afar.

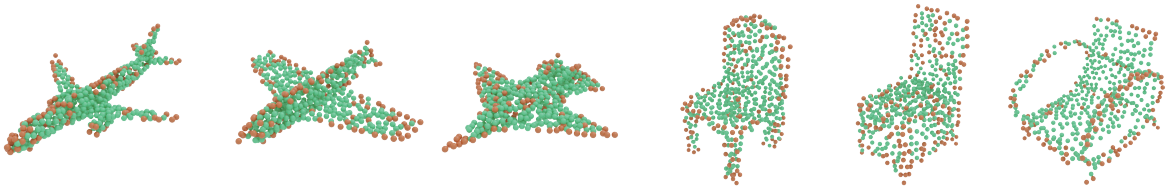


Figure 8 The LAPs are highlighted in orange while the original point clouds are in green.

5 Discussion, Limitation, and Future Work

Our paper is about learning point features, which is an essential step in most point cloud processing tasks including classification, segmentation, and more. Furthermore, we tackle arguably the most fundamental question in this setting, namely, how to find the neighbors of a point P to best characterize its features. With deep learning, many works propose to learn what P's neighbors are and their weights, as discussed in Section 2. Our work improves upon these attentional learning methods and it is motivated by two key observations: a) the learning of attention points should be task-specific; and b) one attention point may be best as it is easier to attain semantic consistency; see Figures 2. The merit of our ideas has been validated through a comprehensive evaluation, demonstrating that our one-LAP approach outperforms the use of multiple attention points, both by previous works (e.g., PointASNL [11], PointWeb [10]), and when they are learned under our framework.

Currently, our LAP selection is *neither symmetric nor invertible* in general. As shown in Figure 7, if two points p and q are symmetric on a shape, $LAP(p)$ and $LAP(q)$ may not be; and $p \neq LAP(LAP(p))$. The figure also shows that the selection may become unstable, as one would expect from the results of a data-driven optimization. However, it is worth noting that in general, we find the learned offset vectors to be more stable than the final attention points, which are obtained by projecting back onto the point clouds. Figure 8 shows the LAPs for each shape. The LAPs are usually mapped to some distinctive and characteristic regions, such as points on the edges or at tips.

On the technical front, our method may not extend to other representations such as voxels. In voxel-based representation, the offset points need to be discretized into voxels which may cause discontinuity in gradients.

Acknowledgements We thank all reviewers for their valuable comments. This work was supported in parts by NSFC (U21B2023, 62161146005, U2001206, 61902254), Guangdong Talent Program (2019JC05X328), Guangdong Science and Technology Program (2020A0505100064), DEGP Key Project (2018KZDXM058, 2020SFKC059), Shenzhen Science and Technology Innovation Program (JCYJ20210324120213036, RCJC20200714114435012), HKSAR (CUHK14206320), NSERC (611370), and Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ).

References

1 Weickert J. *Anisotropic diffusion in image processing*. B. G. Teubner (Stuttgart), 1998.
2 Buades A, Coll B, and Morel J M. A non-local algorithm for image denoising. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, volume 2, pages 60–65. IEEE, 2005.
3 Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. In *Proc. Conf. on Neural Information Processing Systems*, pages 5998–6008, 2017.
4 Liu X, Xia T, Wang J, et al. Fully convolutional attention networks for fine-grained recognition. *arXiv preprint arXiv:1603.06765*, 2016.
5 Wang F, Jiang M, Qian C, et al. Residual attention network for image classification. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, pages 3156–3164, 2017.
6 Peng Y, He X, and Zhao J. Object-part attention model for fine-grained image classification. *IEEE Trans. on Image Processing*, 27(3):1487–1500, 2018.

7 Mnih V, Heess N, Graves A, et al. Recurrent models of visual attention. In *Proc. Conf. on Neural Information Processing Systems*, pages 2204–2212, 2014.
8 Hamilton W, Ying Z, and Leskovec J. Inductive representation learning on large graphs. In *Proc. Conf. on Neural Information Processing Systems*, pages 1024–1034, 2017.
9 Wang Y, Sun Y, Liu Z, et al. Dynamic graph CNN for learning on point clouds. *ACM Trans. on Graphics*, 38(5):146, 2019.
10 Zhao H, Jiang L, Fu C W, et al. PointWeb: Enhancing local neighborhood features for point cloud processing. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, pages 5565–5573, 2019.
11 Yan X, Zheng C, Li Z, et al. PointASNL: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, pages 5589–5598, 2020.
12 Liu X, Han Z, Liu Y S, et al. Point2Sequence: Learning the shape representation of 3D point clouds with an attention-based sequence to sequence network. In *Proc. AAAI Conf. on Artificial Intelligence*, volume 33, pages 8778–8785, 2019.
13 Qi C R, Yi L, Su H, et al. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Proc. Conf. on Neural Information Processing Systems*, pages 5099–5108, 2017.
14 Liu Y, Fan B, Xiang S, et al. Relation-shape convolutional neural network for point cloud analysis. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, pages 8895–8904, 2019.
15 Thomas H, Qi C R, Deschaud J E, et al. KPConv: Flexible and deformable convolution for point clouds. In *Proc. Int. Conf. on Computer Vision*, pages 6411–6420, 2019.
16 Qi C R, Su H, Mo K, et al. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, pages 652–660, 2017.
17 Riegler G, Osman Ulusoy A, and Geiger A. OctNet: Learning deep 3D representations at high resolutions. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, pages 3577–3586, 2017.
18 Atzmon M, Maron H, and Lipman Y. Point convolutional neural networks by extension operators. *ACM Trans. on Graphics (Proc. SIGGRAPH)*, 37(4):1–12, 2018.
19 Li Y, Bu R, Sun M, et al. PointCNN: Convolution on χ-transformed points. In *Proc. Conf. on Neural Information Processing Systems*, pages 820–830, 2018.
20 Guo Y, Wang H, Hu Q, et al. Deep learning for 3D point clouds: A survey. *IEEE Trans. Pattern Analysis & Machine Intelligence*, 2020.
21 Dovrat O, Lang I, and Avidan S. Learning to sample. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, 2019.
22 Xu M, Zhang J, Zhou Z, et al. Learning geometry-disentangled representation for complementary understanding of 3d object point cloud. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3056–3064, 2021.
23 Boulch A. ConvPoint: Continuous convolutions for point cloud processing. *Computers & Graphics*, 88:24–34, 2020.
24 Komarichev A, Zhong Z, and Hua J. A-CNN: Annularly convolutional neural networks on point clouds. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, pages 7421–7430, 2019.
25 Wu W, Qi Z, and Fuxin L. PointConv: Deep convolutional networks on 3D point clouds. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, pages 9621–9630, 2019.
26 Xu M, Ding R, Zhao H, et al. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3173–3182, 2021.
27 Cheng M, Hui L, Xie J, et al. Cascaded non-local neural network for point cloud semantic segmentation. *arXiv preprint arXiv:2007.15488*, 2020.
28 Yang J, Zhang Q, Ni B, et al. Modeling point clouds with self-attention and Gumbel subset sampling. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, pages 3323–3332, 2019.
29 Zhang W and Xiao C. PCAN: 3D attention map learning using contextual information for point cloud based retrieval. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, pages 12436–12445, 2019.
30 Chen C, Fragonara L Z, and Tsourdos A. GAPNet: Graph attention based point neural network for exploiting local feature of point cloud. *arXiv preprint arXiv:1905.08705*, 2019.
31 Zhao H, Jiang L, Jia J, et al. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021.
32 Lu H, Chen X, Zhang G, et al. SCANet: Spatial-channel attention network for 3D object detection. In *Proc. Int. Conf. on Acoustics, Speech & Signal Processing*, pages 1992–1996. IEEE, 2019.
33 Han W, Wen C, Wang C, et al. Point2Node: Correlation learning of dynamic-node for point cloud feature modeling. In *Proc. AAAI Conf. on Artificial Intelligence*, pages 10925–10932, 2020.
34 Zhang L, Xu D, Arnab A, et al. Dynamic graph message passing networks. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, pages 3726–3735, 2020.
35 Xue L, Li X, and Zhang N L. Not all attention is needed: Gated attention network for sequence data. In *Proc. AAAI Conf. on Artificial Intelligence*, pages 6550–6557, 2020.
36 Xie S, Liu S, Chen Z, et al. Attentional ShapeContextNet for point cloud recognition. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, pages 4606–4615, 2018.
37 Liang Z, Yang M, and Wang C. 3D graph embedding learning with a structure-aware loss function for point cloud semantic instance segmentation. *arXiv preprint arXiv:1902.05247*, 2019.
38 Liu Y, Fan B, Meng G, et al. DensePoint: Learning densely contextual representation for efficient point cloud processing. In *Proc. Int. Conf. on Computer Vision*, pages 5239–5248, 2019.
39 Goyal A, Law H, Liu B, et al. Revisiting point cloud shape classification with a simple and effective baseline. *International Conference on Machine Learning*, 2021.
40 Wu Z, Song S, Khosla A, et al. 3D ShapeNets: A deep representation for volumetric shapes. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, pages 1912–1920, 2015.
41 Li R, Li X, Heng P A, et al. PointAugment: an auto-augmentation framework for point cloud classification. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, pages 6378–6387, 2020.
42 Klokov R and Lempitsky V. Escape from cells: Deep Kd-networks for the recognition of 3D point cloud models. In *Proc. Int. Conf. on Computer Vision*, pages 863–872, 2017.
43 Yi L, Kim V G, Ceylan D, et al. A scalable active framework for region annotation in 3D shape collections. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)*, 35(6):1–12, 2016.
44 Chang A X, Funkhouser T, Guibas L, et al. ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015.

45 Tatarchenko M, Park J, Koltun V, et al. Tangent convolutions for dense prediction in 3D. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, pages 3887–3896, 2018.

46 Landrieu L and Simonovsky M. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, pages 4558–4567, 2018.

47 Jiang L, Zhao H, Liu S, et al. Hierarchical point-edge interaction network for point cloud semantic segmentation. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, pages 10433–10441, 2019.

48 Choy C, Gwak J, and Savarese S. 4D spatio-temporal ConvNets: Minkowski convolutional neural networks. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, pages 3075–3084, 2019.

49 Qiu S, Anwar S, and Barnes N. Semantic segmentation for real point cloud scenes via bilateral augmentation and adaptive fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1757–1767, 2021.

50 Hu Z, Zhen M, Bai X, et al. JSENet: Joint semantic segmentation and edge detection network for 3D point clouds. *Proc. Euro. Conf. on Computer Vision*, 2020.

51 Armeni I, Sener O, Zamir A R, et al. 3D semantic parsing of large-scale indoor spaces. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, pages 1534–1543, 2016.

52 Sauder J and Sievers B. Self-supervised deep learning on point clouds by reconstructing space. In *Proc. Conf. on Neural Information Processing Systems*, pages 12962–12972, 2019.