

# PS<sup>2</sup>-Net: A Locally and Globally Aware Network for Point-Based Semantic Segmentation

Na Zhao Tat-Seng Chua Gim Hee Lee  
National University of Singapore

{nazhao, chuats, leegh}@comp.nus.edu.sg

## Abstract

In this paper, we present the PS<sup>2</sup>-Net - a locally and globally aware deep learning framework for semantic segmentation on 3D scene-level point clouds. In order to deeply incorporate local structures and global context to support 3D scene segmentation, our network is built on four repeatedly stacked encoders, where each encoder has two basic components: EdgeConv that captures local structures and NetVLAD that models global context. Different from existing start-of-the-art methods for point-based scene semantic segmentation that either violate or do not achieve permutation invariance, our PS<sup>2</sup>-Net is designed to be permutation invariant which is an essential property of any deep network used to process unordered point clouds. We further provide theoretical proof to guarantee the permutation invariance property of our network. We perform extensive experiments on two large-scale 3D indoor scene datasets and demonstrate that our PS<sup>2</sup>-Net is able to achieve state-of-the-art performances as compared to existing approaches.

## 1. Introduction

Semantic scene segmentation refers to the process of assigning a class label to each element representation of a scene. The outcome of semantic scene segmentation is extremely useful for many applications in artificial intelligence such as interactions of robots (e.g. self-driving cars and autonomous drones) with its environment, and augmented/virtual reality (AR/VR) etc. Semantic scene segmentation on 2D images, where each pixel is the elementary representation of the scene, is a long-standing problem in computer vision; and many impressive results have been shown with deep learning [3, 6, 9, 16, 20] in recent years. In contrast to its 2D image counterpart, semantic scene segmentation on 3D point cloud, where each point is the elementary representation of the scene, has gained attention in the deep learning/ computer vision community only over last few years. This is largely attributed to

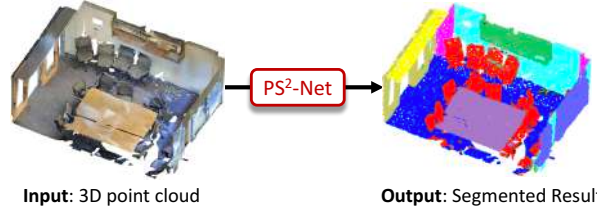


Figure 1. Our PS<sup>2</sup>-Net segments raw point cloud into semantic homogeneous regions.

the non-permutation invariance property of neural networks [18, 29], which work well on 2D image pixels that are arranged in a regular orderly structure, but would fail catastrophically when used on 3D point clouds where the points exist in an unordered and irregular manner.

To circumvent the permutation invariance problem, several existing works attempt to convert the unordered and irregular 3D point cloud into ordered and regular representations that can be directly used in neural networks. Two commonly used approaches include: (a) [4, 11, 14, 22, 23] that project the 3D point cloud onto a set of virtual multi-view image planes, and (b) [5, 7, 17, 24, 27] that quantize the 3D point cloud into regular voxel grids. However, both approaches result in severe loss of information. The former suffers information loss from occlusions and 3D to 2D projections. The latter suffers from quantization error and the high computational cost of 3D convolutions limits its scalability. Recently, [18] pioneered the permutation invariant PointNet that allows deep learning to be directly applied to 3D point clouds. Furthermore, it has shown promising results on both 3D object classification and semantic segmentation tasks. Nonetheless, the design of PointNet, *i.e.* feeding each point individually into several multi-layer perceptrons (MLPs) followed by a maxpooling operation to achieve permutation invariance, inherently prohibits the network from capturing the local information embedded in the neighboring points. However, the local information is essential in modeling fine-grained structures, *e.g.* plane or corner, and convex or concave element. Consequently, several later works [8, 10, 13, 15, 19, 28, 30] proposed to capture the local information by considering the neighboring

points. However, these approaches treat each point in their respective local region independently to achieve permutation invariance, and this prevents the modeling of geometric relationships among neighbor points, which hinders discriminative local feature learning. Moreover, those RNN-based approaches [8, 10, 13, 28] requiring sequential input ordering violate permutation invariance, and [15] inherently is not permutation-invariant. DGCNN [26] is designed to overcome such limitations. It embeds the relationship between a point and its neighbors in the so-called edge features, and uses a channel-wise symmetric aggregation operation (*i.e.* max-pooling) over local neighborhood to capture representative local information. Nonetheless, the global context among different local features is neglected in their work due to a max pooling operation. This limits its ability to encode the semantic information of the entire 3D scene.

In this paper, we propose the PS<sup>2</sup>-Net - an efficient end-to-end framework for Point cloud Semantic Segmentation (PS<sup>2</sup>) that takes local structures and global context into consideration. Our work leverages on EdgeConv [26] to capture local information, and uses the NetVLAD [1] to encode global context. More specifically, PS<sup>2</sup>-Net takes raw point cloud as the input and outputs point level semantic class labels. We design an encoder module that can be stacked repeatedly into a deep network for learning the discriminative representations. Each encoder module consists of two basic components - EdgeConv and NetVLAD. Furthermore, we prove theoretically that the PS<sup>2</sup>-Net guarantees permutation invariance to any order of input points. Our main contributions are summarized as follows:

- We design PS<sup>2</sup>-net - an end-to-end network for semantic scene segmentation on 3D point clouds. Our network is permutation-invariant, and is able to integrate both local structures and global context.
- Our encoder is flexible and can be stacked or recurrently plugged into existing deep learning architectures to exploit the fine-grained local and global properties from point clouds.
- Extensive experimental results on two large-scale 3D indoor datasets show that our PS<sup>2</sup>-net outperforms existing state-of-the-art approaches on the task of semantic scene segmentation on 3D point clouds.

## 2. Related work

In this section, we focus on the literature survey of existing deep learning approaches that directly take point clouds as input for semantic scene segmentation. We omit discussions of existing works that alleviate the permutation invariance problem via conversion of the point clouds into alternative form of representations *e.g.* projections into multi-view virtual images and voxelization.

PointNet [18] pioneered the direct use of 3D points as input in deep learning. It learns point-wise features by passing each point individually through several shared MLPs, followed by a symmetry function (*i.e.* max-pooling) that aggregates over the feature channels into a global feature that represents the point cloud. The operation of the shared MLPs on the individual points and max-pooling made the network permutation invariant. However, the design of PointNet overlooks the exploitation of local structures and prevents the network from learning fine-grained structures. To overcome this limitation, most follow-up works attempt to incorporate the local structures in an efficient way. PointNet++ [19] utilizes the farthest point sampling and ball query to group the point cloud into a set of local regions, and then applies PointNet on each region to capture the local structures. Despite the preservation of permutation invariance and the incorporation of local structures, the grouping operation on the point cloud into independent local regions may impede learning of the global context.

Other variants of PointNet employ various Recurrent Neural Network (RNN) based techniques to learn the contextual information from the local features, which are extracted by PointNet from partitioned local patches based on the spatial locations of the points [8, 10, 28] or geometric homogeneity [13]. Engelmann *et al.* [8] divide a point cloud into a tessellation of ordered blocks. A feature generated from each block using PointNet is then fed sequentially into a Gated Recurrent Unit (GRU). The GRU ensures that information from neighboring blocks are encoded in the final feature. Huang *et al.* [10] propose the RSNET, where the point cloud is sliced into three sets of uniformly-spaced blocks independently along the three coordinate axes (*i.e.* x, y, z). A feature is extracted from each block using a PointNet-like network. Next, an ordered sequence for each set of blocks is defined manually along each axis. Each of the three ordered sequences is respectively fed into a RNN layer to explore the local context. Finally, the updated block features are assigned back to the points in the blocks. Thus, each point-wise feature aggregates its three associated directional block features. In contrast to an independent use of RNNs on the different axes as in RSNET, Ye *et al.* [28] propose a two-direction hierarchical RNN model. Specifically, the first set of RNNs is applied on the blocks along the x-axis where the outputs are used by the second set of RNNs that runs along the y-axis to generate the block features across the two dimensions. However, the RNN-based approaches require sequential ordering in the inputs and this is a violation of the permutation invariance.

More recently, several works [15, 21, 26] are proposed to explore the fine-grained geometric relationships among neighboring points. PointCNN [15] learns a  $K \times K$   $\chi$ -transformation from the  $K$  neighborhoods of the input points. It assumes that each point assigned with a local

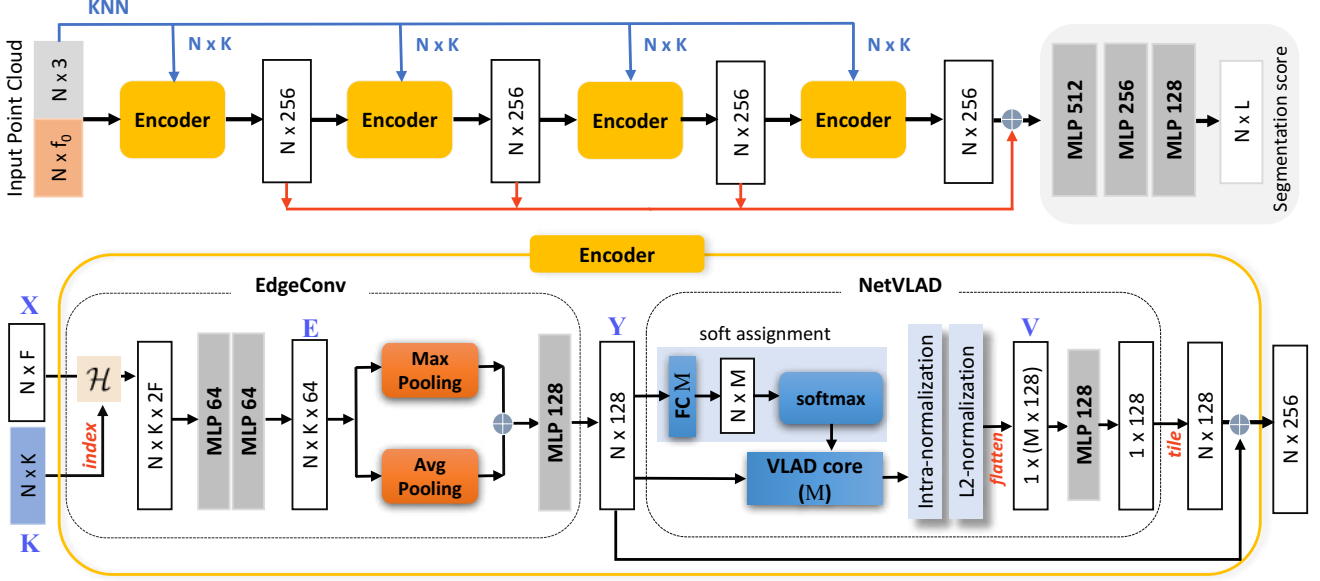


Figure 2. The network architecture of our PS<sup>2</sup>-Net.

coordinate can be permuted into a latent and potentially canonical order with the  $\chi$ -transformation. The learned  $\chi$ -transformation is further packed with typical convolutions to form a new process to extract features from local regions. However, the  $\chi$ -transformation is not permutation invariant. KCNet [21] presents a kernel correlation layer to exploit local geometric structures, where point-set kernel correlation is used to measure similarities between local neighborhood graphs and learned kernel graphs. Despite showing promising performance on object-level related tasks, it is somewhat sensitive to underlying graph structures and its capability in scene-level semantic segmentation is unknown. DGCNN [26] explicitly designs the EdgeConv module to better capture local geometric features of point clouds while preserving permutation invariance. In particular, they dynamically compute a  $k$ -nearest neighbors (KNN) graph for each point in the feature spaces produced by different EdgeConv layers. EdgeConv applies multiple MLP layers over the combination of point-wise features and their respective KNN features to generate edge features, and outputs the enhanced point-wise features by pooling among neighboring edge features. However, the features learned from DGCNN does not encode contextual and semantic information of the whole 3D scene, i.e. global context. Furthermore, the dynamic re-computation of the KNN graphs during network training is extremely computationally expensive and inefficient when applied on large-scale point clouds.

Our PS<sup>2</sup>-Net leverages on EdgeConv to learn the local information, but uses static KNN graphs defined in the spatial coordinate space to reduce computational complexity. Additionally, we use the permutation invariant NetVLAD layer [1] to incorporate the global contextual information into our learned feature.

### 3. Our PS<sup>2</sup>-Net

The network architecture of our PS<sup>2</sup>-Net is presented in Figure 2. Our network consists of four encoders that are repeatedly stacked together, and a final segment of shared 3-layer MLPs. The encoders output different levels of feature abstractions that are subsequently merged and passed to the shared MLPs for point level semantic classification. The input to our network is an  $N \times (3 + f_0)$  matrix of  $N$  points  $\in \mathbb{R}^3$  and an additional feature  $\in \mathbb{R}^{f_0}$  (e.g. color or surface normal). The network output is an  $N \times L$  matrix of predicted probabilities for the  $L$  labels on each of the  $N$  points. A KNN search is done on each of the  $N$  points in the  $\mathbb{R}^3$  space to produce an  $N \times K$  matrix. This  $N \times K$  matrix is fed into each of the encoder to group local regions in the corresponding feature spaces.

An encoder is made up of two components - EdgeConv and NetVLAD. The  $N \times 128$  output of EdgeConv is fed into NetVLAD, and then concatenated with the  $N \times 128$  output of NetVLAD to generate the final  $N \times 256$  output of the encoder. Each encoder first forms an  $N \times K \times 2F$  tensor with the operator  $\mathcal{H}$  (see next paragraph for more details) using the set of input features  $\in \mathbb{R}^F$  that are the  $K$ -nearest neighbors  $\in \mathbb{R}^3$ . This  $N \times K \times 2F$  tensor is fed into several MLP layers to extract the  $N \times K \times 64$  edge features that describe the relationships between each point and its neighbors. The feature dimension  $F$  equals to  $(3 + f_0)$  and 256 for the first and subsequent encoders, respectively. We then apply two symmetric operations, i.e. channel-wise max-pooling and average-pooling, to respectively transform the  $K$  edge features into two locally aggregated representations. Subsequently, the two representations are concatenated and fed into another MLP layer to produce a set of point-wise fea-

tures ( $N \times 128$ ) as the output of EdgeConv.

More formally, let us denote the input feature matrix to EdgeConv as  $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N \mid \mathbf{x}_i \in \mathbb{R}^F\}$ , the indices of KNN as  $\mathbf{K} = \{\mathbf{k}_1, \dots, \mathbf{k}_N \mid \mathbf{k}_i \in \mathbb{R}^K\}$ , the edge feature tensor as  $\mathbf{E} = \{\mathbf{e}_1, \dots, \mathbf{e}_N \mid \mathbf{e}_i \in \mathbb{R}^{K \times 64}\}$ , and output feature matrix of the EdgeConv as  $\mathbf{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_N \mid \mathbf{y}_i \in \mathbb{R}^{128}\}$ . Then, the  $\mathcal{H}$  operator is given by:

$$\mathcal{H} : \mathbf{X} \longrightarrow \mathbf{H}, \quad (1)$$

where

$$\mathbf{H} = \{\mathbf{h}_1, \dots, \mathbf{h}_N \mid \mathbf{h}_i \in \mathbb{R}^{K \times 2F}\}, \quad (2a)$$

$$\mathbf{h}_i(j) = [\mathbf{x}_i, \mathbf{x}_j - \mathbf{x}_i], \quad j \in \mathbf{k}_i. \quad (2b)$$

$\mathbf{h}_i(j)$  is the  $j^{th}$  row of  $\mathbf{h}_i$  and  $[\mathbf{A}, \mathbf{B}]$  represents the concatenation of matrices  $\mathbf{A}$  and  $\mathbf{B}$ . The edge features  $\mathbf{E}$  are obtained by:

$$\mathbf{e}_i = \sigma(\mathcal{F}(\sigma(\mathcal{F}(\mathbf{h}_i; \Theta_1)); \Theta_2)), \quad (3)$$

where  $\mathcal{F}$  represents a shared MLP layer, and  $\sigma$  represents a non-linear ReLU activation function.  $\Theta_1$  and  $\Theta_2$  are two sets of learnable parameters in the two respective shared MLP layers. Finally, the output of edgeConv  $\mathbf{Y}$  is given by:

$$\mathbf{y}_i = \sigma(\mathcal{F}([\max_{k \in \mathbf{k}_i}(\mathbf{e}_k), \text{avg}_{k \in \mathbf{k}_i}(\mathbf{e}_k)]; \Theta_3)), \quad (4)$$

where  $\Theta_3$  is the set of learnable parameters for the EdgeConv output shared MLP.

By taking  $\mathbf{Y}$  as input, NetVLAD outputs an  $1 \times (M \times 128)$  global descriptor, denoted as  $\mathbf{V}$ . The  $m^{th}$  128-vector is given by:

$$\mathbf{v}_m = \sum_{i=1}^N \frac{e^{\mathbf{w}_m^T \mathbf{y}_i + b_m}}{\sum_{m'=1}^M e^{\mathbf{w}_{m'}^T \mathbf{y}_i + b_{m'}}} (\mathbf{y}_i - \mathbf{c}_m). \quad (5)$$

There are two sets of parameters in the NetVLAD module:  $M$  cluster centers (“visual words”)  $\{\mathbf{c}_1, \dots, \mathbf{c}_M \mid \mathbf{c}_m \in \mathbb{R}^{128}\}$  and  $\{\mathbf{w}_m, b_m\}$  for learning the soft assignments, which determines the propagation of information from the input point-wise feature vectors to cluster centroids. Specifically, NetVLAD uses a MLP with  $M$  feature channels followed by a soft-max function to obtain the soft-assignments. Next, the residuals between input point-wise descriptors and  $M$  cluster centroids are aggregated with the soft-assignments. Finally, the matrix is flattened into an  $1 \times (M \times 128)$  output vector  $\mathbf{V}$  with intra- and inter-normalization. To be computational efficient, this high dimensional vector is further compressed into a concise feature representation via a MLP, which is treated as the final global descriptor.

### 3.1. Exploitation of Local Structure

As mentioned earlier, we take inspiration from EdgeConv in [26] to exploit the local structures among the local neighborhood of a point. We replace the original dynamic KNN graphs computation with static KNN graphs

computed on metric space for two reasons. First, we assume that the static metric-based KNN graphs may supervise local structure learning with spatial constraint. Second, static KNN graphs is analogous to the structure of images, where the neighborhoods of pixels remain fixed during convolution. Additionally, the two symmetric operations in aggregating edge features, i.e. channel-wise max-pooling and average-pooling, are designed to compensate the information loss during aggregation. Max-pooling operator individually performs over  $K$  dimensions and selects the maximum feature responses over the neighborhood of each point, while average pooling operator sums up all the features in the same local region. By combining these two operations, fine-grained local information are preserved and the strong responses are emphasized. We progressively expand the receptive field by repeatedly stacking the encoders.

### 3.2. Aggregation of Global Context

In addition to exploiting the local structures, we also design our network to be aware of the global context that provides scene-level semantic information. This information is potentially useful and can alleviate local confusions. Particularly, semantic context helps to distinguish patches with similar appearances/geometrics but different semantic meanings (e.g. differentiate a white wall from a white board on it). To incorporate this globally contextual information, we leverage on the success of NetVLAD [1], which is a technique originally designed for aggregating local descriptors into a global vector in the image domain. [25] adopted it into PointNetVLAD that generates global descriptors for point-based inputs. The descriptor vector of each cluster center is a summation of residuals (i.e. contributions) of each input feature to the center. Consequently, it is able to reveal fine-grained global contexts due to the large receptive field and aggregation of the learned relationships with all points.

### 3.3. Permutation Invariance

We now prove that the PS<sup>2</sup>-Net is permutation-invariant.

**Lemma 1.** *The PS<sup>2</sup>-Net is permutation invariant, i.e. if the rows of the input point cloud matrix are permuted, the output of the network remains unchanged.*

*Proof.* Let  $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_N\}$  denote the input  $N \times 3$  matrix, and  $\mathbf{L}$  denote the output  $N \times L$  matrix. Now we want to prove if the input is  $\pi\mathbf{P}$  where  $\pi$  is an  $N \times N$  permutation matrix, while the output of PS<sup>2</sup>-Net remains as  $\mathbf{L}$ . As the shared MLPs operating on individual points are obviously permutation-invariant. Here we simplify the proof by proving the permutation invariance property of the encoder in PS<sup>2</sup>-Net. Suppose we have a permuted point cloud

$$\tilde{\mathbf{P}} = \{\mathbf{p}_1, \dots, \mathbf{p}_{i-1}, \mathbf{p}_j, \mathbf{p}_{i+1}, \dots, \mathbf{p}_{j-1}, \mathbf{p}_i, \mathbf{p}_{j+1}, \dots, \mathbf{p}_N\}$$



that only reorders points  $\mathbf{p}_i$  and  $\mathbf{p}_j$  in  $\mathbf{P}$ , then the feature representation of  $\tilde{\mathbf{P}}$  is given by

$$\tilde{\mathbf{X}} = \{\mathbf{x}_1, \dots, \mathbf{x}_{i-1}, \mathbf{x}_j, \mathbf{x}_{i+1}, \dots, \mathbf{x}_{j-1}, \mathbf{x}_i, \mathbf{x}_{j+1}, \dots, \mathbf{x}_N\}.$$

As the reordering does not affect the order of nearest neighbors, the KNN indices are still given by  $\mathbf{K}$ . Inputting  $\tilde{\mathbf{X}}$  and  $\mathbf{K}$  into the encoder, the output is formulated as:

$$\tilde{\mathbf{V}} = f^\nu(f^e(\tilde{\mathbf{X}}, \mathbf{K})), \quad (6)$$

where  $f^e(\cdot)$  denotes a series of operations in EdgeConv  $\tilde{\mathbf{Y}} = f^e(\tilde{\mathbf{X}}, \mathbf{K})$  and  $f^\nu(\cdot)$  denotes the NetVLAD function  $\tilde{\mathbf{V}} = f^\nu(\tilde{\mathbf{Y}})$ .

Putting the  $\tilde{\mathbf{X}}$  into the operator  $\mathcal{H}$  in Equation 2, we get

$$\tilde{\mathbf{H}} = \{\mathbf{h}_1, \dots, \mathbf{h}_{i-1}, \mathbf{h}_j, \mathbf{h}_{i+1}, \dots, \mathbf{h}_{j-1}, \mathbf{h}_i, \mathbf{h}_{j+1}, \dots, \mathbf{h}_N\}.$$

Since each  $\mathbf{e}_i$  is processed independently in the shared MLPs (see Equation 3), it is obvious that

$$\tilde{\mathbf{E}} = \{\mathbf{e}_1, \dots, \mathbf{e}_{i-1}, \mathbf{e}_j, \mathbf{e}_{i+1}, \dots, \mathbf{e}_{j-1}, \mathbf{e}_i, \mathbf{e}_{j+1}, \dots, \mathbf{e}_N\}.$$

Finally, the output of edgeConv is given by

$$\tilde{\mathbf{Y}} = \{\mathbf{y}_1, \dots, \mathbf{y}_{i-1}, \mathbf{y}_j, \mathbf{y}_{i+1}, \dots, \mathbf{y}_{j-1}, \mathbf{y}_i, \mathbf{y}_{j+1}, \dots, \mathbf{y}_N\},$$

again this is because each  $\mathbf{e}_i$  is processed independently in the shared edgeConv output MLP in Equation 4.

The output of NetVLAD  $\mathbf{V} = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_M\}$  of the original input features  $\mathbf{X}$  before permutation can now be written as

$$\begin{aligned} \mathbf{v}_m &= f_m^\nu(\mathbf{y}_1) + f_m^\nu(\mathbf{y}_2) + \dots + f_m^\nu(\mathbf{y}_N) \\ &= \sum_{s=1}^N f_m^\nu(\mathbf{y}_s), \quad \forall m, \end{aligned} \quad (7)$$

where

$$f_m^\nu(\mathbf{y}_s) = \frac{e^{\mathbf{w}_m^T \mathbf{y}_s + b_m}}{\sum_{m'=1}^M e^{\mathbf{w}_{m'}^T \mathbf{y}_s + b_{m'}}} (\mathbf{y}_s - \mathbf{c}_m); \quad (8)$$

and the output of NetVLAD with the permuted input features is given by

$$\begin{aligned} \tilde{\mathbf{v}}_m &= f_m^\nu(\mathbf{y}_1) + \dots + f_m^\nu(\mathbf{y}_{i-1}) + f_m^\nu(\mathbf{y}_j) + \\ &\quad f_m^\nu(\mathbf{y}_{i+1}) + \dots + f_m^\nu(\mathbf{y}_{j-1}) + f_m^\nu(\mathbf{y}_i) + \\ &\quad f_m^\nu(\mathbf{y}_{j+1}) + \dots + f_m^\nu(\mathbf{y}_N) \\ &= \sum_{s=1}^N f_m^\nu(\mathbf{y}_s) = \mathbf{v}_m, \quad \forall m. \end{aligned} \quad (9)$$

Hence,

$$\mathbf{V} = f^\nu(f^e(\tilde{\mathbf{X}}, \mathbf{K})). \quad (10)$$

This completes our proof that the encoder of PS<sup>2</sup>-Net is permutation invariant.  $\square$

## 4. Experiments

### 4.1. Datasets

We conduct experiments of our network on two challenging benchmark datasets : S3DIS [2] and ScanNet [7]. The details of these two datasets are as follows.

**S3DIS** This dataset consists of 6 different indoor areas from 3 different buildings. It has 271 rooms with various styles, *e.g.* conference room, lobby, restroom. Each point is labeled by one of 13 semantic classes. These classes are partitioned into *structural type* (ceiling, floor, wall, beam, column, window, door), *furniture type* (table, chair, sofa, bookcase, board) and *clutter*. In the experiments, we adopt the 6-fold training/testing split used in [18].

**ScanNet** This dataset consists of 1,513 scans from 707 unique indoor scenes. The space type is very diverse, ranging from very small (*e.g.* bathroom, closet, utility room) to very large (*e.g.* apartment, classroom, library) spaces. Each point is annotated by one out of 21 semantic classes, including 20 object classes plus 1 extra class representing free space. Following the experimental settings in [19], we adopt 1,201 scans for training and 312 scans for testing.

### 4.2. Implementation Details

Our PS<sup>2</sup>-Net consists of four repeatedly stacked encoders with the same configuration. Each EdgeConv module has two shared MLPs (64,64) to extract edge features and one shared MLP (128) to fuse the max- and avg- pooled edge features. The NetVLAD module has 16 clusters and produces a  $(16 \times 128)$ -dimensional global descriptor that is fed into a shared MLP (128) for dimension reduction. Skip link is added from the output of EdgeConv to the output of NetVLAD for integrating local and global features. The outputs from all the four encoders are concatenated and forwarded to three shared MLPs (512,256,128) to map the learned features to point labels. Dropout with a drop-ratio of 0.3 is used in the first layer. Batch normalization and ReLU are added to all the respective MLPs. The number of nearest neighbors is set to 20.

Our framework is implemented using PyTorch deep learning library on a NVIDIA GTX 1080Ti. We optimize the network using ADAM [12] with an initial learning rate of 0.001 and weight decay 1e-5. Multi-class cross-entropy is used as the loss function. In most experiments, the learning rate is decayed by half after every 100 epochs. In general, the networks converged at  $\sim 150$  epochs. The batch size is set to 6 for experiments with 4,096 points as input, otherwise, the batch size is updated with respect to the number of input points. Note that we do not use any data augmentation in our experiments.

We adopt the two widely used metrics: overall accuracy (OA) and mean intersection over union (mIoU) to evaluate

Table 1. Comparison of performances on S3DIS dataset using 6-fold cross validation. Results in the upper and lower table are obtained with data preparation setups in [18] (denoted as *P1*) and [15] (denoted as *P2*), respectively. Class-wise IoU is also given.

| Method                       | OA           | mIoU         | ceiling      | floor        | wall         | beam         | column       | window       | door         | table        | chair        | sofa         | bookcase     | board        | clutter      |
|------------------------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| PointNet [18]                | 78.5         | 47.6         | 88           | 88.7         | 69.3         | 42.4         | 23.1         | 47.5         | 51.6         | 54.1         | 42           | 9.6          | 38.2         | 29.4         | 35.2         |
| G+RCU [8]                    | 81.1         | 49.7         | 90.3         | 92.1         | 67.9         | <b>44.7</b>  | 24.2         | 52.3         | 51.2         | 58.1         | 47.4         | 6.9          | 39           | 30           | 41.9         |
| DGCNN [26]                   | 84.1         | 56.1         | —            | —            | —            | —            | —            | —            | —            | —            | —            | —            | —            | —            | —            |
| RNNCF [28]                   | <b>86.9</b>  | 56.3         | 92.9         | 93.8         | 73.1         | 42.5         | 25.9         | 47.6         | 59.2         | 60.4         | <b>66.7</b>  | 24.8         | <b>57</b>    | 36.7         | 51.6         |
| RSNet [10]                   | —            | 56.47        | 92.48        | 92.83        | 78.56        | 32.75        | 34.37        | 51.62        | <b>68.11</b> | 60.13        | 59.72        | <b>50.22</b> | 16.42        | 44.85        | 52.03        |
| <b>PS<sup>2</sup>-Net—P1</b> | 86.69        | <b>61.56</b> | <b>93.40</b> | <b>95.64</b> | <b>79.94</b> | 37.17        | <b>40.93</b> | <b>59.83</b> | 66.65        | <b>63.65</b> | 65.71        | 37.16        | 49.83        | <b>54.56</b> | <b>55.66</b> |
| PointCNN [15]                | 88.14        | 65.39        | <b>94.78</b> | <b>97.3</b>  | 75.82        | <b>63.25</b> | 51.71        | 58.38        | 57.18        | <b>71.63</b> | 69.12        | 39.08        | <b>61.15</b> | 52.19        | 58.59        |
| <b>PS<sup>2</sup>-Net—P2</b> | <b>88.22</b> | <b>66.60</b> | 93.04        | 96.26        | <b>83.22</b> | 41.61        | <b>54.05</b> | <b>60.08</b> | <b>70.40</b> | 67.37        | <b>73.13</b> | <b>48.75</b> | 58.73        | <b>58.68</b> | <b>60.48</b> |

the segmentation performance of our network. Additionally, we also report the class-wise IoU, which is computed for each point that belongs to its corresponding semantic class.

### 4.3. Segmentation on S3DIS Dataset

#### 4.3.1 Data preparation

We observed some differences in the data preparation setups among the existing approaches for S3DIS dataset. One widely-adopted setup is proposed by PointNet [18], which splits each room into non-overlapping blocks of  $1\text{m} \times 1\text{m}$  area on the  $xy$  plane and each point is represented by a 9-dim vector containing the  $(x, y, z)$  coordinates,  $(r, g, b)$  color and normalized coordinates. 4,096 points are randomly sampled from each block during training and testing. The other setup is presented by PointCNN [15]. It slices the rooms into  $1.5\text{m}$ -by- $1.5\text{m}$  blocks with  $0.3\text{m}$  padding on each side and each point is associated with a 6-dim vector containing the  $(x, y, z)$  coordinates and  $(r, g, b)$  color.  $\mathcal{N}(2048, 256^2)$  ( $\mathcal{N}$  denotes the Gaussian distribution) points are sampled from each block during training, while each block is sampled multiple times to make sure that all the points are evaluated during testing. In order to make fair comparisons, we conduct experiments using both data preparation setups<sup>1</sup> and report the comparisons accordingly.

#### 4.3.2 Results and Discussion

Table 1 illustrates the performance of our PS<sup>2</sup>-Net in comparison to previous state-of-the-art approaches on the S3DIS dataset. All approaches [8, 10, 26, 28] from the upper table prepared their experimental data according to the data preparation setup in PointNet [18]. We can see that our PS<sup>2</sup>-Net achieves the best performance in the mIoU criteria, which is a more precise criteria than the overall accuracy as the dataset is highly unbalanced. In particular, our PS<sup>2</sup>-Net improves the mIoU by 9.7% and overall accuracy by 3% when compared with DGCNN [26], which is made up of a stacking of EdgeConv layers. We argue that the incorporation of global context in our network contributes to this

<sup>1</sup>We use the codes in [PointNet](#) and [PointCNN](#) for preprocessing.

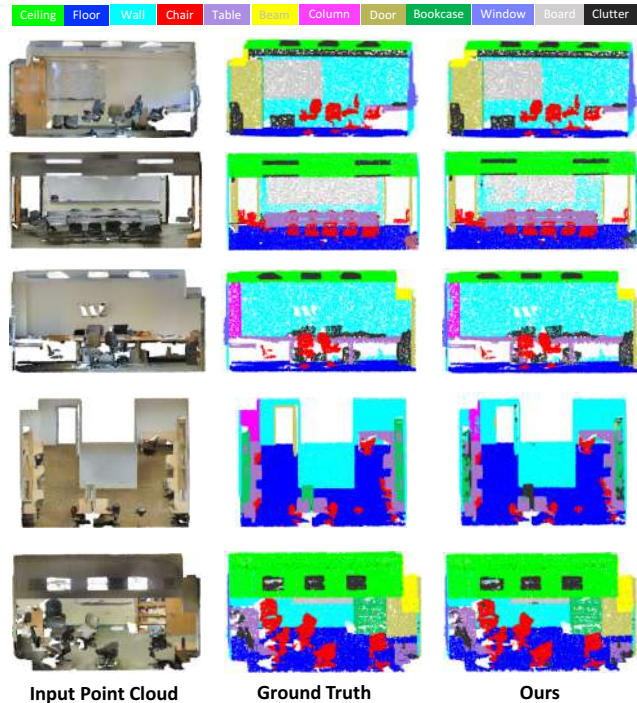


Figure 3. PS<sup>2</sup>-Net semantic segmentation results on S3DIS.

significant improvement. Additionally, our PS<sup>2</sup>-Net outperforms the three existing RNN-based methods [8, 10, 28] in the mIoU criteria.

In the lower part of Table 1, we show competitive performance of our PS<sup>2</sup>-Net with PointCNN [15] when the same data preprocessing setup is used. Moreover, the performance of our PS<sup>2</sup>-Net—P2 increases by 8.2% in mIoU compared to PS<sup>2</sup>-Net—P1. This indicates that data preprocessing plays an important role in training, and the extensive online sampling used in PointCNN is able to expose the model to larger groups of samples during training.

Furthermore, several qualitative results on the S3DIS dataset are shown in Figure 3. As we can see from the examples, the dataset is very challenging in many scenarios, *e.g.* “the white boards on the white wall”, “the open doors with only visible door frames”, and “the white column in the boundaries of the white wall” *etc.* Interestingly, our PS<sup>2</sup>-Net successfully segments the boards, doors, columns

Table 2. Comparison of performances on ScanNet dataset using XYZ information as input. Results in the upper and lower table are obtained with data preparation setups in [19] (denoted as P3) and [15] (denoted as P2), respectively. Class-wise IoU is also given.

| Method                       | OA           | mIoU         | wall         | floor       | chair        | table        | desk         | bed          | bookshelf    | sofa         | sink         | bathtub      |
|------------------------------|--------------|--------------|--------------|-------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| PointNet [18]                | —            | 14.69        | 69.44        | 88.59       | 35.93        | 32.78        | 2.63         | 17.96        | 3.18         | 32.79        | 0            | 0.17         |
| PointNet++ [19]              | —            | 34.26        | 77.48        | 92.5        | 64.55        | 46.6         | 12.69        | 51.32        | 52.93        | 52.27        | 30.23        | 42.72        |
| RSNet [10]                   | —            | 39.35        | <b>79.23</b> | <b>94.1</b> | 64.99        | <b>51.04</b> | <b>34.53</b> | <b>55.95</b> | <b>53.02</b> | <b>55.41</b> | <b>34.84</b> | 49.38        |
| <b>PS<sup>2</sup>-Net—P3</b> | —            | <b>40.17</b> | 72.44        | 91.51       | <b>65.08</b> | 45.61        | 26.27        | 48.90        | 39.96        | 53.94        | 24.58        | <b>64.78</b> |
| PointCNN [15]                | 85.1         | —            | —            | —           | —            | —            | —            | —            | —            | —            | —            | —            |
| <b>PS<sup>2</sup>-Net—P2</b> | <b>87.21</b> | 44.90        | 77.02        | 91.22       | 68.36        | 56.66        | 31.62        | 53.55        | 36.32        | 58.75        | 43.07        | 70.11        |

| Method                       | toilet       | curtain      | counter      | door        | window       | shower curtain | refridgerator | picture     | cabinet      | other furniture |
|------------------------------|--------------|--------------|--------------|-------------|--------------|----------------|---------------|-------------|--------------|-----------------|
| PointNet [18]                | 0            | 0            | 5.09         | 0           | 0            | 0              | 0             | 0           | 4.99         | 0.13            |
| PointNet++ [19]              | 31.37        | 32.97        | 20.04        | 2.02        | 3.56         | 27.43          | 18.51         | 0           | 23.81        | 2.2             |
| RSNet [10]                   | 54.16        | 6.78         | 22.72        | 3           | 8.75         | 29.92          | <b>37.9</b>   | 0.95        | <b>31.29</b> | <b>18.98</b>    |
| <b>PS<sup>2</sup>-Net—P3</b> | <b>60.87</b> | <b>41.20</b> | <b>24.62</b> | <b>8.37</b> | <b>21.55</b> | <b>47.24</b>   | 19.68         | <b>2.63</b> | 28.02        | 16.22           |
| PointCNN [15]                | —            | —            | —            | —           | —            | —              | —             | —           | —            | —               |
| <b>PS<sup>2</sup>-Net—P2</b> | 66.28        | 41.94        | 23.73        | 10.94       | 17.82        | 51.02          | 44.19         | 3.17        | 32.82        | 19.33           |

in most cases despite the similar colors (e.g. wall vs column) or geometries (e.g. wall vs board). We believe that the correct classifications are consequences of integrating local structures and global context into our network.

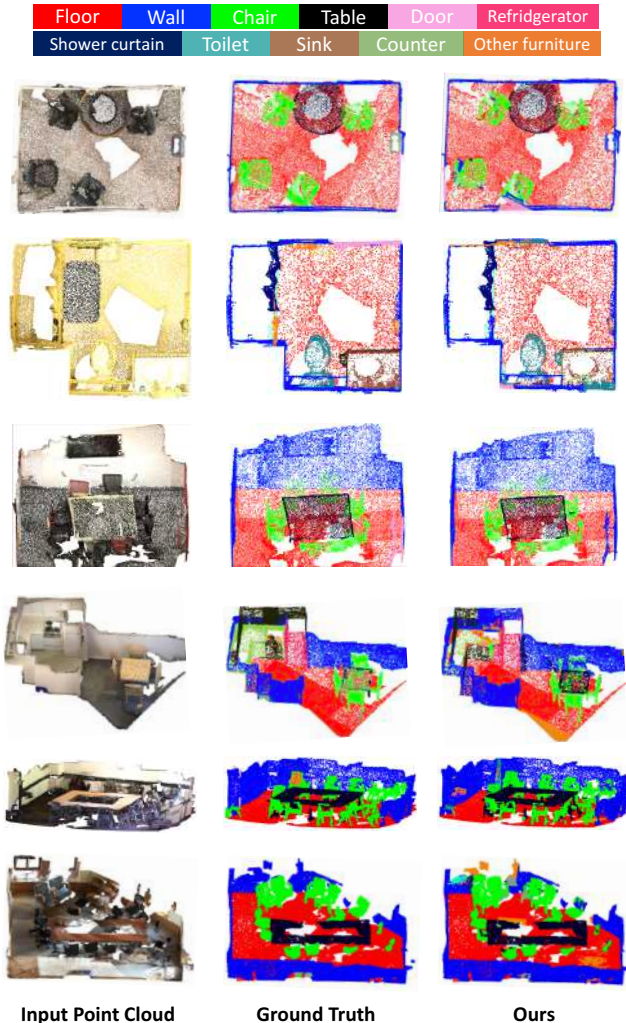


Figure 4. PS<sup>2</sup>-Net semantic segmentation results on ScanNet.

## 4.4. Segmentation on ScanNet Dataset

### 4.4.1 Data preparation

Similar to S3DIS, there are two data preparation setups in the ScanNet dataset. The first setup is proposed by PointNet++ [19]. It follows the same setup in [7] to first generate a tessellation of  $1.5\text{m} \times 1.5\text{m} \times 3\text{m}$  cubes with  $2\text{cm}^3$  voxels from the 3D point clouds. Next, cubes with  $\geq 2\%$  occupied voxels and  $\geq 70\%$  valid annotations on the voxel surfaces are extracted.

During training, 8,192 points are sampled from each cube. Each point is represented by only the  $(x, y, z)$  coordinates. The second setup is proposed by PointCNN [15]. It prepares the data in the same way as S3DIS except that only the  $(x, y, z)$  coordinates are used on each point. Additionally, it converts the the segmentation results on the test data into semantic voxel labeling for comparison with test results from the first data preprocessing setup. Again, to make fair comparisons, we conduct experiments using both data processing setups<sup>2</sup>. Note that different from S3DIS, we only use the  $(x, y, z)$  information as the input in this dataset in order to be compliant with the previous approaches.

### 4.4.2 Results and Discussion

The comparison of performances on the ScanNet dataset is summarized in Table 2. It can be seen that the performance of our PS<sup>2</sup>-Net outperforms the previous state-of-the-art methods [10, 19] in the mIoU criteria when the preprocessing setup in PointNet++ [19] is used. Notably, we achieve remarkable improvements in the classification results of several challenging classes with very few training data, i.e. bathtub (0.3%), toilet (0.3%), curtain (1.5%), window (0.9%) and shower curtain (0.2%). We reckon that the increase in performance comes from the use of EdgeConv, where the discriminative representations learned from lo-

<sup>2</sup>We use the code of PointNet++ for preprocessing.



Table 3. Ablation test of PS<sup>2</sup>-Net variants on S3DIS [A5].

| Model                         | OA           | mIoU         |
|-------------------------------|--------------|--------------|
| w/o local features            | 81.66        | 45.25        |
| w/o EdgeConv                  | 83.50        | 50.35        |
| w/o NetVLAD                   | 84.00        | 50.18        |
| Our full PS <sup>2</sup> -Net | <b>84.60</b> | <b>52.95</b> |

cal structures are superior to the point-wise features that are individually processed in [10, 18, 19]. We obtain an impressive improvement in mIoU (11.8%) on PS<sup>2</sup>-Net—P2 compared to PS<sup>2</sup>-Net—P3; and furthermore, we surpass PointCNN [15]. Particularly, our PS<sup>2</sup>-Net achieves more accurate segmentation results (class-wise IoU > 0.4) in 10 out of 20 classes.

Several segmentation results are visualized in Figure 4. Our PS<sup>2</sup>-Net is able to recognize both frequently, *e.g.* wall, floor, chair, and rarely, *e.g.* toilet, sink, refrigerator, seen objects.

#### 4.5. Ablation studies

In this section, we investigate the contributions of the respective components (*i.e.* EdgeConv and NetVLAD) in our PS<sup>2</sup>-Net, and evaluate the effects of several key hyper-parameters (*i.e.* number of nearest neighbors  $K$  in EdgeConv, clusters  $M$  in NetVLAD, and stacked encoders). We conduct ablation experiments on the fifth fold of the S3DIS dataset, *i.e.* we test on Area 5 and train on the remaining data. More specifically, the testing area is collected in a separated building from all the training areas. All settings remain unchanged as the baseline PS<sup>2</sup>-Net in the ablation experiments except the target hyper-parameter.

**Effectiveness of network modules** We study the effect of each network module by removing them individually from the network, and compare the performances before and after the removal. To further measure the contribution of local features, we design two settings: in (1) “w/o local features”, we substitute the EdgeConv module with a point-wise feature learning module which does not exploit local structures, and in (2) “w/o EdgeConv”, we apply point-wise feature learning followed by a max-pooling operator over  $K$ -nearest neighbors to extract the local features. Additionally, in “w/o NetVLAD” we replace NetVLAD with a max-pooling operation similar to DGCNN [26]. Table 3 shows the comparison results. The poor performance of “w/o local features” in mIoU accords well with our claim of the importance of local information in modeling fine-grained structures. Furthermore, we can see that the integration of local structures and global context in our PS<sup>2</sup>-Net contribute to the improvements over “w/o EdgeConv” and “w/o NetVLAD”.

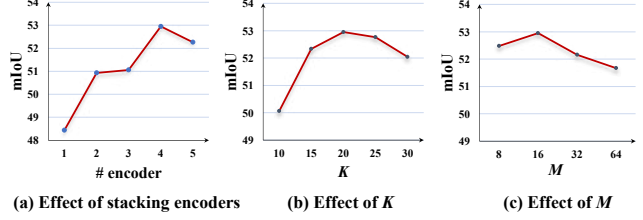


Figure 5. Effectiveness of hyper-parameters on S3DIS [A5].

**Number of encoders** We evaluate the performance of our PS<sup>2</sup>-Net with different number of stacked encoders, and show the results in Figure 5(a). Generally, the network becomes deeper and is capable of learning more discriminative representations when the number of encoders increases. This can be seen from the improvement of performance when the number of encoders is increased from 1 to 4. However, we also observe that the performance begins to drop after a certain number of encoders. This might be due to a deeper network with more parameters requires more training data to prevent overfitting. In other words, a deeper network may fail to generalize well to new cases when the training dataset is not sufficiently large. Hence, we use the best setting of 4 encoders in all our experiments.

**Number of nearest neighbors** This hyper-parameter controls the range of local regions and thus influences the amount of local information included in our network. From Figure 5(b), we can see a trade-off in selecting  $K$ . A small  $K$  may lead to limited regions with insufficient local context, while a big  $K$  may bring irrelevant noises and increases the computation complexity. We set  $K$  to 20 in the experiments according to Figure 5(b).

**Number of clusters** As is revealed in Figure 5(c), this hyper-parameter gives lower performances when  $M$  is bigger than 16. It is likely because we reduce the dimension of  $V$  from  $(M \times 128)$ -dim to 128-dim. If  $M$  is too big, the dimension reduction operation may induce uncertain information loss. Consequently, we selected  $M=16$  that achieved the best performance.

## 5. Conclusion

In this paper, we proposed the PS<sup>2</sup>-Net, an end-to-end deep neural network for the 3D point cloud semantic segmentation task. PS<sup>2</sup>-Net is built on four repeatedly stacked encoders, where each encoder has two basic components: EdgeConv and NetVLAD that capture local structures and global context, respectively. We provided proof to guarantee the permutation invariance property of our PS<sup>2</sup>-Net. We obtained state-of-the-art performances with our locally and globally aware PS<sup>2</sup>-Net on the two challenging 3D indoor scene datasets for point-based semantic segmentation.



## References

- [1] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *CVPR*, pages 5297–5307, 2016. [2](#), [3](#), [4](#)
- [2] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3d semantic parsing of large-scale indoor spaces. In *CVPR*, pages 1534–1543, 2016. [5](#)
- [3] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (12):2481–2495, 2017. [1](#)
- [4] A. Boulch, J. Guerry, B. Le Saux, and N. Audebert. Snapnet: 3d point cloud semantic labeling with 2d deep segmentation networks. *Computers & Graphics*, 71:189–198, 2018. [1](#)
- [5] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv preprint arXiv:1608.04236*, 2016. [1](#)
- [6] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 40(4):834–848, 2018. [1](#)
- [7] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Niessner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, pages 5828–5839, 2017. [1](#), [5](#), [7](#)
- [8] F. Engelmann, T. Kontogianni, A. Hermans, and B. Leibe. Exploring spatial context for 3d semantic segmentation of point clouds. In *CVPR*, pages 716–724, 2017. [1](#), [2](#), [6](#)
- [9] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *ICCV*, pages 2980–2988. IEEE, 2017. [1](#)
- [10] Q. Huang, W. Wang, and U. Neumann. Recurrent slice networks for 3d segmentation of point clouds. In *CVPR*, pages 2626–2635, 2018. [1](#), [2](#), [6](#), [7](#), [8](#)
- [11] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri. 3d shape segmentation with projective convolutional networks. In *CVPR*, pages 6630–6639, 2017. [1](#)
- [12] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. [5](#)
- [13] L. Landrieu and M. Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR*, pages 4558–4567. [1](#), [2](#)
- [14] F. J. Lawin, M. Danelljan, P. Tosteberg, G. Bhat, F. S. Khan, and M. Felsberg. Deep projective 3d semantic segmentation. In *International Conference on Computer Analysis of Images and Patterns*, pages 95–107. Springer, 2017. [1](#)
- [15] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems*, pages 820–830, 2018. [1](#), [2](#), [6](#), [7](#), [8](#)
- [16] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015. [1](#)
- [17] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015. [1](#)
- [18] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 652–660, 2017. [1](#), [2](#), [5](#), [6](#), [7](#), [8](#)
- [19] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017. [1](#), [2](#), [5](#), [7](#), [8](#)
- [20] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. [1](#)
- [21] Y. Shen, C. Feng, Y. Yang, and D. Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *CVPR*, volume 4, 2018. [2](#), [3](#)
- [22] A. Sinha, J. Bai, and K. Ramani. Deep learning 3d shape surfaces using geometry images. In *ECCV*, pages 223–240. Springer, 2016. [1](#)
- [23] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *CVPR*, pages 945–953, 2015. [1](#)
- [24] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, and S. Savarese. Segcloud: Semantic segmentation of 3d point clouds. In *3D Vision (3DV), 2017 International Conference on*, pages 537–547. IEEE, 2017. [1](#)
- [25] M. A. Uy and G. H. Lee. Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition. *CVPR*, 2018. [4](#)
- [26] 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*, 2019. [2](#), [3](#), [4](#), [6](#), [8](#)
- [27] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–1920, 2015. [1](#)
- [28] X. Ye, J. Li, H. Huang, L. Du, and X. Zhang. 3d recurrent neural networks with context fusion for point cloud semantic segmentation. In *ECCV*, pages 415–430. Springer, 2018. [1](#), [2](#), [6](#)
- [29] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. In *Advances in Neural Information Processing Systems*, pages 3391–3401, 2017. [1](#)
- [30] W. Zeng and T. Gevers. 3dcontextnet: Kd tree guided hierarchical learning of point clouds using local and global contextual cues. In *ECCV*, pages 314–330, 2018. [1](#)

## Appendix A. More Visualizations and Discussions on PS<sup>2</sup>-Net Variants

Fig. 6 shows the visualizations of some qualitative results from the ablation studies - “w/o Local”, “w/o EdgeConv”, “w/o NetVLAD”, and “full PS<sup>2</sup>-Net” mentioned in Sec. 4.5 of our main paper. We have several interesting findings on the effectiveness of integrating the local and global contexts from these qualitative results:

- The method “w/o Local” (3rd column of Fig. 6) shows difficulties in capturing local structures, such as plane or corner, and convex or concave, without the exploitation of local context. It can be seen from the results that the method fails to differentiate between “column” from “wall”, which tend to have similar color but different geometric structures. Additionally, the segmentation regions are not homogeneous because each point is processed independently in “w/o Local”. See “column” in Row A, “chair” in Row B, and “bookcase” in Row E.
- In comparison to “w/o Local”, “w/o EdgeConv” (4th column of Fig. 6) considers neighboring points to some extent. Hence, it performs slightly better in capturing the geometric differences, which can be seen from the correct classification of “column” in Row B and E. However, despite the consideration of local regions, “w/o EdgeConv” still treats each point in its local region independently. Consequently, this leads to insufficient exploitation of the fine-grained local structures. As we can see from the wrongly segmented “chair” and “clutter” (on the “table”) in Row B and D, “w/o EdgeConv” is confused by the neighboring points from the “bookcase” class. This indicates that it lacks the ability to capture complex geometry in classes such as “clutter” and “chair”.
- We observe that “w/o NetVLAD” (5th column of Fig. 6) fails to model scene-level semantic information without the aggregation of the global context. As shown in Row A, “w/o NetVLAD” is confused with the “chairs” and “board” classes, and this does not happen with the global context. Similar confusions can be seen in Row D and F, where the “wall” class is confused with the “table” and “chair” classes, respectively.
- Our **full PS<sup>2</sup>-Net** (6th column of Fig. 6) that incorporates both local structures and global context achieves the best segmentation results compared to the other methods that remove either the local or global context exploitation module. This suggests that the locally and globally aware framework we proposed is able to capture fine-grained local structures (e.g. successful segmentation of “column” class in Row A-F, distinction

between “clutter” and “bookcase” classes in Row B and D) as well as the global context (e.g. distinction between the white “wall” and white “board” classes in Row B,C,D,F)

## Appendix B. More Discussions on Unbalance Dataset Problem

The two datasets used in our experiments are very challenging as both of them are highly unbalanced. The statistics of data portion of the two datasets are given in Table 4 and Table 5, respectively. As seen in Table 4, “wall”, “ceiling”, “floor” that are the dominant classes on S3DIS have 10 ~ 66 times more data than the rare classes, such as “sofa”, “board”, and “beam”. This unbalance problem is more severe on the ScanNet dataset, where the most dominate class (i.e. “wall”) has over 200 times more training data than the rarest class (i.e. “shower curtain”). In the experiments, we did not implement any strategy (e.g. over-sampling and weighted loss based on class frequency) to explicitly solve this problem. Interestingly, we still achieve acceptable performance on the rarest classes, such as “column”, “window”, “board” on the S3DIS dataset, and “bathtub”, “toilet”, “window”, “shower curtain” on the ScanNet dataset. This may indicate that our proposed method is robust to unbalanced data distributions.

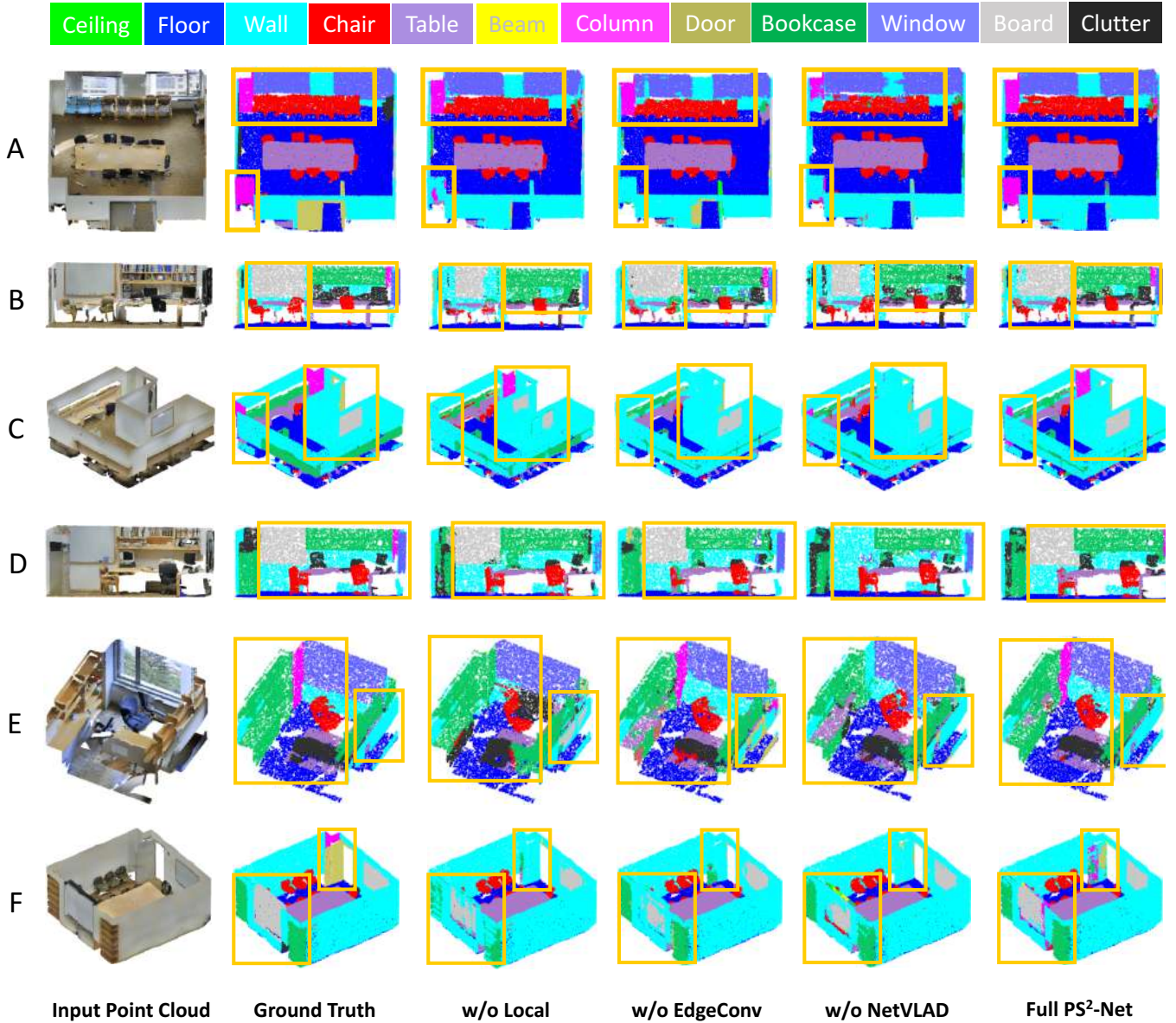


Figure 6. Visualized examples for ablation test of PS<sup>2</sup>-Net variants on S3DIS [A5]. The interesting areas are highlighted by yellow bounding boxes. Best viewed in color.

Table 4. Statistics of class-wise instance number and point portion on S3DIS dataset.

| Class name         | ceiling | floor | wall  | beam | column | window | door | table | chair | sofa | bookcase | board | clutter |
|--------------------|---------|-------|-------|------|--------|--------|------|-------|-------|------|----------|-------|---------|
| # object           | 391     | 290   | 1,552 | 165  | 260    | 174    | 549  | 461   | 1,369 | 61   | 590      | 143   | —       |
| Data percentage(%) | 19.27   | 16.52 | 27.81 | 1.73 | 2.02   | 2.52   | 4.78 | 3.39  | 3.43  | 0.42 | 6.33     | 1.24  | 10.54   |

Table 5. Statistics of class-wise point portion on ScanNet dataset.

| Class name               | wall  | floor | chair | table | desk | bed  | bookshelf | sofa | sink | bathhtub | toilet |
|--------------------------|-------|-------|-------|-------|------|------|-----------|------|------|----------|--------|
| Train data percentage(%) | 36.8  | 24.90 | 4.60  | 2.53  | 1.66 | 2.58 | 2.04      | 2.59 | 0.34 | 0.34     | 0.27   |
| Test data percentage(%)  | 36.46 | 24.38 | 5.36  | 2.90  | 1.65 | 2.14 | 2.15      | 2.44 | 0.33 | 0.22     | 0.26   |

| Class name               | curtain | counter | door | window | shower curtain | refridgerator | picture | cabinet | other furniture |
|--------------------------|---------|---------|------|--------|----------------|---------------|---------|---------|-----------------|
| Train data percentage(%) | 1.48    | 0.62    | 2.33 | 0.94   | 0.18           | 0.43          | 0.37    | 2.59    | 2.46            |
| Test data percentage(%)  | 0.98    | 0.65    | 2.15 | 0.66   | 0.10           | 0.34          | 0.17    | 2.43    | 3.23            |