Superpoint Network for Point Cloud Oversegmentation

Le Hui, Jia Yuan, Mingmei Cheng, Jin Xie*, Xiaoya Zhang and Jian Yang*
PCA Lab, Nanjing University of Science and Technology, China
{le.hui, yajha, chengmm, csjxie, zhangxiaoya, csjyang}@njust.edu.cn

Abstract

Superpoints are formed by grouping similar points with local geometric structures, which can effectively reduce the number of primitives of point clouds for subsequent point cloud processing. Existing superpoint methods mainly focus on employing clustering or graph partition to generate superpoints with handcrafted or learned features. Nonetheless, these methods cannot learn superpoints of point clouds with an end-to-end network. In this paper, we develop a new deep iterative clustering network to directly generate superpoints from irregular 3D point clouds in an end-to-end manner. Specifically, in our clustering network, we first jointly learn a soft point-superpoint association map from the coordinate and feature spaces of point clouds, where each point is assigned to the superpoint with a learned weight. Furthermore, we then iteratively update the association map and superpoint centers so that we can more accurately group the points into the corresponding superpoints with locally similar geometric structures. Finally, by predicting the pseudo labels of the superpoint centers, we formulate a label consistency loss on the points and superpoint centers to train the network. Extensive experiments on various datasets indicate that our method not only achieves the state-of-the-art on superpoint generation but also improves the performance of point cloud semantic segmentation. Code is available at `https://github.com/fpthink/SPNet`.

1. Introduction

Superpoints are an oversegmentation of point clouds, which can semantically group points of similar geometric features. They can capture redundancy of point clouds and greatly reduce the computational cost of subsequent point cloud processing algorithms. Due to the representational and computational efficiency of superpoints, they are becoming increasingly popular for use in point cloud processing tasks such as 3D object modeling [4] and point cloud semantic segmentation [26, 21]. Nonetheless, due to the complex geometric structures of point clouds, how to group geometrically similar points to form accurate superpoints is still a challenging problem.

In the past few years, research efforts have been dedicated to superpoint generation of point clouds. Most of superpoint generation methods rely on the hand-crafted features of point clouds to group similar points by employing the clustering or graph partition methods. For example, in the voxel cloud connectivity segmentation (VCCS) method [33], the spatial connectivity and fast point feature histograms (FPFHs) of point clouds are used as the local features to cluster superpoints. Lin *et al.* [29] proposed a subset selection method to generate superpoints, where the FPFHs of point clouds are also used to characterize the local structures of point clouds. *Guinard et al.* [14] extracted the local linearity, planarity, scattering and verticality features of point clouds to generate superpoints with a greedy graph-cut algorithm [24]. However, the performance of these methods is limited by the handcrafted features of point clouds. Lately, the supervised superpoint (SSP) method [21] uses a deep network to obtain deep embeddings of point clouds and combines them with a graph-structured deep metric learning to oversegment point clouds. Nonetheless, it still uses the optimization-based method in [14] to generate superpoints. Therefore, SSP is not an end-to-end superpoint generation method. Recently, in [18], the differentiable SLIC superpixel method on 2D images is proposed for end-to-end training by constructing a soft association map. We build upon the idea for end-to-end superpoint generation on 3D point clouds, which can be integrated into other trainable deep neural networks for point cloud processing tasks.

In this paper, we propose a simple yet effective end-to-end framework to generate superpoints on 3D point clouds. In our framework, we develop a deep iterative clustering network to learn the association map between the points

*Corresponding authors
Le Hui, Jia Yuan, Mingmei Cheng, Jin Xie, Xiaoya Zhang and Jian Yang are with PCA Lab, Key Lab of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education, and Jiangsu Key Lab of Image and Video Understanding for Social Security, School of Computer Science and Engineering, Nanjing University of Science and Technology, China.

and superpoint centers. Specifically, we first use a deep neural network to extract local geometric features of point clouds. Before clustering, we employ the farthest point sampling (FPS) algorithm in the coordinate space of point clouds to obtain the initial superpoint centers. Then, we adaptively learn the bilateral weights between the points and superpoint centers from the coordinate and feature spaces of point clouds simultaneously. For each point, by assigning different weights to the superpoint centers with different geometric structures, it is expected to selectively focus on the most similar superpoint centers. Based on the learned bilateral weights, we can construct a soft association map between the points and superpoint centers. Consequently, we update the features and coordinates of the superpoint centers by weighting the embeddings of the corresponding points with the constructed association map. By iteratively updating the learned bilateral weights and superpoint centers, we can gradually learn an accurate point-superpoint association map. Finally, by voting to predict the pseudo labels of superpoint centers with the point-superpoint association map, we formulate a label consistency loss on the points and corresponding superpoint centers to train the deep iterative clustering network. Experimental results on indoor and outdoor datasets including S3DIS [1], ScanNet v1 [8], vKITTI [10] demonstrate that the proposed superpoint network (dubbed "SPNet") outperforms other superpoint generation algorithms. Thanks to the end-to-end manner, the inference speed of our method is also faster than SSP [21]. Moreover, with the learned superpoints, we can further improve the performance of point cloud semantic segmentation while reducing the inference time.

The contributions of this paper are as follows:

- To the best of our knowledge, our deep iterative clustering network is the first end-to-end network for superpoint generation.
- We jointly learn the adaptive bilateral weights from the coordinate and feature spaces of point clouds to construct the point-superpoint association map.
- We formulate a label consistency loss to train our network for superpoint generation.
- We demonstrate that the end-to-end learned superpoints can further improve the performance of point cloud semantic segmentation.

2. Related Work

Deep learning on point clouds. Recent efforts have been made on deep representation learning on point clouds. Deep learning methods on point clouds can be roughly categorized into four classes: point based [44, 52, 15, 16], graph based [40, 3, 22, 25], multi-view based [42, 27] and voxelization based methods [7, 31]. As a pioneer work, PointNet [35] uses the multi-layer perceptron (MLP) and max pooling operation to extract features of point clouds. In order to characterize the local geometric structures of point clouds, Qi *et al.* [37] proposed a hierarchical feature learning framework called PointNet++, which applies MLP to the local neighborhoods to learn local features of point clouds. In PointCNN [28], $\mathcal{X}$-transformation is proposed to simultaneously weight and permute the input features so that the convolution operation can be applied to the transformed features. The graph based methods [39, 6, 5, 50] represent point clouds as a graph by constructing the spatial neighborhoods of point clouds for local feature extraction. For example, in dynamic graph CNN (DGCNN) [46], an EdgeConv operation is proposed, which acts on the graph to dynamically aggregate local geometric features of point clouds. Wang *et al.* [45] proposed graph attention convolution (GACNet), which uses a graph attention module to adaptively learn structured features of point clouds.

The voxelization based 3D deep learning methods [32, 36, 48] represent irregular and unordered point clouds with regularly volumetric occupancy grids so that 3D convolution neural networks (3D CNNs) can be used to extract features. However, volumetric representation usually leads to the burden of computational resources due to large amount of voxels. In order to reduce the consuming resource of 3D CNNs, Kd-Net [20] and OctNet [38] are proposed to focus on the informative voxels of point clouds instead of the empty voxels. Also, sparse 3D CNN [12] is proposed to speed up the standard 3D CNN, which applies 3D convolution to the set of informative 3D voxels rather than empty voxels. In addition to voxelization based 3D deep learning methods, multi-view based 3D deep learning methods [42, 47] project point clouds into a set of 2D images rendered from multiple views and use the 2D convolution operation to extract features of point clouds. Nonetheless, it is difficult to discriminatively characterize the local geometric structures of 3D objects with the rendered images across multiple views.

Point Cloud Oversegmentation. The goal of point cloud oversegmentation is to segment point clouds into superpoints. Similar to superpixels in 2D images, superpoints are a set of 3D points, which are compactly distributed in the 3D space with geometrically similar structures. In [33], voxel cloud connectivity segmentation (VCCS) is proposed for superpoint generation with a variant of k-means clustering, where the spatial connectivity and geometric features of point clouds are used so that superpoints can accurately conform to object boundaries. However, it is sensitive to the seeding initialization. To solve this problem, Gao *et al.* [11] proposed a new saliency-guided method for generating superpoints, which applies saliency-guided seeding rather seeding such initialization. Besides, Lin *et al.* [29] formulated superpoint oversegmentation as a

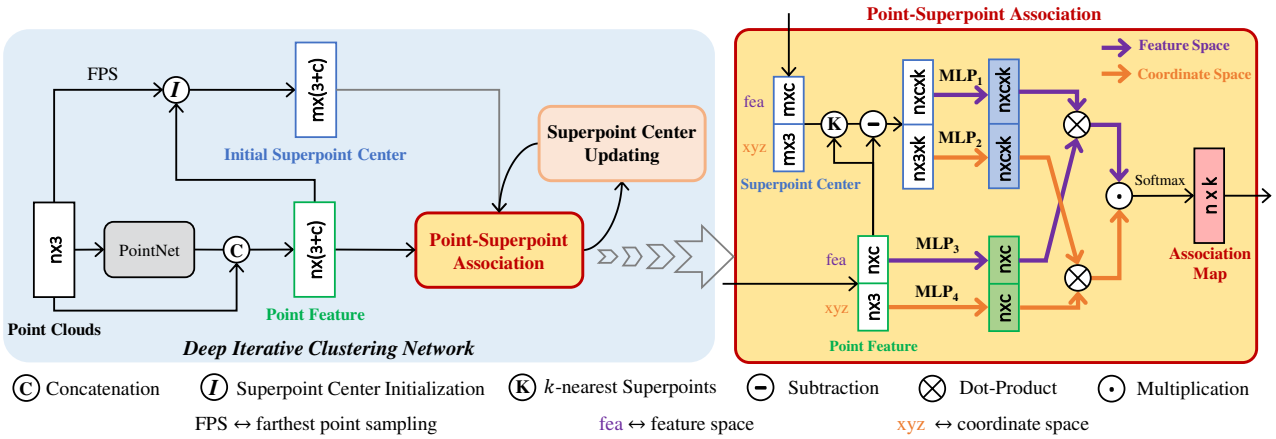


Figure 1: **Left:** Overview of the proposed deep iterative clustering network (SPNet). Given point clouds, we first use a PointNet-like structure to extract deep features. Then we use the farthest point sampling (FPS) algorithm to sample initial superpoint centers. After that, we learn the point-superpoint association from both the coordinate and feature spaces. Next, we update the superpoint center by weighting point coordinates and features, respectively. By iteratively updating the point-superpoint association map and superpoint centers, the network can gradually learn an accurate point-superpoint association map. **Right:** The architecture of the point-superpoint association module. The association map is jointly learned from the coordinate and feature spaces.

subset selection problem that can be solved with a heuristic optimization method. Based on the graph structure, Song *et al.* [41] proposed a graph-structured method, which first detects boundaries by analyzing the consecutive points and then clusters the remaining points after excluding the boundary points. Likewise, *Guinard et al.* [14] casted point cloud oversegmentation as a structured optimization problem and used the greedy graph-cut algorithm [23] to generate superpoints. In point cloud local variation (PCLV) [2], the 2D local variation (LV) graph-based over-segmentation algorithm is extended to 3D point clouds to generate superpoints. Lately, the supervised superpoint (SSP) [21] proposes a graph structure based deep metric learning method to oversegment point clouds with the deep embeddings of point clouds, where the optimization-based method in [14] is used to generate superpoints.

3. Method

3.1. Deep Iterative Clustering Network

Given the point cloud $\boldsymbol{P} = \{\boldsymbol{p}_i \in \mathbb{R}^3 \mid i = 1, \ldots, n\}$ with n points, the task of superpoint generation is to assign each point to one of m superpoint centers with the highest probability. Therefore, we can construct a point-superpoint association map $\boldsymbol{H} \in \mathbb{Z}^{n \times m}$ between the points and superpoint centers to obtain the matching scores. Nonetheless, in practice, in order to make the construction of the association map efficient in terms of computation and memory, we only compute the association between each point and its k-nearest superpoints in the coordinate space. In this way, we reformulate the association map as $\boldsymbol{H} \in \mathbb{Z}^{n \times k}$.

In order to obtain an accurate association map between the points and its k-nearest superpoint centers, we develop a deep iterative clustering network to iteratively update the association map and superpoint centers. As shown in Fig. 1, we first adopt a PointNet-like network proposed in [21] to encode the local geometric features of the original point clouds into the feature space, denoted by $\boldsymbol{F} = \{\boldsymbol{f}_i \in \mathbb{R}^c \mid i = 1, \ldots, n\}$, where c is the feature dimension. Before clustering, we employ the farthest point sampling (FPS) algorithm in the coordinate space of point clouds to obtain the initial superpoint coordinates $\boldsymbol{\mathcal{X}}^0 \in \mathbb{R}^{m \times 3}$ of the superpoint centers. In addition, the initial superpoint feature $\boldsymbol{\mathcal{S}}^0 \in \mathbb{R}^{m \times c}$ of the superpoint center is computed by averaging the corresponding point features surrounding the superpoint in the coordinate space. It is noted that different sampling methods such as inverse density important sampling (IDIS) [13] and minimum density sampling (MDS) [30] can also be used. Nonetheless, in order to balance the speed and uniformity of the sampled superpoint centers, we adopt the FPS algorithm to generate initial superpoint centers from point clouds.

After initialization, we alternately update point-superpoint association map and superpoint centers by adaptively learning the bilateral weights between the points and superpoint centers from the coordinate and feature spaces. After v iterations, we obtain the final association map $\boldsymbol{G}^v \in \mathbb{R}^{n \times k}$, which can more accurately group the points into the corresponding superpoints. Finally, to obtain superpoints from the point clouds, we can convert the soft association map $\boldsymbol{G}^v \in \mathbb{R}^{n \times k}$ to the hard association map $\boldsymbol{H}^v \in \mathbb{R}^{n \times 1}$ by assigning each point to one of the k-nearest superpoints with the highest probability.

Point-Superpoint association. For the i-th point, we first compute its k-nearest superpoint centers in the coordinate space, denoted by $\mathcal{N}_i = \{\boldsymbol{\mathcal{C}}_1, \ldots, \boldsymbol{\mathcal{C}}_k\}$, where $\boldsymbol{\mathcal{C}}_k \in \mathbb{R}^3$ indicates the spatial coordinate of the superpoint center. We then adaptively learn the bilateral weights to construct the association map between the points and superpoint centers. For each point, by assigning different weights to the different superpoint centers, we can discriminatively capture the difference of the geometric structures between the point and superpoint centers. Furthermore, due to the complex geometric structures of point clouds, the fact that the points are close in the feature space does not mean that they are close in the coordinate space. Therefore, in order to keep the spatial connectivity of the superpoint across the points, we learn the bilateral weights from the coordinate and feature spaces of point clouds simultaneously.

Formally, at the t-th iteration, the association between the i-th point and the j-th superpoint center is given by:

$$\boldsymbol{G}_{ij}^t = \phi^\top(\boldsymbol{p}_i, \boldsymbol{x}_j) g(\boldsymbol{p}_i) \cdot \varphi^\top(\boldsymbol{f}_i, \boldsymbol{s}_j) h(\boldsymbol{f}_i) \tag{1}$$

where $\boldsymbol{x}_j \in \mathbb{R}^3$ is the spatial coordinate of the superpoint center and $\boldsymbol{s}_j \in \mathbb{R}^c$ is the feature of the superpoint center. $\phi(\cdot,\cdot) : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^c$ and $\varphi(\cdot,\cdot) : \mathbb{R}^c \times \mathbb{R}^c \to \mathbb{R}^c$ are two mapping functions in the coordinate and feature spaces, respectively. Besides, $g(\cdot) : \mathbb{R}^3 \to \mathbb{R}^c$ and $h(\cdot) : \mathbb{R}^c \to \mathbb{R}^c$ are unary mapping functions implemented by the MLP. The mapping functions $\phi(\boldsymbol{p}_i, \boldsymbol{x}_j)$ and $\varphi(\boldsymbol{f}_i, \boldsymbol{s}_j)$ are defined as:

$$\begin{aligned} \phi(\boldsymbol{p}_i, \boldsymbol{x}_j) &= \mathrm{ReLU}(\boldsymbol{W}_\phi^\top(\boldsymbol{p}_i - \boldsymbol{x}_j)), \\ \varphi(\boldsymbol{f}_i, \boldsymbol{s}_j) &= \mathrm{ReLU}(\boldsymbol{W}_\varphi^\top(\boldsymbol{f}_i - \boldsymbol{s}_j)) \end{aligned} \tag{2}$$

where $\boldsymbol{W}_\phi \in \mathbb{R}^{3\times c}$ and $\boldsymbol{W}_\varphi \in \mathbb{R}^{c\times c}$ are the weights to be learned, and ReLU is the activation function. Vectors $\boldsymbol{p}_i - \boldsymbol{x}_j$ and $\boldsymbol{f}_i - \boldsymbol{s}_j$ can encode the differences between the i-th point and the j-th superpoint center in the coordinate and feature spaces, respectively. It is noted that in Eq. (1), the dot-product similarity is adopted to compute the association. To obtain the probability to assign the i-th point to the j-th superpoint, we employ the softmax funtion to obtain the normalized point-superpoint association map $\hat{\boldsymbol{G}}^t \in \mathbb{R}^{n\times k}$ across the neighborhood $\mathcal{N}_i$ as follows:

$$\hat{G}_{ij}^t = \frac{\exp(G_{ij}^t)}{\sum_{l=1}^k \exp(G_{il}^t)}. \tag{3}$$

Superpoint center updating. After computing the point-superpoint association map $\hat{\boldsymbol{G}}^t \in \mathbb{R}^{n\times k}$ at the iteration t, we in turn update the superpoint centers for the iteration t+1. Specifically, we update the coordinates and feature of the superpoint center by weighting the coordinates and features of the corresponding points with the learned association map $\hat{\boldsymbol{G}}^t$, respectively. Given the feature $\boldsymbol{F} = \{\boldsymbol{f}_i \in \mathbb{R}^c \mid i = 1, \ldots, n\}$ of point clouds, the new feature of the j-th superpoint center is defined as follows:

$$\boldsymbol{\mathcal{S}}_j^{t+1} = \frac{1}{\mathcal{G}} \sum\nolimits_{i=1}^n [j \in \mathcal{N}_i] \hat{G}_{ij}^t \boldsymbol{f}_i \tag{4}$$

where the indicator function $[j \in \mathcal{N}_i]$ indicates whether the j-th superpoint is located in the k-nearest superpoint set $\mathcal{N}_i$ of the i-th point. Here, $[j \in \mathcal{N}_i]$ equals to 1 if the j-th superpoint belongs to $\mathcal{N}_i$, and 0, otherwise. Note that since the updated superpoint center is used for the t+1 iteration, it is denoted by $\boldsymbol{\mathcal{S}}_j^{t+1} \in \mathbb{R}^c$. Besides, $\mathcal{G} = \sum_{j=i}^n [j \in \mathcal{N}_i] \hat{G}_{ij}^t$ is the normalization factor.

Similarly, given the coordinates $\boldsymbol{P} = \{\boldsymbol{p}_i \in \mathbb{R}^3 \mid i = 1, \ldots, n\}$ of point clouds, we can obtain the new coordinates of the j-th superpoint center as follows:

$$\boldsymbol{\mathcal{X}}_j^{t+1} = \frac{1}{\mathcal{G}} \sum\nolimits_{i=1}^n [j \in \mathcal{N}_i] \hat{G}_{ij}^t \boldsymbol{p}_i \tag{5}$$

where $\boldsymbol{\mathcal{X}}_j^{t+1} \in \mathbb{R}^3$ and $\boldsymbol{p}_i \in \mathbb{R}^3$ is the spatial coordinates of the point. Finally, we can obtain the new superpoint centers with $\boldsymbol{\mathcal{X}}^{t+1} \in \mathbb{R}^{m\times 3}$ and $\boldsymbol{\mathcal{S}}^{t+1} \in \mathbb{R}^{m\times c}$ at the iteration t+1. The scheme of our deep iterative clustering network is outlined in Algorithm 1.

Algorithm 1 Deep Iterative Clustering Network

Input: Point cloud $\underset{n\times 3}{\boldsymbol{P}}$.

Output: Point-Superpoint association $\underset{n\times k}{\boldsymbol{G}}$.

1: Point features, $\underset{n\times c}{\boldsymbol{F}} = \mathrm{Network}(\boldsymbol{P})$.
2: Initial superpoint centers with farthest point sampling, $\underset{m\times c}{\boldsymbol{\mathcal{S}}^0}, \underset{m\times 3}{\boldsymbol{\mathcal{X}}^0} = \mathcal{J}(\mathrm{FPS}(P))$.
3: **for** each iteration t from 1 to v **do**
4: Compute association between the point i and the k-nearest superpoint j:
$G_{ij}^t = \phi^\top(\boldsymbol{p}_i, \boldsymbol{x}_j) f(\boldsymbol{p}_i) \cdot \varphi^\top(\boldsymbol{f}_i, \boldsymbol{s}_j) g(\boldsymbol{f}_i)$.
5: Compute soft association:
$\hat{G}_{ij}^t = \frac{\exp(G_{ij}^t)}{\sum_{l=1}^k \exp(G_{il}^t)}$.
6: Compute new superpoint centers:
$\boldsymbol{\mathcal{S}}_j^{t+1} = \frac{1}{\mathcal{G}} \sum_{i=1}^n [j \in \mathcal{N}_i] \hat{G}_{ij}^t \boldsymbol{f}_i$,
$\boldsymbol{\mathcal{X}}_j^{t+1} = \frac{1}{\mathcal{G}} \sum_{i=1}^n [j \in \mathcal{N}_i] \hat{G}_{ij}^t \boldsymbol{p}_i$,
$\mathcal{G} = \sum_{i=1}^n [j \in \mathcal{N}_i] \hat{G}_{ij}^t$.
7: **end for**
8: Compute the hard association $\underset{n\times 1}{\boldsymbol{H}^v}$:
$H_i^v = \underset{j\in[1,2,\ldots,k]}{\arg\max}\ \hat{G}_{ij}^v$

3.2. Loss Function

In order to train our deep iterative clustering network to generate high-quality superpoints, we formulate a label consistency loss and a compactness loss on the points and the superpoint centers.

Label consistency loss. It is expected that the points in the same superpoint should have the same label. Given the label vector $\boldsymbol{E} = \{\boldsymbol{e}_i \in \mathbb{R}^l \mid i = 1, \ldots, n\}$ of the point clouds, where $\boldsymbol{e}_i$ is the one-hot vector, we can generate

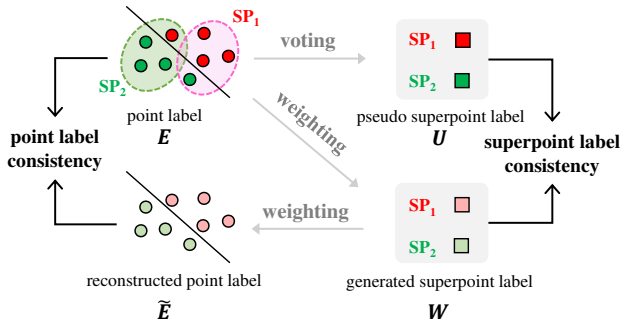


Figure 2: The overview of the label consistency loss. The circle represents the point label and the square represents the superpoint label. SP_1 and SP_2 are two superpoints.

the label vectors $\boldsymbol{W} = \{\boldsymbol{w}_j \in \mathbb{R}^l \mid j = 1, \ldots, m\}$ of m superpoint centers by weighting the label vectors of the points with the point-association map $\hat{\boldsymbol{G}}^v \in \mathbb{R}^{n \times k}$ as follows:

$$\boldsymbol{w}_j = \frac{1}{\mathcal{G}} \sum\nolimits_{i=1}^{n} [j \in \mathcal{N}_i] \hat{G}_{ij}^v \boldsymbol{e}_i \tag{6}$$

where $\mathcal{N}_i$ is the k-nearest superpoints set, and $[j \in \mathcal{N}_i]$ is the indicator function that indicates whether the j-th superpoint is located in $\mathcal{N}_i$, $\mathcal{G} = \sum_{j=i}^{n} [j \in \mathcal{N}_i] \hat{G}_{ij}^v$ is the normalization factor.

Then, the generated label vectors $\boldsymbol{W}$ of the superpoint centers are mapped back onto point clouds through the association map $\hat{\boldsymbol{G}}^v$. In other words, we reconstruct the label vectors of the points $\tilde{\boldsymbol{E}} = \{\tilde{\boldsymbol{e}}_i \in \mathbb{R}^l \mid i = 1, \ldots, n\}$ by weighting the label vectors of the k-nearest superpoints with the corresponding probability in $\hat{\boldsymbol{G}}^v$, defined as follows:

$$\tilde{\boldsymbol{e}}_i = \sum\nolimits_{j=1}^{k} \hat{G}_{ij}^v \boldsymbol{w}_j \tag{7}$$

Furthermore, we generate the pseudo label vectors of the superpoint centers by directly taking votes on the label vectors of the points in the same superpoint, denoted by $\boldsymbol{U} = \{\boldsymbol{u}_j \in \mathbb{R}^l \mid j = 1, \ldots, m\}$. Based on the pseudo label vectors of the superpoint centers, we propose a label consistency loss to train our deep iterative clustering network. Fig. 2 illustrates the label consistency loss. The label consistency loss encourages the pseudo label vectors of the superpoints ($\boldsymbol{U}$) to be consistent to the generated label vectors of the superpoints ($\boldsymbol{W}$). Also, the reconstructed label vectors of the points ($\tilde{\boldsymbol{E}}$) should be consistent with the original label vectors of the points ($\boldsymbol{E}$). Mathematically, the label consistency loss is defined as:

$$L_{cons} = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_{loss}(\boldsymbol{e}_i, \tilde{\boldsymbol{e}}_i) + \frac{1}{m} \sum_{j=1}^{m} \mathcal{L}_{loss}(\boldsymbol{w}_j, \boldsymbol{u}_j) \tag{8}$$

where $\mathcal{L}_{loss}$ is the loss function and we choose the cross-entropy loss in the experiment.

Compactness loss. We define the compactness loss to encourage the superpoints to be spatially compact. It is expected that the points belonging to the same superpoint should be close to the superpoint center in the coordinate space. We encourage the compactness of the superpoints by minimizing the distances between the points and the superpoint centers. Specifically, given point clouds $\boldsymbol{P} \in \mathbb{R}^{n \times 3}$, we can obtain the coordinates of the superpoint centers $\boldsymbol{Q} \in \mathbb{R}^3$. The compactness loss of our superpoint generation network is defined as:

$$L_{compact} = \|\boldsymbol{P} - \boldsymbol{Q}\|_F \tag{9}$$

where $||\cdot||_F$ indicates the Frobenius distance.

In this paper, the final loss is a combination of the label consistency loss and the compactness loss, $L = L_{cons} + \lambda L_{compact}$, where we empirically set $\lambda = 10^{-3}$ in all experiments.

4. Experiments

4.1. Datasets

We evaluate our method on both indoor and outdoor datasets. The indoor datasets include S3DIS [1] and ScanNet v1 [8]. S3DIS is a dense large-scale dataset, which contains 3D RGB point clouds (about 273 million points) with 13 categories from six indoor areas. For a fair comparison, we follow [35, 26] to choose Area 5 of the S3DIS dataset as our testing set and the remaining as our training set. Compared with S3DIS, ScanNet v1 is a sparser large-scale dataset, which contains 3D RGB point clouds (about 242 million points) with 21 categories. Following [37], we evaluate our method on the offline test dataset. For the outdoor dataset, we conduct experiments on the sparse large-scale vKITTI [10] dataset, which contains RGB point clouds (about 15 million points) with 13 categories from six sequences. Following [26, 21], we report the evaluation results of 6-fold cross validation on the vKITTI dataset.

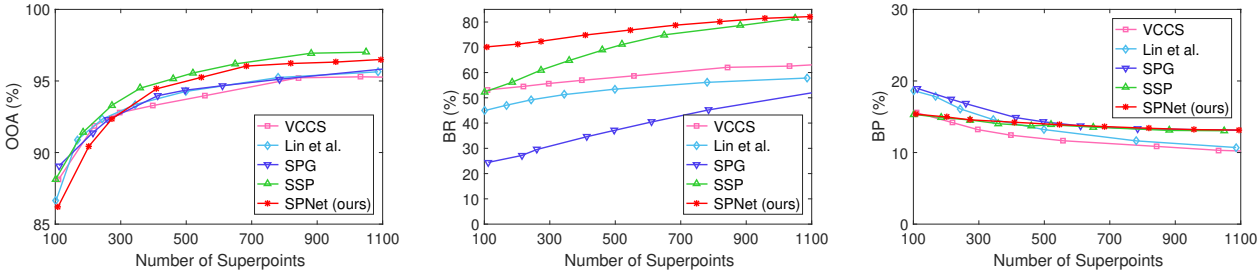
4.2. Implementation Details

Our model is implemented with the PyTorch [34] deep learning platform. For all experiments, we use Adam [19] with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We train our model for 1000 epochs. The initial learning rate is 0.001 and is divided by 10 for every 300 epochs.

For a fair comparison, we adopt the same strategy as in SSP [21] to process the indoor and outdoor point clouds. For all datasets, we subsample point clouds using a regular grid of voxels (voxel width of 3cm for S3DIS and ScanNet v1, 5cm for vKITTI). In each voxel, we average the position and color of the contained points. For training, we randomly select a subregion (15,000 voxels for S3DIS and vKITTI, 8,000 voxels for ScanNet v1) of each room or scene. We generate subregions by using breadth-first search (BFS) on the voxels' adjacency graph. In practice, we observe that when selecting 0.8% of the voxels in the subregion as the initial seed points (superpoint centers), the model reaches

Method	S3DIS Area 5				ScanNet v1				vKITTI 6-fold			
	OOA	BR	BP	F1	OOA	BR	BP	F1	OOA	BR	BP	F1
w/ RGB												
VCCS [33]	95.22	62.06	10.86	18.48	74.59	45.65	7.09	12.27	50.05	64.75	13.19	21.91
SPG [26]	95.84	52.06	13.11	20.94	95.02	42.38	10.73	17.12	91.92	75.13	24.66	37.13
SSP [21]	**97.04**	80.73	13.02	22.42	96.39	78.53	**14.07**	**23.86**	92.85	90.37	**26.18**	**40.59**
SPNet (**ours**)	96.50	**82.11**	**13.14**	**22.65**	**96.47**	**79.10**	13.58	23.18	**94.37**	**94.15**	24.88	39.36
w/o RGB												
VCCS [33]	94.23	50.09	10.34	17.14	74.53	44.62	7.01	12.11	86.23	69.12	10.44	18.14
Lin *et al.* [29]	95.24	56.12	11.63	19.26	96.56	65.31	12.25	20.63	91.90	80.86	15.39	25.85
SPG [26]	93.86	40.98	12.25	18.86	95.10	42.55	10.76	17.17	90.12	72.34	**27.40**	39.74
SSP [21]	96.47	74.64	**13.09**	22.27	96.42	78.14	**14.16**	**23.97**	85.17	70.16	20.71	31.98
SPNet (**ours**)	**96.55**	**78.77**	13.06	**22.40**	**96.52**	**78.42**	13.43	22.93	**92.16**	**86.34**	26.29	**40.30**

Table 1: Comparison results of generated superpoints on the S3DIS, ScanNet v1, and vKITTI datasets.



(a) Oracle Overall Accuracy (OOA) (b) Boundary Recall (BR) (c) Boundary Precision (BP)

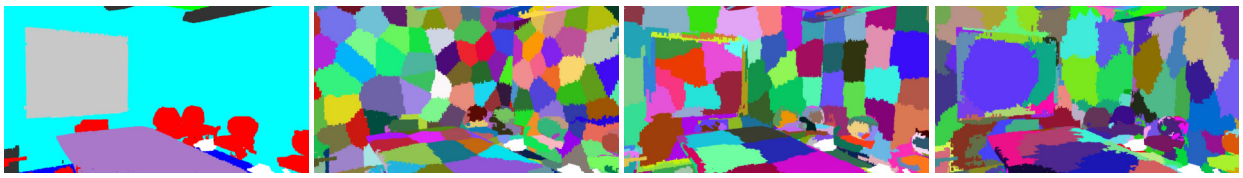
Figure 3: Metric results of different methods in the cases of different numbers of superpoints on Area 5 of the S3DIS dataset.

the best performance. Note that superpoint evaluation results are obtained at the voxel level, while all results on the semantic segmentation are obtained at the point level.

4.3. Learned Superpoints

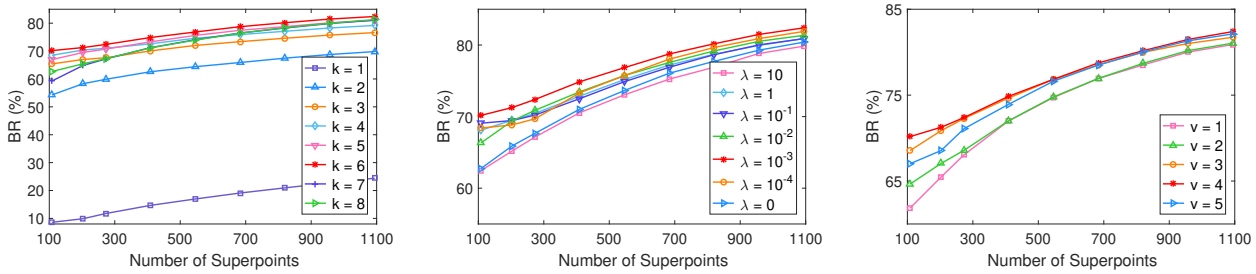
Evaluation metrics. We adopt the same metrics as in SSP [21] to evaluate the proposed method. Specifically, we use the Oracle Overall Accuracy (OOA), Boundary Recall (BR), Boundary Precision (BP) to evaluate the quality of the superpoints. OOA measures the highest accuracy achievable for semantic segmentation that utilizes the superpoints as units, whereas BR and BP measure how well the superpoint boundaries align with the ground truth boundaries. Note that for the definition of point clouds boundaries and the calculation of OOA, BR, and BP, please refer to [21]. In addition, we also report the F1 score: F1 = 2BP·BR/(BP+BR).

Quantitative results. We evaluate our method on three popular datasets including S3DIS [1], ScanNet v1 [8], and vKITTI [10]. The quantitative results are shown in Tab. 1. For a fair comparison, the number of superpoints generated by different methods on the same dataset is almost the same. For example, the number of superpoints on Area 5 of the S3DIS dataset is 1042 (VCCS), 1059 (Lin *et al.*), 1053 (SPG), 1048 (SSP), and 1050 (ours), respectively. From the table, it can be observed that our method can achieve better performance than the other methods with or without color information. Compared with the handcrafted features in the traditional methods [33, 29], the deep features in our method learned from the specifically designed network can better characterize the complex local geometric structures of point clouds. Therefore, the performance of our method surpasses the performance of these traditional methods [33, 29]. Different from SSP, we use the label consistency loss to optimize the network during training, so that the generated superpoints adhere to the boundaries of objects. Thus, on the metric of BR, our method can achieve better performance than SSP. From experimental results, one can see that our method tends to cluster multiple superpoints inside an object. The boundaries of those superpoints are regarded as false boundaries when calculating the BP. Therefore, the BP of our method is a little bit low. Especially, in the outdoor vKITTI dataset, our method clusters multiple superpoints on the road, while SPG and SSP divide the road into large-scale superpoints through the graph-cut algorithm. Therefore, the BP of our method is slightly lower on the vKITTTI dataset. However, since false boundaries of the superpoints inside an object do not generate new labels to superpoints, the false boundaries inside the object will not affect the segmentation results of the object (please refer to the segmentation results in Section 4.4). We also report the F1 scores in Tab. 1. It can be seen that our method is comparable to SSP in terms of the F1 score. As illustrated in Fig. 3, we plot the performance curves



(a) Semantic Label (b) Lin *et al.* [29] (c) SSP [21] (d) SPNet (**ours**)

Figure 4: Visual results of different methods on Area 5 of the S3DIS dataset.



(a) Hyper-parameter k (b) Hyper-parameter λ (c) Hyper-parameter v

Figure 5: Ablation study results of different hyper-parameters on Area 5 of the S3DIS dataset.

Method	type	training time	inference time
VCCS [33]	CPU	-	92s
Lin *et al.* [29]		-	187s
SPG [26]		-	995s
SSP [21]	GPU	∼6h	2327s
SPNet (**ours**)		∼7h	646s

Table 2: Training time and inference time of different methods on Area 5 of the S3DIS dataset.

of different methods in the cases of different numbers of generated superpoints on Area 5 of the S3DIS dataset. From the figure, it can be found that the performance of our method is comparable with that of SSP. Moreover, with the decrease of the number of superpoints, our SPNet can still maintain the high performance for BR, while SSP drops rapidly.

Visual results. As shown in Fig. 4, we visualize the superpoints generated by different methods. From the figure, one can see that Lin *et al.* [29] cannot segment the whiteboard from the wall, while SSP [21] and our method can segment it well with clearer boundaries. In addition, the boundaries of the whiteboard generated by our method are more accurate than SSP. Although the shape of generated superpoints with the methods in [29, 21] seems more regular, the object boundaries are not clear.

Time costs. To evaluate the time costs, we compare the running time of different methods. For a fair comparison, we use a single Core i5 CPU to evaluate VCCS [33], Lin *et al.* [29], and SPG [26], respectively. For learning-based methods, both SSP and our method are run on a single NVIDIA TITAN RTX GPU using the PyTorch framework. We evaluate the running time of generating superpoints on Area 5 of S3DIS dataset. Besides, we also compute the training time of our method and SSP. As shown in Tab. 2, it can be seen that our method runs faster than SSP. Since our method is an end-to-end method, we can generate superpoints directly from the network without any post-processing. However, SSP first uses the network to extract the embedding of points and then uses the optimization-based method to generate superpoints. Thus, SSP spends more time than our method. Besides, our method and SSP have similar training time. Although the time cost of our method is higher than the traditional methods [33, 29], our method can achieve better performance on all three datasets as shown in Tab. 1.

Method	OOA	BR	BP
only coordinate space	95.10	78.35	9.43
only feature space	95.43	82.03	12.01
both (SPNet)	**96.50**	**82.11**	**13.14**

Table 3: Ablation study results of different settings on Area 5 of S3DIS dataset.

Ablation studies. In Fig. 5 and Tab. 3, we show the ablation study results to experimentally verify the rationality of the settings in our experiments. Specifically, in Fig. 5, we study the impact of the hyper-parameters λ, k, and v on performance of our network. For a fair comparison, we only adjust the corresponding parameters and fix the remaining parameters unchanged during training. As shown in Fig. 5, we display the curves of boundary recall (BR) on Area 5 of the S3DIS dataset at different numbers of superpoints. It can be observed that we can achieve the best performance when setting k=6, λ=10^{-3}, and v=4, respectively. According to Fig. 5, hyper-parameters λ, v, and k are insensitive to superpoint generation. Except for k=1, the performance will not change greatly in the cases of different values of k.

Moreover, we also conduct experiments to verify the effectiveness of using both the coordinate and feature spaces in the point-superpoint association learning. As shown in Tab. 3, we report the performance of OOA, BR, and BP on Area 5 of the S3DIS dataset. From the table, it can be found that learning the bilateral weights from the coordinate and feature spaces simultaneously can improve the performance of superpoint generation.

4.4. Semantic Segmentation

Experimental settings. We present results on the semantic segmentation benchmarks of S3DIS [1], ScanNet v1 [8], and vKITTI [10]. For a fair comparison, we adopt the same network as in [26, 21] to conduct experiments with our generated superpoints. We use the same settings as in [26, 21] to train the network. To evaluate the segmentation results, we use the following metrics: mean per-class intersection-over-union (mIoU), mean per-class accuracy (mAcc), overall accuracy (OA).

Experimental results. As shown in Tab. 4, we report the segmentation results on the three datasets. In the table, we divide the methods into two types: point-based methods and superpoint-based methods. Note that SPG [26], SSP [21], and our method are superpoint-based methods and the rest are point-based methods. Moreover, we adopt the same segmentation method and the same experimental settings as those in SPG and SSP. It can be seen that our method can obtain better performance than SPG and SSP on all three dataset benefiting from the good superpoint partition results. The quantitative results have demonstrated that the generated superpoints can effectively improve the performance of semantic segmentation. In addition, compared to the point-based methods, our method also outperforms most of them. It is noted that some point-based methods [45, 44] have achieved the state-of-the-art due to the advanced data processing and vote strategy. Since our superpoint generation method does not employ this kind of advanced data processing, our segmentation results are slightly lower than KPConv [44].

In Tab. 4, we also report the inference time of different methods during testing. Note that for superpoint-based methods, we report the total time of superpoint generation (colored in blue) and semantic segmentation (colored in red). It can be found that our method runs faster than other methods. In particular, based on the superpoints, our method can even run faster than PointNet [35]. Specifically, in our method, a simplified PointNet (3-layer MLP plus spatial transformer network (STN)) is used, while in PointNet 5-layer MLP plus coordinate and feature STNs are used for segmentation. Since our network is shallow and the number of superpoints is far less than points, our method is faster than PointNet. The time consumption of our method on the S3DIS 6-fold is 2828s (superpoint generation 2588s + simplified PointNet 132s + superpoint graph 108s), which is lower than PointNet (6050s). Therefore, it further demonstrates the advantages of our method on the semantic segmentation task.

Method	inference time	mIoU	mAcc	OA
S3DIS 6-fold				
PointNet [35]	6050	47.6	66.2	78.5
PointCNN [28]	12675	65.3	75.6	**88.1**
ShellNet [51]	-	66.8	-	87.1
KPConv [44]	10136	**70.6**	79.1	-
PointWeb [52]	11982	66.7	76.2	87.3
RandLA-Net [15]	-	70.0	**82.0**	88.0
SPG [26]	4243 (3999+244)	62.1	73.0	85.5
SSP [21]	9581 (9325+256)	68.4	78.3	87.9
SPNet (**ours**)	2828 (2588+240)	68.7	79.7	88.0
ScanNet v1				
PointNet [35]	430	14.7	19.9	-
PointNet++ [37]	705	34.2	43.8	-
RSNet [17]	-	39.3	48.3	-
TangConv [43]	452	40.9	**55.1**	80.1
SPG [26]	1873 (1733+140)	41.7	53.2	81.6
SSP [21]	2011 (1866+145)	41.8	52.3	82.6
SPNet (**ours**)	362 (220+142)	**43.2**	54.6	**82.7**
vKITTI 6-fold				
PointNet [35]	3948	34.4	47.0	79.7
3P-RNN [49]	-	41.6	54.1	87.8
G+RCU [9]	-	35.6	57.6	79.7
SPG [26]	5054 (5004+50)	55.4	65.1	82.7
SSP [21]	6075 (6021+54)	52.0	67.3	84.3
SPNet (**ours**)	1411 (1359+52)	**57.0**	**67.8**	**89.9**

Table 4: Results of semantic segmentation on 6-fold of S3DIS, ScanNet v1, and vKITTI datasets. Here, blue numbers correspond to the time of superpoint generation, and red numbers correspond to the time of semantic segmentation.

5. Conclusion

In this paper, we developed a novel deep iterative clustering network for superpoint generation. In our network, we jointly learned a soft point-superpoint association map from both coordinate and feature spaces of point clouds. In order to obtain a more accurate point-superpoint association map, we iteratively updated the association map and superpoint centers. Finally, by predicting the pseudo labels of the superpoint centers, we formulated a label consistency loss on the points and superpoint centers to train the network. Extensive experiments on indoor and outdoor datasets demonstrate that the proposed method can not only generate high-quality superpoints from point clouds but also improve the performance of semantic segmentation.

Acknowledgments

This work was supported by the National Science Fund of China (Grant Nos. U1713208, 61876084).

References

[1] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3D semantic parsing of large-scale indoor spaces. In *CVPR*, 2016.

[2] Yizhak Ben-Shabat, Tamar Avraham, Michael Lindenbaum, and Anath Fischer. Graph based over-segmentation methods for 3D point clouds. *Computer Vision and Image Understanding*, 174:12–23, 2018.

[3] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[4] Jie Chen and Baoquan Chen. Architectural modeling from sparsely scanned range data. *International Journal of Computer Vision*, 78(2-3):223–236, 2008.

[5] Mingmei Cheng, Le Hui, Jin Xie, and Jian Yang. SSPC-Net: Semi-supervised semantic 3D point cloud segmentation network. In *AAAI*, 2021.

[6] Mingmei Cheng, Le Hui, Jin Xie, Jian Yang, and Hui Kong. Cascaded non-local neural network for point cloud semantic segmentation. In *IROS*, 2020.

[7] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, 2019.

[8] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *CVPR*, 2017.

[9] Francis Engelmann, Theodora Kontogianni, Alexander Hermans, and Bastian Leibe. Exploring spatial context for 3D semantic segmentation of point clouds. In *ICCV*, 2017.

[10] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *CVPR*, 2016.

[11] Ge Gao, Mikko Lauri, Jianwei Zhang, and Simone Frintrop. Saliency-guided adaptive seeding for supervoxel segmentation. *ISPRS*, 2017.

[12] Ben Graham. Sparse 3D convolutional neural networks. *arXiv preprint arXiv:1505.02890*, 2015.

[13] Fabian Groh, Patrick Wieschollek, and Hendrik Lensch. Flex-convolution (million-scale point-cloud learning beyond grid-worlds). *arXiv preprint arXiv:1803.07289*, 2018.

[14] Stéphane Guinard and Loic Landrieu. Weakly supervised segmentation-aided classification of urban scenes from 3D lidar point clouds. *ISPRS Workshop*, 2017.

[15] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. RandLA-Net: Efficient semantic segmentation of large-scale point clouds. *arXiv preprint arXiv:1911.11236*, 2019.

[16] Zeyu Hu, Mingmin Zhen, Xuyang Bai, Hongbo Fu, and Chiew-lan Tai. JSENet: Joint semantic segmentation and edge detection network for 3D point clouds. *arXiv preprint arXiv:2007.06888*, 2020.

[17] Qiangui Huang, Weiyue Wang, and Ulrich Neumann. Recurrent slice networks for 3D segmentation of point clouds. In *CVPR*, 2018.

[18] Varun Jampani, Deqing Sun, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz. Superpixel samping networks. In *ECCV*, 2018.

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[20] Roman Klokov and Victor Lempitsky. Escape from cells: Deep Kd-networks for the recognition of 3D point cloud models. In *ICCV*, pages 863–872, 2017.

[21] Loic Landrieu and Mohamed Boussaha. Point cloud over-segmentation with graph-structured deep metric learning. In *CVPR*, 2019.

[22] Loic Landrieu, Clément Mallet, and Martin Weinmann. Comparison of belief propagation and graph-cut approaches for contextual classification of 3D lidar point cloud data. In *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 2768–2771. IEEE, 2017.

[23] Loic Landrieu and Guillaume Obozinski. Cut Pursuit: fast algorithms to learn piecewise constant functions. In *AISTATS*, 2016.

[24] Loic Landrieu and Guillaume Obozinski. Cut Pursuit: Fast algorithms to learn piecewise constant functions on general weighted graphs. *SIAM Journal on Imaging Sciences*, 10(4):1724–1766, 2017.

[25] Loic Landrieu, Hugo Raguet, Bruno Vallet, Clément Mallet, and Martin Weinmann. A structured regularization framework for spatially smoothing semantic labelings of 3D point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 132:102–118, 2017.

[26] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR*, 2018.

[27] Truc Le, Giang Bui, and Ye Duan. A multi-view recurrent neural network for 3D mesh segmentation. *Computers & Graphics*, 66:103–112, 2017.

[28] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. PointCNN: Convolution on $\mathcal{X}$-transformed points. In *NeurIPS*, 2018.

[29] Yangbin Lin, Cheng Wang, Dawei Zhai, Wei Li, and Jonathan Li. Toward better boundary preserved supervoxel segmentation for 3D point clouds. *ISPRS journal of photogrammetry and remote sensing*, 143:39–47, 2018.

[30] Minghua Liu, Lu Sheng, Sheng Yang, Jing Shao, and Shi-Min Hu. Morphing and sampling network for dense point cloud completion. In *AAAI*, 2020.

[31] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-Voxel cnn for efficient 3D deep learning. In *NeurIPS*, 2019.

[32] Daniel Maturana and Sebastian Scherer. VoxNet: A 3D convolutional neural network for real-time object recognition. In *IROS*, 2015.

[33] Jeremie Papon, Alexey Abramov, Markus Schoeler, and Florentin Worgotter. Voxel cloud connectivity segmentation-supervoxels for point clouds. In *CVPR*, 2013.

[34] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[35] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, 2017.

[36] Charles R Qi, Hao Su, Matthias Niebner, Angela Dai, Mengyuan Yan, and Leonidas J Guibas. Volumetric and multi-view cnns for object classification on 3D data. In *CVPR*, 2016.

[37] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017.

[38] Gernot Riegler, Ali Ulusoy, and Andreas Geiger. OctNet: Learning deep 3D representations at high resolutions. In *CVPR*, 2017.

[39] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *CVPR*, 2018.

[40] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *CVPR*, 2017.

[41] Soohwan Song, Honggu Lee, and Sungho Jo. Boundary-enhanced supervoxel segmentation for sparse outdoor LiDAR data. *Electronics Letters*, 50(25):1917–1919, 2014.

[42] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *ICCV*, 2015.

[43] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3D. In *CVPR*, 2018.

[44] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. KPConv: Flexible and deformable convolution for point clouds. In *ICCV*, 2019.

[45] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *CVPR*, 2019.

[46] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018.

[47] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. SqueezeSegV2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a LiDAR point cloud. In *ICRA*, 2019.

[48] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *CVPR*, 2015.

[49] Xiaoqing Ye, Jiamao Li, Hexiao Huang, Liang Du, and Xiaolin Zhang. 3D recurrent neural networks with context fusion for point cloud semantic segmentation. In *ECCV*, 2018.

[50] Li Yi, Hao Su, Xingwen Guo, and Leonidas J Guibas. Syncspeccnn: Synchronized spectral cnn for 3D shape segmentation. In *CVPR*, 2017.

[51] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. ShellNet: Efficient point cloud convolutional neural networks using concentric shells statistics. In *ICCV*, 2019.

[52] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. PointWeb: Enhancing local neighborhood features for point cloud processing. In *CVPR*, 2019.