

# Bidirectional Feature Globalization for Few-shot Semantic Segmentation of 3D Point Cloud Scenes

Yongqiang Mao<sup>1,†</sup>, Zonghao Guo<sup>1,†</sup>, Xiaonan Lu<sup>1</sup>, Zhiqiang Yuan<sup>1</sup>, Haowen Guo<sup>2,\*</sup>

<sup>1</sup>University of Chinese Academy of Sciences, Beijing China

<sup>2</sup>Wuhan University, Wuhan China

mao.wingkeung@gmail.com, guozonghao19@mails.ucas.ac.cn, ghw@whu.edu.cn

## Abstract

Few-shot segmentation of point cloud remains a challenging task, as there is no effective way to convert local point cloud information to global representation, which hinders the generalization ability of point features. In this study, we propose a bidirectional feature globalization (BFG) approach, which leverages the similarity measurement between point features and prototype vectors to embed global perception to local point features in a bidirectional fashion. With point-to-prototype globalization (Po2PrG), BFG aggregates local point features to prototypes according to similarity weights from dense point features to sparse prototypes. With prototype-to-point globalization (Pr2PoG), the global perception is embedded to local point features based on similarity weights from sparse prototypes to dense point features. The sparse prototypes of each class embedded with global perception are summarized to a single prototype for few-shot 3D segmentation based on the metric learning framework. Extensive experiments on S3DIS and ScanNet demonstrate that BFG significantly outperforms the state-of-the-art methods.

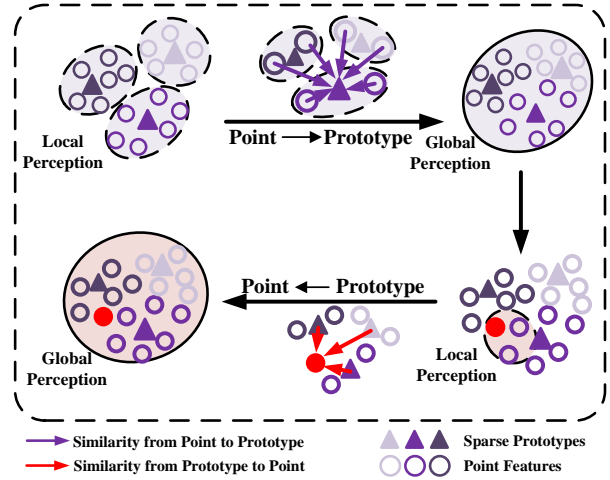


Figure 1. Overview of our bidirectional feature globalization (BFG). BFG first realizes the prototype feature globalization by integrating dense point features through similarity weights (Upper). Point feature globalization is then performed by integrating sparse prototypes through similarity weights (Lower). Equipped with BFG, both prototypes and point features are endowed with global perception, which facilitates few-shot 3D segmentation.

## 1. Introduction

Thanks to the powerful representation capabilities of Convolutional Neural Networks and the open source of numerous point cloud annotation datasets, we have witnessed unprecedented progress [18, 3, 11, 26, 30, 16, 17] in point cloud segmentation. However, annotating large-scale point cloud datasets is laborious and extensive, which hinders the application of point cloud segmentation in various scenarios.

In recent years, few-shot 3D point cloud segmentation [34] is explored. Given base classes with sufficient training data and new classes of few supervisions, this task aims to generalize the 3D representation model initialized

upon the base classes to new classes. Early researches simply imitate the few-shot segmentation methods from 2D to 3D tasks. For example, the single prototype method [5] leverages global average pooling to produce 3D semantic prototypes, which are used to classify point sets in a metric learning framework. The multi-prototype method [34] generates prototypes by aggregating point features from different object parts to improve the semantic representation.

In spite of the substantial progress, existing methods are impacted by serious false segmentation when the instance consists of complex parts. By our investigation, we realize that the false segmentation is caused by the locality of point convolution, which lacks the ability to capture global feature perception. Such global perception is crucial to pro-

\*Corresponding author. †Equal contribution.

duce correct segmentation when deformation or scale variation occur.

In this paper, we focus on designing a bidirectional feature globalization (BFG) approach (Fig. 1) to regularize the training procedure of semantic prototypes and endow each point feature and prototype the global feature perception. BFG defines a bidirectional module which uses the dense local point features to generate sparse global prototypes and then leverages the global prototypes to guide globalization of local point features. With such a bidirectional module, both point features and prototypes are endowed with global perception.

Specifically, the proposed approach is rooted in a metric learning framework, which consists of two branches (support branch and query branch), Fig. 2. The support branch is responsible for the globalization of prototypes and support point features, which goes through two modules in order: Point-to-Prototype Globalization (Po2PrG) and Prototype-to-Point Globalization (Pr2PoG). Given the sparse prototypes initialized by the sparse prototype generation (SP-Gen) module, Po2PrG and Pr2PoG perform globalization on prototypes and point features in a bidirectional fashion according to the similarity weights between sparse prototypes and dense point features. Finally, sparse prototype assembly (SPA) is carried out to obtain the optimal prototype representation of support point features for metric learning. The query branch generates the similarity maps between the prototypes from the support branch and the query point features to obtain point cloud segmentation results.

To conclude, the main contributions of our BFG are summarized as follows:

- We propose bidirectional feature globalization (BFG), defining a simple-yet-effective way to embed global perception to local point features and their prototypes in a mutual enhancement fashion.
- We design point-to-prototype globalization (Po2PrG) and prototype-to-point globalization (Pr2PoG) modules based on the similarity weights, which activate the global perception of prototypes and point features, respectively.
- By assembling sparse prototypes embedded with global perception, we achieve a new state-of-the-art performance on the popular S3DIS and ScanNet datasets.

## 2. Related Work

**Point Cloud Segmentation.** Point Cloud semantic segmentation for point-wise classification of a class of instances has been extensively studied. PointNet [18] learns the features of each point independently through MLP, and utilizes a symmetric function (such as max pooling) to

solve the disorder problem while aggregating global features. On this basis, a large number of point-based methods [19, 15, 26] have sprung up. Point-wise MLP methods [19, 11, 33, 7] employ shared MLP as the basic block of the network for feature extraction. Point Convolution methods [15, 2, 3, 28, 13, 29, 23, 14] aim to extract high-quality point features and learn local relationships by designing efficient point convolution operators. Graph-based methods [26, 25, 12] aim to learn the spatial geometric features of points through constructing graphs inside point sets and designing novel graph convolutions.

However, these methods heavily depend on large-scale training sets and are incapable of generalizing to new classes, which limits the application of many real-world scenarios. To improve the generalization capability of 3D representation models, few-shot segmentation task of point cloud has been the research focus of the community.

**Few-shot Learning.** Current methods of few-shot learning mainly concentrate on metric learning [24, 22, 32, 31, 21] and meta-learning [27, 20, 6, 9]. The methods based on metric learning mainly focus on employing distance metric to predict whether two regions belong to the same class. The main idea of meta-learning based methods is to specify an optimization procedure or loss function to gain the ability to learn faster and adapt to new classes. The effectiveness of the prototype concept for few-shot learning is demonstrated [24, 31] in various metric learning frameworks. Inspired by these, the methods of prototype learning is widely adopted in the few-shot segmentation task, which greatly reduces the computational budget while maintaining high performance.

**Few-shot 3D Point Cloud Segmentation.** Current few-shot segmentation methods [5, 34] of point cloud largely follow the metric learning framework, which learns semantics from the support point sets. Such stores the semantics in the form of prototype vectors, which are generalized to segment the query point sets. ProtoNet [5] uses a single prototype to centrally express the features of each semantic class in the support point sets. It designs a mask average pooling strategy to generate prototype vectors, and then applies a similarity measurement function to build the relationship between the prototypes and the features of query point sets. MPTI [34] introduces a method of transductive learning to predict the semantic classes of the query point sets based on the prototypes. It also extracts multiple prototypes of the support point features to better represent the rich foreground semantic.

In spite of the substantial progress, existing methods are impacted by serious false segmentation when there exists deformation and/or scale variation. By our investigation, we realize that the false segmentation is caused by the locality of point convolution, which lacks the ability to capture global feature perception. The multi-prototype method [34]

took a step to alleviate this. However, it requires the significant increase of the number of prototypes, which aggregates the computational complexity. In this paper, we propose the bidirectional feature globalization (BFG) approach, which aims to globalize features to obtain the optimal representation with a single prototype vector of each class.

### 3. Method

#### 3.1. Overview

The flowchart of our BFG approach is illustrated in Fig. 2, which uses ProtoNet [5] as the baseline. As a few-shot 3D segmentation network, BFG consists of two network branches: the support branch (upper) and the query branch (lower). The two network branches use a weight shared feature embedding network to extract point features. Let  $F$  and  $F_Q$  represent point features after passing through the embedding network of support branch and query branch, respectively. In the support branch, the prototypes are first generated by sparse prototype generation (SPGen) upon the point features  $F$  and the corresponding mask. By passing the point-to-prototype globalization (Po2PrG) and prototype-to-point globalization (Pr2PoG) modules, these prototype vectors are endowed with global feature perception. A sparse prototype assembly (SPA) module is applied to aggregate the prototypes for semantic representation. In the query branch, similarity maps between the extracted features  $F_Q$  of query point sets and the prototypes extracted by the support branch are calculated by the distance function (cosine distance or squared Euclidean distance). Such similarity maps are directly used to produce semantic segmentation results. In the query branch, the network is driven by the cross-entropy loss  $\mathcal{L}_{CE}$ , as  $\mathcal{L}_{total} = \mathcal{L}_{CE}$ .

In what follows, we first introduce the SPGen module and then present the feature globalization procedure with Po2PrG and Pr2PoG modules.

#### 3.2. Sparse Prototype Generation

Sparse prototype generation (SPGen) produces the initial representation of prototypes, Fig. 2. For each class of support point sets, we first use the mask of points to get the foreground and background points. Inspired by [10, 8], the foreground points are partitioned into multiple groups which correspond to object parts. Each part corresponds to a prototype vector. Following [34], the sparse prototypes are initialized by two steps: object part construction and prototype extraction.

*Object Part Construction.* Denote  $N$  and  $D$  as the number and the channel of point features, respectively. Given the support point feature  $F \in \mathbb{R}^{N \times D}$ , the coordinate  $J \in \mathbb{R}^{N \times 3}$ , and the mask  $M^c \in \mathbb{R}^{N \times 1}$  ( $c$  represents the class) of support point sets, the masked point feature  $\mathcal{F}^c = \{f_i^c\}_{i=1}^{m^c}$  and its coordinate  $\mathcal{J}^c = \{j_i^c\}_{i=1}^{m^c}$  ( $m^c$  represents the num-

ber of support points belonging to the class  $c$ ) are obtained through its corresponding mask  $M^c$  and implemented by keeping points of class  $c$  and culling points of other classes.

Based on the point feature  $\mathcal{F}^c$  and coordinate  $\mathcal{J}^c$ , sampling seed points and point-to-seed assignment [34] are executed sequentially. The farthest point sampling (FPS) algorithm is employed to sample a subset of  $K$  seed points which are from the same class. The seed points represent the centers of the parts. Let  $\{s_k^c\}_{k=1}^K \subset \{f_i^c\}_{i=1}^{m^c}$  denote the sampled seeds. After that, we compute the point-to-seed distance and assign point features to these part centers according to the index of the closest part center of each point.

*Prototype Extraction.* We perform global average pooling within each part to extract prototypes. Formally, the initial sparse prototype  $\mu^c$  of class  $c$  is defined as:

$$\mu^c = \left\{ \mu_1^c, \dots, \mu_K^c \mid \mu_k^c = \frac{1}{|\mathcal{I}_k^c|} \sum_{f_i^c \in \mathcal{I}_k^c} f_i^c \right\}, \quad (1)$$

$$s.t. \argmin_{\mathcal{I}^c} \sum_{k=1}^K \sum_{f_i^c \in \mathcal{I}_k^c} \|f_i^c - s_k^c\|_2$$

where the masked point features  $\mathcal{F}^c = \{f_i^c\}_{i=1}^{m^c}$  is partitioned to  $K$  sets  $\mathcal{I}^c = \{\mathcal{I}_1^c, \dots, \mathcal{I}_K^c\}$  such that  $f_i^c \in \mathcal{I}_k^c$  is assigned to  $s_k^c$ . At the same time, the coordinate of each point  $j_i^c \in \mathcal{I}_{\mathcal{J},k}^c$  is also assigned to  $s_k^c$ . In this way, we get the coordinates of the part sets as  $\mathcal{I}_{\mathcal{J}}^c = \{\mathcal{I}_{\mathcal{J},1}^c, \dots, \mathcal{I}_{\mathcal{J},K}^c\}$ . Similarly, the coordinate of each prototype is defined as:

$$\mu_{\mathcal{J}}^c = \left\{ \mu_{\mathcal{J},1}^c, \dots, \mu_{\mathcal{J},K}^c \mid \mu_{\mathcal{J},k}^c = \frac{1}{|\mathcal{I}_{\mathcal{J},k}^c|} \sum_{j_i^c \in \mathcal{I}_{\mathcal{J},k}^c} j_i^c \right\} \quad (2)$$

#### 3.3. Bidirectional Feature Globalization

Since the initialized prototypes are extracted within object parts, the prototype semantics are limited to local point features. To solve this issue, we propose point-to-prototype globalization (Po2PrG) to perform global representation of sparse prototypes, Fig. 3(left). Similarly, due to the locality of point convolution, dense point features extracted by the embedding network ignore the global semantic perception between object parts. Prototype-to-point globalization (Pr2PoG) is proposed to solve this problem, Fig. 3(right).

The Po2PrG and Pr2PoG modules leverage the semantic perception of point-to-prototype and prototype-to-point similarity, respectively. With Po2PrG and Pr2PoG, BFG embeds the global perception to both prototype vectors and point features in a bidirectional fashion.

Before introducing Po2PrG and Pr2PoG, we define the similarity measurement for the generation of similarity weights. To leverage the spatial information, the coordinate  $\mathcal{J}^c$  is introduced to the similarity measurement. Thus, the similarity between point features  $\mathcal{F}^c$  and sparse prototypes

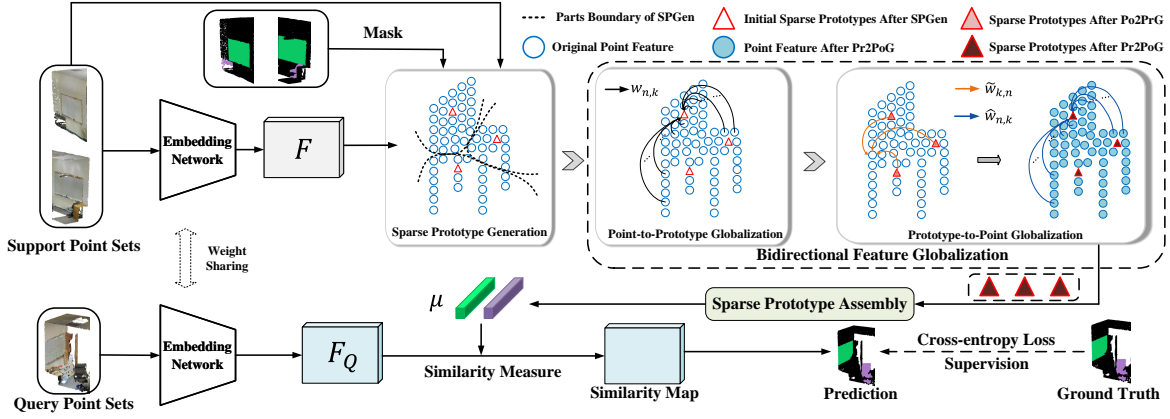


Figure 2. Flowchart of our BFG approach. The support branch is responsible for sparse prototype generation (SPGen), bidirectional feature globalization (BFG), and sparse prototype assembly (SPA). The query branch calculates the similarity maps between the query point features and the prototypes, and predicts the segmentation results.

$\mu^c$  can be defined as:

$$f(\mathcal{F}^c, \mu^c; \mathcal{J}^c, \mu_{\mathcal{J}}^c) = e^{-\mathcal{D}(\mathcal{F}^c, \mu^c; \mathcal{J}^c, \mu_{\mathcal{J}}^c)}, \quad (3)$$

where  $\mathcal{D}(\cdot)$  denotes the distance measurements defined on either  $L_2$ -Norm or inner product operation in what follows.

(1)  *$L_2$ -Norm* is commonly used to calculate the similarity between two feature vectors. With this operation,  $\mathcal{D}(\cdot)$  is defined as:  $\mathcal{D} = \sqrt{d(\mathcal{F}^c, \mu^c) + d(\mathcal{J}^c, \mu_{\mathcal{J}}^c)}$ , where  $d(\mathcal{F}_n^c, \mu_k^c) = \frac{\lambda}{\max(\mathcal{F}_n^c)} \sum_{i=1}^D \|\mathcal{F}_{n,i}^c - \mu_{k,i}^c\|^2$ ,  $d(\mathcal{J}_n^c, \mu_{\mathcal{J},k}^c) = \sum_{l=1}^3 \|\mathcal{J}_{n,l}^c - \mu_{\mathcal{J},k,l}^c\|^2$ , and  $\|\cdot\|$  denotes the  $L_2$ -Norm.  $\lambda$  is the scale factor that keeps  $\mathcal{F}_n^c$  feature-based distances and  $\mathcal{J}_n^c$  coordinate-based distances orders of magnitude consistent and set to 0.85 in our experiments.

(2) *Inner product operation* is formulated as:  $\mathcal{D} = \xi \cdot [d(\mathcal{F}^c, \mu^c) + d(\mathcal{J}^c, \mu_{\mathcal{J}}^c)]$ , where  $d(\mathcal{F}_n^c, \mu_k^c) = \mu_k^{cT} \mathcal{F}_n^c$  and  $(\cdot)^T$  is the transpose operation of the matrix. Similarly,  $d(\mathcal{J}_n^c, \mu_{\mathcal{J},k}^c) = \mu_{\mathcal{J},k}^{cT} \mathcal{J}_n^c$ .  $\xi$  denotes the concentration factor, which is set to 0.5 in our experiments.

### 3.3.1 Point-to-Prototype Globalization.

Point-to-Prototype globalization is carried out based upon the similarity weights from dense point features to sparse prototypes. Specifically, similarity measurement, weights generation, and prototype globalization are carried out in order.

*Similarity Measurement.* Given the initialized prototypes  $\{\mu^c, \mu_{\mathcal{J}}^c\}$ , we compute the similarity between point features and prototype vectors, as  $f_1(\mathcal{F}^c, \mu^c; \mathcal{J}^c, \mu_{\mathcal{J}}^c) = e^{-\mathcal{D}_1}$ . In experiments,  $\mathcal{D}_1$  is defined as  $L_2$ -Norm.

*Weights Generation.* In this procedure, we convert the dense point features into the weights  $w_{n,k}$ , which is defined

on the similarity between the point features and each prototype vector, as:

$$w_{n,k} = \frac{f_{1;n,k}(\mathcal{F}^c, \mu^c; \mathcal{J}^c, \mu_{\mathcal{J}}^c)}{\sum_{n=1}^N f_{1;n,k}(\mathcal{F}^c, \mu^c; \mathcal{J}^c, \mu_{\mathcal{J}}^c)}, \quad (4)$$

where  $f_{1;n,k}(\mathcal{F}^c, \mu^c; \mathcal{J}^c, \mu_{\mathcal{J}}^c)$  is the similarity between the  $n$ -th point feature  $\mathcal{F}_n^c$  and the  $k$ -th prototype vector  $\mu_k^c$ . Such similarity weights represent the semantic similarity between prototypes and point features. A larger weight value means higher semantic similarity.

*Prototype Globalization.* After generating the semantic similarity, prototype globalization is implemented by calculating the weighted average of the point features through the similarity weights. The new prototype  $v_k^c$  is formulated as:

$$v_k^c = \sum_{n=1}^N w_{n,k} \mathcal{F}_n^c. \quad (5)$$

With the weighted average of point features, the global perception is embedded to the sparse prototypes.

### 3.3.2 Prototype-to-Point Globalization.

After Po2PrG, the sparse prototypes have acquired global perception. However, the local point features remain local dependency. To solve it, Pr2PoG based on similarity weights is introduced to embed global perception to local point features. This is implemented through four steps: similarity measurement, weights generation, point feature globalization, and prototype globalization.

*Similarity Measurement.* Given sparse prototypes  $v^c = \{v_k^c\}_{k=1}^K$  which incorporate global perception, we compute the similarity between point features and global prototypes,

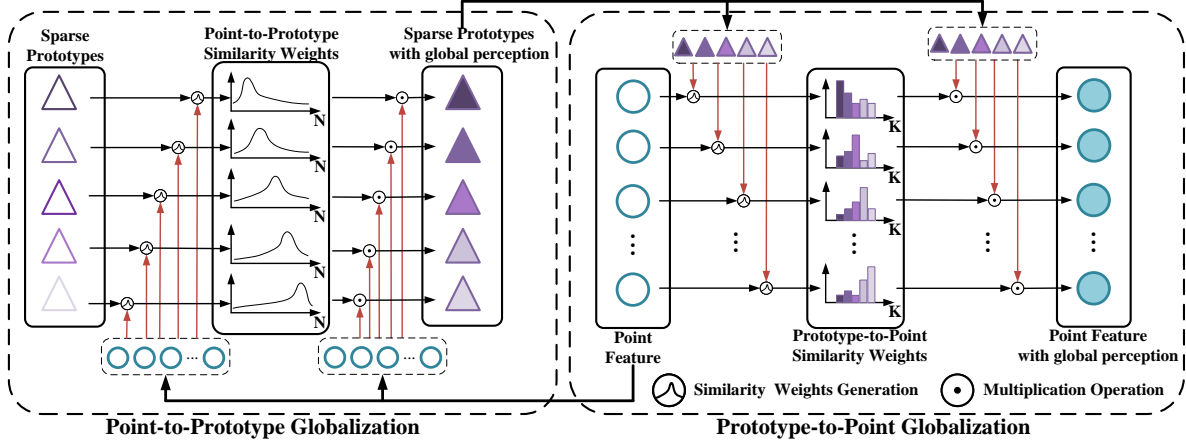


Figure 3. Diagram of point-to-prototype globalization (Po2PrG) and prototype-to-point globalization (Pr2PoG). Prototypes and point features are enhanced with global perception by generating similarity weights in a bidirectional fashion.

as  $f_2(\mathcal{F}^c, v^c; \mathcal{J}^c, v_{\mathcal{J}}^c) = e^{-\mathcal{D}_2}$  where  $v_{\mathcal{J}}^c = \mu_{\mathcal{J}}^c$ . In experiments,  $\mathcal{D}_2$  is defined as the inner product operation.

**Weights Generation.** With the similarity measurement, we correlate the global prototypes with local point features through the semantic perception *i.e.*, the similarity weights from global prototypes to dense point features. The similarity weight  $\tilde{w}_{k,n}$  from the  $k$ -th prototype to the  $n$ -th point feature is defined as:

$$\tilde{w}_{k,n} = \frac{f_{2;k,n}(\mathcal{F}^c, v^c; \mathcal{J}^c, v_{\mathcal{J}}^c)}{\sum_{k=1}^K f_{2;k,n}(\mathcal{F}^c, v^c; \mathcal{J}^c, v_{\mathcal{J}}^c)}. \quad (6)$$

In the weights space, a prototype with higher similarity to the point feature is assigned a higher weight, and vice versa.

**Point Feature Globalization.** Based on the semantic similarity, the updated point features  $\mathfrak{F}_n^c$  are obtained by the weighted average through the similarity weights, as:

$$\mathfrak{F}_n^c = \mathcal{F}_n^c + \sum_{k=1}^K \tilde{w}_{k,n} v_k^c. \quad (7)$$

**Prototype Globalization.** To globalize the prototypes, we employ weights generation to convert the updated point features into new weights  $\hat{w}_{n,k}$ , which is formulated as:

$$\hat{w}_{n,k} = \frac{f_{2;n,k}(\mathfrak{F}_n^c, v^c; \mathcal{J}^c, v_{\mathcal{J}}^c)}{\sum_{n=1}^N f_{2;n,k}(\mathfrak{F}_n^c, v^c; \mathcal{J}^c, v_{\mathcal{J}}^c)}. \quad (8)$$

The prototype globalization of Po2PrG is then used to compute the weighted average through the similarity weights and obtain enhanced prototypes  $r_k^c$ , as:

$$r_k^c = \sum_{n=1}^N \hat{w}_{n,k} \mathfrak{F}_n^c. \quad (9)$$

After Pr2PoG, point features are correlated with the sparse prototypes and the enhanced prototypes are obtained for the following prototype assembly.

### 3.4. Sparse Prototype Assembly

After prototype globalization, we obtain the enhanced prototypes with global semantic perception, as  $r^c = \{r_k^c\}_{k=1}^K$ . When performing semantic segmentation in the following metric learning procedure, it requires to generate the similarity maps between query features and prototype vectors. To perform the measurement, the sparse prototypes from each class require to be fused at first. We first apply MLP to the prototypes, as:  $\hat{r}^c = \mathcal{M}\{r^c\} \in \mathbb{R}^{D \times K}$ . We then calculate the mean prototype  $z^c$  of each class, as

$$z^c = \sum_{k=1}^K \alpha_k \circ \hat{r}_k^c, \quad (10)$$

where  $\alpha_k = \frac{e^{\hat{r}_k^c}}{\sum_{i=1}^K e^{\hat{r}_i^c}} \in \mathbb{R}^{D \times 1}$  is the normalized weight vector of the prototypes for class  $c$ , and  $\circ$  is the Hadamard product. The prototypes of all classes are  $z = \{z^c\}_{c=1}^C$ .

## 4. Experiments

The proposed BFG is evaluated on two point cloud segmentation benchmarks, including the Stanford Large-Scale 3D Indoor Spaces (S3DIS) [1] and ScanNet [4].

### 4.1. Datasets

**S3DIS.** S3DIS is a dataset which collects 3D RGB point clouds from 272 rooms in six indoor environments. Each point is annotated with one of the semantic labels from 13 classes (12 semantic classes plus the clutter). **ScanNet.** The ScanNet dataset contains a total of 1513 scanned scenes.

Table 1. Performance on S3DIS. ‘Embed. Net’ denotes the embedding network (backbone). ‘DGCNN w/o SAN’ and ‘DGCNN w/ SAN’ denote DGCNN backbone without and with SAN, respectively.  $S^i$  represents the split  $i$  is selected to test our BFG.

Method	Embed. Net	2-way						3-way					
		1-shot			5-shot			1-shot			5-shot		
		$S^0$	$S^1$	mean	$S^0$	$S^1$	mean	$S^0$	$S^1$	mean	$S^0$	$S^1$	mean
FT [34]	DGCNN w/o SAN	36.34	38.79	37.57	56.49	56.99	56.74	30.05	32.19	31.12	46.88	47.57	47.23
ProtoNet [5]		48.39	49.98	49.19	57.34	63.22	60.28	40.81	45.07	42.94	49.05	53.42	51.24
MPTI [34]		52.27	51.48	51.88	58.93	60.56	59.75	44.27	46.92	45.60	51.74	48.57	50.16
<b>BFG(ours)</b>		52.50	53.26	<b>52.88</b>	59.26	62.82	<b>61.04</b>	43.80	47.76	<b>45.78</b>	49.80	55.10	<b>52.45</b>
ProtoNet [5]	DGCNN w/ SAN	50.98	51.90	51.44	61.02	65.25	63.14	42.16	46.76	44.46	52.20	56.20	54.20
MPTI [34]		53.77	55.94	54.86	61.67	67.02	64.35	45.18	49.27	47.23	54.92	56.79	55.86
<b>BFG(ours)</b>		55.60	55.98	<b>55.79</b>	63.71	66.62	<b>65.17</b>	46.18	48.36	<b>47.27</b>	55.05	57.80	<b>56.43</b>

Table 2. Performance on ScanNet. ‘Embed. Net’ denotes the embedding network (backbone). ‘DGCNN w/o SAN’ and ‘DGCNN w/ SAN’ denote DGCNN backbone without and with SAN, respectively.  $S^i$  represents the split  $i$  is selected to test our BFG.

Method	Embed. Net	2-way						3-way					
		1-shot			5-shot			1-shot			5-shot		
		$S^0$	$S^1$	mean	$S^0$	$S^1$	mean	$S^0$	$S^1$	mean	$S^0$	$S^1$	mean
FT [34]	DGCNN w/o SAN	31.55	28.94	30.25	42.71	37.24	39.98	23.99	19.10	21.55	34.93	28.10	31.52
ProtoNet [5]		33.92	30.95	32.44	45.34	42.01	43.68	28.47	26.13	27.30	37.36	34.98	36.17
MPTI [34]		39.27	36.14	37.71	46.90	43.59	<b>45.25</b>	29.96	27.26	28.61	38.14	34.36	36.25
<b>BFG(ours)</b>		38.63	36.82	<b>37.73</b>	45.67	42.36	44.02	30.57	29.02	<b>29.80</b>	38.64	34.75	<b>36.70</b>
ProtoNet [5]		37.99	34.67	36.33	52.18	46.89	49.54	32.08	28.96	30.52	44.49	39.45	41.97
MPTI [34]	DGCNN w/ SAN	42.55	40.83	<b>41.69</b>	54.00	50.32	<b>52.16</b>	35.23	30.72	32.98	46.74	40.80	43.77
<b>BFG(ours)</b>		42.15	40.52	41.34	51.23	49.39	50.31	34.12	31.98	<b>33.05</b>	46.25	41.38	<b>43.82</b>

All the points except the unannotated space are annotated by 20 semantic classes.

**Data pre-processing.** Following [34], we divide the semantic classes of each dataset into two non-overlapping combinations  $S^0$  (S3DIS: *beam, board, bookcase, ceiling, chair, column*. ScanNet: *bathub, bed, bookshelf, cabinet, chair, counter, curtain, desk, door, floor*) and  $S^1$  (S3DIS: *door, floor, sofa, table, wall, window*. ScanNet: *other furniture, picture, refrigerator, show curtain, sink, sofa, table, toilet, wall, window*) according to the alphabetical order. To facilitate the point cloud be fed to the network, we process the datasets following [18, 34]: splitting the rooms into  $1m \times 1m$  blocks and sampling 2048 points from the block each time. After that, the S3DIS and ScanNet datasets are split into 7547 and 36350 blocks, respectively. Since the area of each block is small, the sampled points can only contain one instance object or a local area of an instance object. For each dataset, the cross-validation is performed to our method, which is implemented by selecting one split  $S^i$  as the train class set and regarding the other split  $S^{1-i}$  as the test class set. If  $S^i$  ( $S^0$  or  $S^1$ ) is used as the test class set, the blocks containing the  $S^i$  class are selected as the test set, and the blocks containing the  $S^{1-i}$  class is selected as the training set. Following [34], the sampling process of each episode in the training process is as follows: (1) Randomly selecting a combination of  $N$  classes from the train class set  $S^i$  to set up  $N$ -way as the foreground, and the remaining classes are regarded as the background. (2) Randomly sampling  $K$  ( $K$ -shot) support sets and query sets based on the selected  $N$ -way class combination in the training set. When testing, we take the same operation to preprocess the

data of the test class set.

## 4.2. Experimental Settings

**Evaluation Protocols.** Mean intersection over union (mIoU), which is widely used in point cloud segmentation, is selected as the metric for performance evaluation.

**Implementation Details.** DGCNN [26] is proposed as a basic point cloud classification and segmentation network that is widely used for many point cloud processing tasks. SAN [34] is introduced to explore correlations of semantic context within the point set. Based on this, DGCNN (without or with SAN) is selected as the backbone. Our approach utilizes ProtoNet [34] as the baseline. Following [34], Gaussian jittering operation and random rotation operation around z-axis data augmentation strategies are used in the training process. Our approach runs on a single NVIDIA TITAN RTX GPU with the batch size set to 1 and the number of training iterations set to 80000. The Adam optimizer is used with an initial learning rate of 0.0001 for the embedding network. Furthermore, an initial learning rate for the remaining parts is set to 0.001. Since S3DIS and ScanNet belong to the same type of indoor scene point cloud data, the number of sparse prototypes is set to 5 on both S3DIS and ScanNet through extensive experiments.

## 4.3. Performance

**S3DIS.** In Table 1, our BFG is compared with the state-of-the-art methods. DGCNN without and with SAN are selected as the backbone, BFG outperforms state-of-the-art methods in all experimental settings. With the 2-way 1-shot setting and DGCNN (with SAN) backbone, our BFG



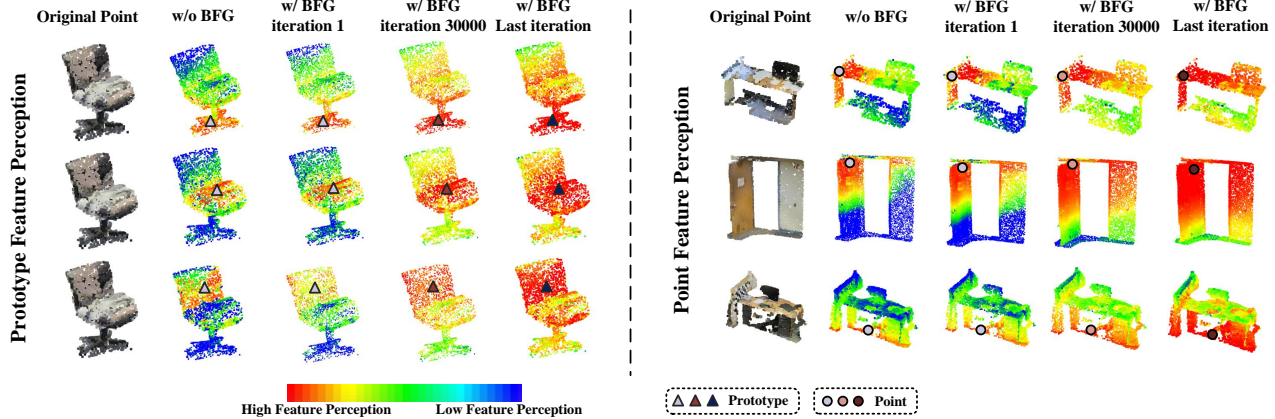


Figure 4. Feature perception of prototypes and point features during training. Both prototype vectors and local point features with BFG are gradually embedded with global perception, which extends the receptive fields to full instance extent. (Best viewed in color)

achieves **4.35%** (55.79% vs. 51.44%) performance improvement over the baseline ProtoNet. Note that for the  $S^0$  setting in 2-way 1-shot experiments, BFG improves the baseline ProtoNet by **4.62%**, which outperforms the state-of-the-art method MPTI by **1.83%**. Compared with ProtoNet, it is concluded that the extraction of sparse prototypes does help to improve the segmentation performance. Compared with MPTI, we can see that too many prototypes will not have higher performance. On the contrary, a moderate number of prototypes with the optimal global representation after BFG greatly improve the performance.

**ScanNet.** Table 2 displays the segmentation performance on ScanNet. BFG again outperforms state-of-the-art methods in most experimental settings. For the 2-way 1-shot setting and DGCNN (without SAN) backbone, BFG improves the baseline ProtoNet by **5.29%** (37.73% vs. 32.44%), which is a large margin for the challenging few-shot 3D segmentation problem. Compared to S3DIS, there are more classes and point sets in ScanNet, which facilitates learning richer representation related to various instances. Thereby, the improvement on 2-way 1-shot setting of ScanNet is larger than that on S3DIS. With the DGCNN (with SAN) backbone, our BFG is better than or on par with state-of-the-arts.

#### 4.4. Visualization Analysis

**Feature Perception.** As Fig. 4 shows, we visualize the feature perception of different prototypes and point features during training, respectively. In the left part, the feature perception of different prototypes in the same class without and with BFG is clearly displayed. Prototype feature perception in Fig. 4 is reflected by the similarity between the point features and each prototype. In the right part, the feature perception of point features without and with BFG is given. Point feature perception in Fig. 4 is reflected by the point

feature maps before and after adding BFG. One can see that with BFG, both the prototypes and the point features gradually approach global perception. From the visualizations, the feature perception of both the prototypes and the point features after our BFG can be extended to full instance extent. These clearly demonstrate the superiority of our BFG in feature representation over previous prototype methods.

**Segmentation Results.** As Fig. 5 shows, we compare the segmentation performance by the baseline method and our BFG on S3DIS and ScanNet datasets. The segmentation results show that by introducing bidirectional feature globalization, BFG has achieved good segmentation performance on the instances with scale variation and deformation. In addition, the optimal sparse prototypes also greatly alleviate the false segmentation between classes.

#### 4.5. Ablation Study

We conducted ablation studies with 2-way 1-shot setting on the S3DIS dataset to verify the effectiveness of BFG. ProtoNet [5] is selected as the baseline.

Table 3. Ablation study of modules in our BFG approach. The first row is the performance of the baseline ProtoNet. ‘SPGen’ denotes the sparse prototype generation module with Sparse Prototype Assembly, ‘Po2PrG’ denotes Point-to-Prototype Globalization, and ‘Pr2PoG’ denotes Prototype-to-Point Globalization.

+SPGen	+Po2PrG	+Pr2PoG	Mean	$\Delta$	$\Sigma\Delta$
			51.44		
✓			53.34	1.90	1.90
✓	✓		54.39	1.05	2.95
✓	✓	✓	<b>55.79</b>	1.40	4.35

**SPGen.** As shown in Table 3, with sparse prototype generation, BFG improves the segmentation performance

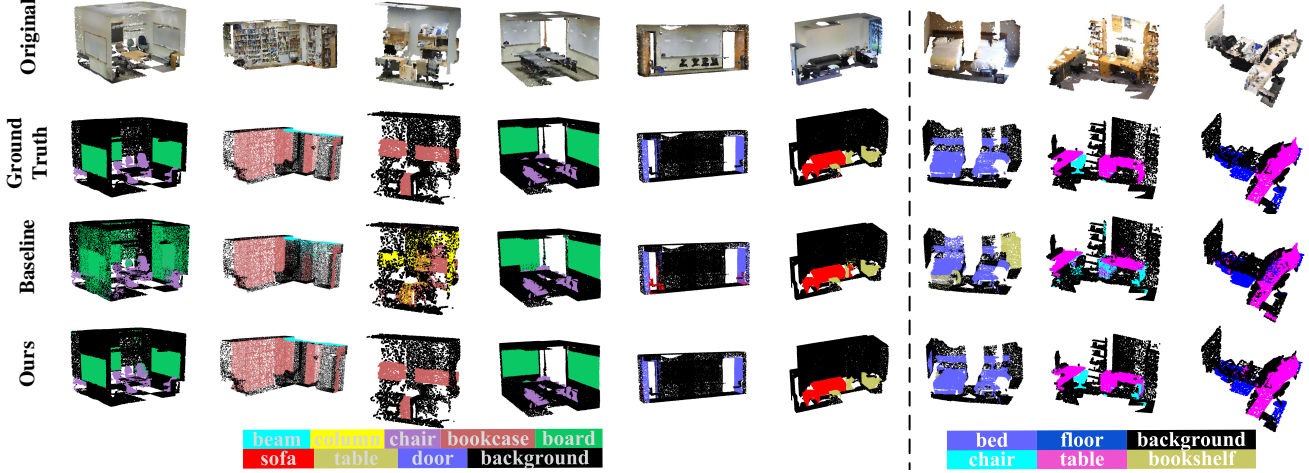


Figure 5. Visualizations of segmentation results on S3DIS and ScanNet. Left: S3DIS. Right: ScanNet.

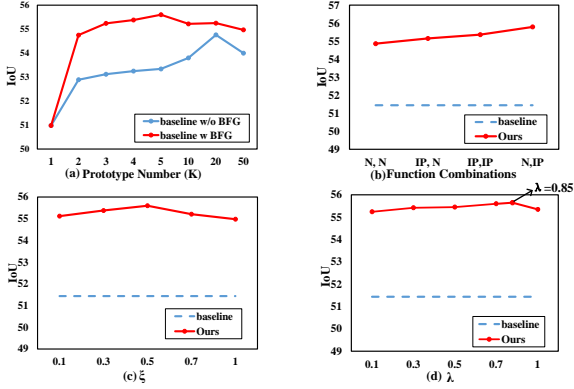


Figure 6. Ablation study of the hyper-parameters and modules. (a) Prototype number  $K$ . (b) Distance measurement function combinations ('N' represents  $L_2$ -Norm and 'IP' represents inner product operation). (c) Hyper-parameter  $\xi$ . (d) Hyper-parameter  $\lambda$ .

by 1.90% (53.34% vs. 51.44%) on mean IoU of the  $S^0$  and  $S^1$  split, which validates that multiple prototypes improve segmentation performance more significantly. To maximizing the performance gain, a proper number of prototypes should be used. As shown in Fig. 6(a), 20 prototypes can reach the best performance.

**Po2PrG.** In Table 3, Po2PrG further improves the performance by 1.05% (54.39% vs. 53.34%), which validates the prototypes acquire global perception after Po2PrG embeds local point features to the prototypes.

**Pr2PoG.** As shown in Table 3, by using Pr2PoG, our BFG improves the segmentation performance by 1.40% (55.79% vs. 54.39%), validating that the local point features obtain the global perception through embedding the global prototypes to the local point features. This clearly demonstrates the superiority of our BFG over other meth-

ods on global feature representation.

**Number of Sparse Prototypes.** Since S3DIS and ScanNet are divided into the blocks of  $1m \times 1m$ , the input of the network is 2048 points sampled from a small block. Therefore, for the input of the network, the point set only contains a single object or a part of a single object, and there will not be multiple objects. Thus, the theory that each of the prototypes represents a part of an object is sound enough. From the red line in Fig. 6(a), we validate that after adding BFG, setting the number of prototypes to  $K = 5$  can achieve the best performance, where the number of prototypes is within an acceptable range. Moreover, the value of  $K$  shows an upward trend in segmentation performance within a certain interval  $[1, 5]$ , and reaches a peak when  $K = 5$ . As the  $K$  value gradually increases, the segmentation performance decreases. Adding BFG makes the peak of the performance curve come early and greatly reduces the number of multiple prototypes, which surpasses the performance of MPTI with 100 prototypes while using just  $K = 5$  prototypes. This once again verifies that BFG can extract the optimal representation of prototypes.

**Distance Measurement.** In Fig. 6(b), we compare the combination of  $L_2$ -Norm and inner product operation for distance measurement. The results from the combinations show that the distance measurement defined by ( $L_2$ -Norm, Inner Product) is preferable. As Fig. 6(c) and (d) show, we illustrate the effects of hyper-parameter  $\xi$  in inner product operation and  $\lambda$  in  $L_2$ -Norm. For  $\lambda$ , the best performance of our BFG occurs at  $\lambda = 0.85$ . Furthermore, the best performance of the parameter  $\xi$  which is a numerical adjustment of the distance occurs at 0.5. Obviously, the segmentation performance of our BFG is insensitive to them.



## 5. Conclusion

We propose bidirectional feature globalization (BFG), which improves the representation ability of prototypes by incorporating global feature perception in a bidirectional fashion. BFG realizes the optimal representation of part-wised semantics through globalizing sparse prototypes and dense point features. BFG implements prototype combination towards fusing part-wised object semantics. Extensive experiments on commonly used 3D segmentation datasets demonstrate the effectiveness of BFG, in striking contrast with other state-of-the-art methods. As a simple-yet-effective approach, BFG provides a fresh insight to the challenging few-shot segmentation task of point clouds.

## 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 *IEEE CVPR*, pages 1534–1543, 2016.
- [2] Matan Atzmon, Haggai Maron, and Yaron Lipman. Point convolutional neural networks by extension operators. *arXiv preprint arXiv:1803.10091*, 2018.
- [3] Alexandre Boulch. Convpoint: Continuous convolutions for point cloud processing. *Computers & Graphics*, 88:24–34, 2020.
- [4] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *IEEE CVPR*, pages 5828–5839, 2017.
- [5] Nanqing Dong and Eric P Xing. Few-shot semantic segmentation with prototype learning. In *BMVC*, volume 3, 2018.
- [6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pages 1126–1135. PMLR, 2017.
- [7] 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. In *IEEE CVPR*, pages 11108–11117, 2020.
- [8] Le Hui, Jia Yuan, Mingmei Cheng, Jin Xie, Xiaoya Zhang, and Jian Yang. Superpoint network for point cloud oversegmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5510–5519, 2021.
- [9] Muhammad Abdullah Jamal and Guo-Jun Qi. Task agnostic meta-learning for few-shot learning. In *IEEE CVPR*, pages 11719–11727, 2019.
- [10] Varun Jampani, Deqing Sun, Ming-Yu Liu, Ming-Hsuan Yang, and Jan Kautz. Superpixel sampling networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 352–368, 2018.
- [11] Mingyang Jiang, Yiran Wu, Tianqi Zhao, Zelin Zhao, and Cewu Lu. Pointsift: A sift-like network module for 3d point cloud semantic segmentation. *arXiv preprint arXiv:1807.00652*, 2018.
- [12] Loïc Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *IEEE CVPR*, pages 4558–4567, 2018.
- [13] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *NIPS*, 31:820–830, 2018.
- [14] Jinxian Liu, Bingbing Ni, Caiyuan Li, Jiancheng Yang, and Qi Tian. Dynamic points agglomeration for hierarchical point sets learning. In *IEEE ICCV*, pages 7546–7555, 2019.
- [15] Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan. Densepoint: Learning densely contextual representation for efficient point cloud processing. In *IEEE ICCV*, pages 5239–5248, 2019.
- [16] Yongqiang Mao, Kaiqiang Chen, Wenhui Diao, Xian Sun, Xiaonan Lu, Kun Fu, and Martin Weinmann. Beyond single receptive field: A receptive field fusion-and-stratification network for airborne laser scanning point cloud classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 188:45–61, 2022.
- [17] Yongqiang Mao, Xian Sun, Wenhui Diao, Kaiqiang Chen, Zonghao Guo, Xiaonan Lu, and Kun Fu. Semantic segmentation for point cloud scenes via dilated graph feature aggregation and pyramid decoders. *arXiv preprint arXiv:2204.04944*, 2022.
- [18] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE CVPR*, pages 652–660, 2017.
- [19] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017.
- [20] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.
- [21] Amirreza Shaban, Shray Bansal, Zhen Liu, Irfan Essa, and Byron Boots. One-shot learning for semantic segmentation. *arXiv preprint arXiv:1709.03410*, 2017.
- [22] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *IEEE CVPR*, pages 1199–1208, 2018.
- [23] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotequi, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *IEEE ICCV*, pages 6411–6420, 2019.
- [24] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *NIPS*, 29:3630–3638, 2016.
- [25] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *IEEE CVPR*, pages 10296–10305, 2019.
- [26] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *TOG*, 38(5):1–12, 2019.
- [27] Yu-Xiong Wang and Martial Hebert. Learning to learn: Model regression networks for easy small sample learning. In *ECCV*, pages 616–634. Springer, 2016.
- [28] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *IEEE CVPR*, pages 9621–9630, 2019.

- [29] Mutian Xu, Runyu Ding, Hengshuang Zhao, and Xiaojuan Qi. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *IEEE CVPR*, pages 3173–3182, 2021.
- [30] 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*, pages 403–417, 2018.
- [31] Chi Zhang, Guosheng Lin, Fayao Liu, Rui Yao, and Chunhua Shen. Canet: Class-agnostic segmentation networks with iterative refinement and attentive few-shot learning. In *IEEE CVPR*, pages 5217–5226, 2019.
- [32] Xiaolin Zhang, Yunchao Wei, Yi Yang, and Thomas S Huang. Sg-one: Similarity guidance network for one-shot semantic segmentation. *IEEE transactions on cybernetics*, 50(9):3855–3865, 2020.
- [33] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. Pointweb: Enhancing local neighborhood features for point cloud processing. In *IEEE CVPR*, pages 5565–5573, 2019.
- [34] Na Zhao, Tat-Seng Chua, and Gim Hee Lee. Few-shot 3d point cloud semantic segmentation. In *IEEE CVPR*, pages 8873–8882, 2021.