Efficient Point Cloud Segmentation with Geometry-aware Sparse Networks

Maosheng Ye[1], Rui Wan[2], Shuangjie Xu[1], Tongyi Cao[2], and Qifeng Chen[1]

[1] HKUST, Hong Kong, China
{myeag,sxubj}@connect.ust.hk, cqf@ust.hk
[2] DeepRoute.Ai, Shenzhen, China
{ruiwan,tongyicao}@deeproute.ai

Abstract. In point cloud learning, sparsity and geometry are two core properties. Recently, many approaches have been proposed through single or multiple representations to improve the performance of point cloud semantic segmentation. However, these works fail to maintain the balance among performance, efficiency, and memory consumption, showing incapability to integrate sparsity and geometry appropriately. To address these issues, we propose the Geometry-aware Sparse Networks (GASN) by utilizing the sparsity and geometry of a point cloud in a single voxel representation. GASN mainly consists of two modules, namely Sparse Feature Encoder and Sparse Geometry Feature Enhancement. The Sparse Feature Encoder extracts the local context information, and the Sparse Geometry Feature Enhancement enhances the geometric properties of a sparse point cloud to improve both efficiency and performance. In addition, we propose deep sparse supervision in the training phase to help convergence and alleviate the memory consumption problem. Our GASN achieves state-of-the-art performance on both SemanticKITTI and Nuscenes datasets while running significantly faster and consuming less memory.

1 Introduction

Large-scale outdoor point cloud segmentation has been a crucial task for autonomous driving systems and has demanding requirements for efficiency, performance, and memory consumption. PointNet [29] and PointNet++ [30] are the pioneering works that directly operate on the raw point cloud to maintain and utilize the pointwise geometry (accurate measurement information), which is one of the core properties of a point cloud. However, it is hard to apply these approaches in outdoor scenarios due to memory consumption and runtime efficiency. RandLA [14] applies a random sampling strategy to reduce the number of points to improve efficiency, which leads to some information loss. With the popularity of sparse convolutions [10, 50], there has been some progress in utilizing the sparse voxel-based representation (e.g., AF2S3Net [6] and Cylinder3D [59]), which is a kind of representation that preserves the metric space. Compared with point representation, the merits of the sparse voxel-based representation lie in the effectiveness and efficiency of quickly expanding the receptive

fields based on sparsity, another important property of a point cloud. Furthermore, sparse voxel-based representation, which aggregates point features within the local neighborhood, can significantly reduce memory usage. Moreover, traditional or current popular convolutional neural networks (CNNs) can be directly applied to extract better context information.

Recently, several works recognize the limitation of a single representation and explore richer information by combining multiple representations. PVCNN [21] fuses point-based and voxel-based representations with MLP layers and dense 3D convolution layers but does not take point cloud sparsity into consideration. SPVCNN [35] and DRINet [52] design the sparse convolution layers and pointwise operational layers to fuse features considering sparsity and geometry. Furthermore, RPVNet [47] combines the range-view, point, and voxel representations for point cloud segmentation. The general framework of current multi-representation learning is to utilize sparse convolutions for locality and sparsity and pointwise operations for geometry learning, aiming to combine sparsity and geometry for better performance and efficiency. While these approaches lead to some performance improvements, they are not efficient enough to meet the need of a real-time system due to the extra computation cost brought by the additional views or representations. Meanwhile, according to the results of the experiments in these approaches, the voxel-based representation is still the dominant one, with which these methods have already achieved decent performance. Inspired by these observations, we propose the Geometry-aware Sparse Networks to explore extra geometric properties based on a single sparse voxel-based representation. Our Geometry-aware Sparse Networks incorporate sparsity and geometry in a single representation without introducing extra computation costs from multi-representation fusion. Our Geometry-aware Sparse Networks mainly

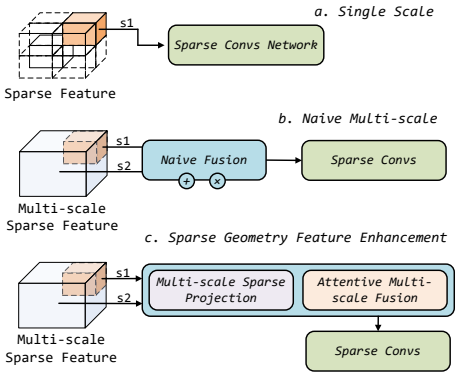

Fig. 1. Comparison between two common ways to deal with sparse features and our proposed method.

have two modules: Sparse Feature Encoder (SFE) and Sparse Geometric Feature Enhancement (SGFE). Each module takes the output of the other module

as input to fully explore the sparsity and geometry of a point cloud at a low computation cost and memory usage. In SGFE (shown in Fig. 1), we propose a novel multi-scale sparse projection layer to explore more geometry and Attentive Scale Selection for multi-scale feature selection. Apart from that, we apply deep sparse supervision compared with the most common dense manner to alleviate the pressure of memory consumption.

Our contributions are summarized as follows:

- We propose a novel network architecture to fully exploit the sparsity and geometry properties. Multi-scale sparse projection layer and Attentive Scale Selection in the Sparse Geometric Feature Enhancement are proposed to enhance the geometry for feature learning.
- Deep Sparse Supervision is proposed to design the supervision in a sparse manner to reduce the memory cost.
- We evaluate our proposed approach on large-scale outdoor scenario datasets including SemanticKITTI [1] and nuScenes-lidarseg [3] to demonstrate the effectiveness of our method. We achieve state-of-the-art performance on both datasets with a runtime speed of $59ms$ on average on an Nvidia RTX 2080 Ti GPU.

2 Related Work

Indoor Point Cloud Segmentation. The point cloud from an indoor scene often has closely positioned points with a small range. The existing indoor point cloud segmentation approaches can be classified according to their model representations. For point-based approaches, PointNet [29], PointNet++ [29], and their related works [19, 45, 53, 20, 28, 31] based on a similar architecture are popular models in this task. Most of these works explore the local neighborhood context while preserving the inherent geometry of a point cloud. They use different grouping and permutation invariant operations to promote performance. The other mainstream methods [33, 25, 21, 16, 17] follow the volumetric representation by partitioning the space as discrete pixels/voxels and then applying 2D/3D CNN architectures to the regular representation. Graph-based approaches for point cloud learning [40, 55, 37, 41, 43, 48] are also popular due to the nature of graphs to deal with unorderedness and the capability to model the relationship among points. Currently, with the popularity of transformers, some works [56, 11] achieve state-of-the-art performance in indoor point cloud learning by introducing transformer-based architectures. Although a number of novel architectures have been proposed to improve point cloud learning, some of them fail to generalize to outdoor scenarios with thousands of hundreds of points.

Outdoor Point Cloud Segmentation. Compared with indoor point cloud segmentation, the sparsity and larger number of points pose great challenges for existing approaches. Point-based methods such as KPConv [38] and RandLA [14] extend the architecture of PointNet [29] or PointNet++ [30] and adopt sampling

strategies to alleviate these problems but lead to extra information loss. KPConv [38] introduces the kernel point selection process to generate high-quality sampling points. Range-view-based approaches [44, 46, 8] project the point cloud into range views or spherical representations and apply efficient CNN architectures. However, the range view cannot maintain the metric space and introduces distortions, which potentially leads to performance degradation. Some other approaches [59, 54, 7, 6, 49] quantize a point cloud into some pre-defined space or representations (e.g., polar grids, 2D grids, and sparse 3D grids) and then apply regular convolution neural networks or sparse convolutions [10, 50, 42] to achieve the balance between efficiency and performance. A line of works integrates the multiple representations, including range views, voxel representations, and point representation, to exploit the potential of different representations deeply [35, 47, 52, 21]. These works utilize different architectures for different representations and propose various fusion strategies and show strong performance gain compared to single-representation-based methods, at the cost of extra running time.

Image Segmentation to Point Cloud Segmentation. The fully convolutional network (FCN) [22] is one of the pioneering works for image segmentation with deep learning. Based on FCN and existing prevalent CNN architecture, DeepLab [4], PSPNet [57] and their following works [51, 5] are proposed with multi-scale or multiple dilation rate strategies to explore more hierarchical local context information. Furtherly, HRNet [34] fuses different resolution heatmaps in a single framework and keeps the high resolution to improve the performance. Considering the great process achieved in image segmentation, lots of works [52, 26, 39, 54, 59] have applied these tricks, including hierarchy learning, attention mechanism, or backbones into point cloud segmentation. Some works [59, 7] are built on U-net [32] with sparse convolution acceleration.

3 Approach

The overall network architecture of our approach consists of two modules: 1) Sparse Feature Encoder and 2) Sparse Geometry Feature Enhancement. Sparse Feature Encoder serves as a basic block for fast local context aggregation. While Sparse Geometry Feature Enhancement, which takes the output of Sparse Feature Encoder as input, enhances the geometric information by multi-scale sparse projection and attentive scale selection layer. Both modules interact in sparse space, saving the computation cost and increasing the runtime efficiency. Moreover, we apply Deep Sparse Supervision at the voxel level to alleviate the memory issues resulting from dense supervision. Fig. 2 demonstrates the overall framework of our proposed approach.

3.1 Prerequisite

Voxelization. Given a grid size s, voxelization is the process that discretizes a point $p_i = (x_i, y_i, z_i)$ to its voxel index V_i by the following equation:

$$V_i = (\lfloor x_i/s \rfloor, \lfloor y_i/s \rfloor, \lfloor z_i/s \rfloor), \tag{1}$$

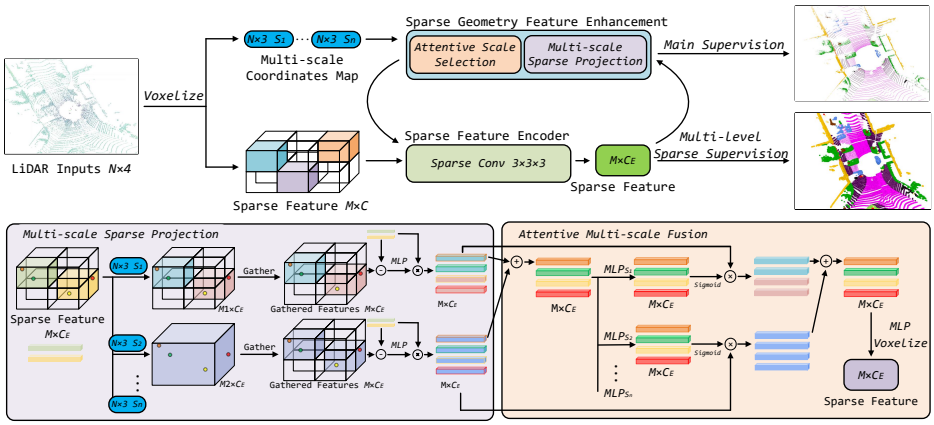

Fig. 2. The overall structure of our GASN. In the top half of the figure, LiDAR input is firstly voxelized as sparse features. Then the sparse feature encoder utilizes sparse convolutions to process the sparse features. Furthermore, sparse geometry feature enhancement will enhance the features by multi-scale sparse projection and attentive scale selection layer to generate the input of sparse feature encoder at the next stage. Sparse supervision will be attached to the output of the sparse feature encoder as an auxiliary loss. The bottom line describes the details of multi-scale sparse projection and attentive scale selection. N is the number of points, M_i is the number of voxels, C_E is the channel dimension.

where $\lfloor \cdot \rfloor$ is a floor function. V_i is represented as a scalar of the voxel index for the i-th point p_i, and. Since each voxel could contains multiples points or multiple fine-grained sub-voxels, we define a scatter function Ψ and gather function Φ which is widely used in [52, 58], where the former performs the clustering process to cluster point features or sub-voxel features $\mathcal{S} \in R^{N_\mathcal{S} \times C}$ to voxel features $\mathcal{V} \in R^{N_\mathcal{V} \times C}$ under larger voxel scale while the latter reverses from voxel features $\mathcal{V}$ to point features or sub-voxel features $\mathcal{S}$:

$$\mathcal{V} = \Psi\left(\mathcal{S}, V_\mathcal{S}\right) = \left\{ \sum_j \mathcal{S}_j \left| V_{\mathcal{S}_j} \in V_{\mathcal{V}_i}, i = 1, \ldots, N_\mathcal{V} \right. \right\}, \tag{2}$$

$$\mathcal{S} = \Phi\left(\mathcal{V}, V_\mathcal{V}\right) = \left\{ \mathcal{V}_j | V_{\mathcal{S}_i} \in V_{\mathcal{V}_j}, i = 1, \ldots, N_\mathcal{S} \right\}, \tag{3}$$

where C is the channel number, $V_\mathcal{V}$ and $V_\mathcal{S}$ is the voxel index for point clouds, $V_{\mathcal{V}_j}$ means the voxel index for the voxel feature $\mathcal{V}_j$, which is the same as $V_{\mathcal{S}_i}$. The Equ. 2 and Equ. 3 mainly reveal the mutual transformation between point features and voxel features: Scatter and Gather. To convert a raw point cloud into voxelwise features, we use the similar approach used in DRINet [52] and Cylinder3D [59] namely **GAFE**.

Algorithm 1 Multi-scale Sparse Projection

Input: Input sparse voxel features F_v with corresponding voxel index V and pre-defined scale set S

Output: Multi-scale features O

$L = []$
for each $s \in S$ **do**
  $\mathcal{V}^s = (\lfloor \mathcal{V}_x/s \rfloor, \lfloor \mathcal{V}_y/s \rfloor, \lfloor \mathcal{V}_z/s \rfloor)$
  $F_v^s = \Psi\left(F_v, \mathcal{V}^s\right)$
  $G^s = \mathrm{MLP}(\Phi(F_v^s, \mathcal{V}))$
  $O^s = G^s * F_v$
  L.append(O^s)
end for
$O \leftarrow \mathrm{Stack}(L)$
return O

3.2 Sparse Feature Encoder

The sparse voxel representation makes it easy to apply standard convolution operations to extract local context information. Thus, in order to keep high runtime efficiency and explore more locality, we utilize sparse convolution layers [10, 42, 50] instead of dense convolution [21]. One of the good merits of sparse convolution lies in the sparsity, with which the convolution operation only considers the non-empty voxels. Based on this observation, we build our sparse feature encoder (SFE) with sparse convolution to quickly expand the receptive field with less computation cost. We adopt the ResNet BottleNeck [12] while replacing the ReLU activation with Leaky ReLU activation [24]. In order to keep a high efficiency, all the channel number for sparse convolution is set to 64.

3.3 Sparse Geometry Feature Enhancement

After obtaining the sparse voxelwise features from the sparse feature encoder, we aim to enhance the voxelwise features with more geometric guidance by our Sparse Geometry Feature Enhancement (SGFE).

Multi-scale Sparse Projection Layer. Inspired by previous works [57, 30, 52] which focus on multi-scale features aggregation, we notice that hierarchical context information helps enhance the capability of feature extraction, especially for point cloud, which has inherent scale invariance and geometry. Multi-scale features bring point cloud learning more geometric enhancement since each voxel scale reflects one specific physical dimension property. However, it is not applicable to directly apply Pyramid Pooling Module from PSPNet [57] due to the sparsity of point clouds. Also, DRINet [52] proposes points pooling at point level by scattering operation while introducing extra huge memory cost when considering large-scale scenarios. As a result, we propose **Multi-scale Sparse Projection Layer** to exploit the multi-scale features at a sparse voxel level with a much lower memory cost.

Given the input voxelwise features $F_v \in R^{N_\mathcal{V} \times C}$ with corresponding voxel index V and pre-defined pooling scale set S, where $N_\mathcal{V}$ is the voxel number and C is the channel number. Multi-scale Sparse Projection is described in Algo. 1. Different scales contain different geometry prior since the scale semantics in the point cloud are proportional to the real dimension, reflecting the physical metric space. For each pooling scale s, we first re-calculate the voxel index under the scale s to ensure the features within the same pooling window will have the same voxel index. Then, we apply *Scatter* operation Ψ to perform the sparse pooling in order to obtain feature mean with local geometric priors $F_v^s \in R^{N_{\mathcal{V}_s} \times C}$ within each pooled region, where $N_{\mathcal{V}_s}$ is the voxel number under scale s. The embedding features $G^s \in R^{N_\mathcal{V} \times C}$ are based on the normalized features with learnable MLP layers. We then use tensor elementwise multiplication to obtain projection features O^s at scale s. Finally, features that reveal the geometry at different scales are stacked together.

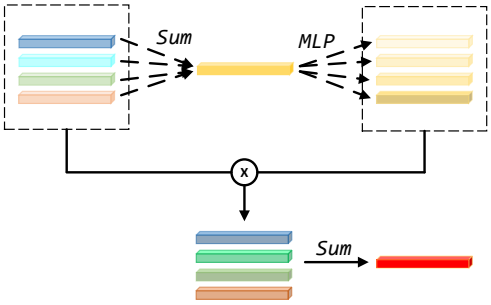

Fig. 3. An example that demonstrates the attentive scale selection. $\otimes$ indicates tensor element-wise multiplication.

Attentive Scale Selection. After obtaining the multi-scale features from the multi-scale sparse projection layer, the naive way of multi-scale features fusion is to apply tensor concatenation or tensor summation. As such, all the features from different scales share the same weights and are treated equally. A common consensus in the point cloud is that features from different scales have different geometry prior and focus on scene understanding. From this point of view and motivated by the SENet [13] and SKNet [18], a better approach to fusing multi-scale features is to apply a re-weighting strategy along scale dimension for each feature channel to re-distribute the importance of each scale.

We first sum over all the input tensors $\{O^s | s \in S\}$ for the first stage fusion to collect all the information from different scales as follows:

$$\mathcal{O} = \sum_s O^s, \tag{4}$$

where $\mathcal{O} \in R^{N_\mathcal{V} \times C}$ in the summed features. Inspired by recent popular attention works [18], we apply a scale-wise MLP layer with sigmoid activation for the re-

sults to get the attentive embedding for each scale. Finally, tensor multiplication between attention weight tensor and multi-scale features is applied, followed by the scale dimension's tensor sum:

$$\mathcal{A}\_\mathcal{O} = \sum_s \text{Sigmoid}(\text{MLP}^s(\mathcal{O})) * O^s, \tag{5}$$

where $\mathcal{A}\_\mathcal{O} \in R^{N_\mathcal{V} \times C}$ is the output attentive features for the SGFE module. Fig. 3 demonstrates the overall process.

3.4 Deep Sparse Supervision

Dense supervision on each pixel/voxel is a popular way in both 2D and 3D semantic segmentation tasks. Previous methods [54, 59] generate dense feature maps with dense supervision. Although these works have considered the sparsity with their network architectures, they ignore this property when designing the loss, one of the key differences between 2D and 3D data. In fact, dense supervision with fine-grained feature maps brings a significant overload on memory usage. For example, when using a grid size of $0.2m$, the memory consumption (about 500 Mb) for a single dense feature map with 20 classes could be problematic. Based on these observations and inspired by [23], we propose a novel **Deep Sparse Supervision (DSS)** to deal with supervision in a deep sparse style.

Specifically, we generate voxel semantic labels at different scales in the training stage, and the supervision only acts on valid voxels rather than whole dense feature maps. Each voxel label is assigned with the major vote point labels within the voxel. Since we stack multiple blocks of SFE which generate sparse voxelwise features at different scales, we apply sparse supervision to the output voxelwise features stage by stage as auxiliary loss, one of the useful optimization techniques. We also apply sparse supervision for the main final prediction branch. The auxiliary loss helps optimize the training, while the main branch loss accounts for the most gradients.

In the testing stages, all the auxiliary branches are disabled to keep the runtime efficiency. This kind of training strategy has proven its effectiveness in image-based segmentation [22]. We consider the sparsity of point clouds and apply it in a sparse manner to save memory consumption.

3.5 Final Prediction

For the final semantic prediction, we fuse the multi-stage features from the output of each Sparse Geometry Feature Enhancement Layer by gathering operations to the most fine-grained scale of the voxel. To obtain pointwise results, we also apply the gathering strategy, in which each point is attached with the semantic features from the voxel that it lies in. The whole algorithm for our framework is illustrated in Algo. 2.

Algorithm 2 Geometry-aware Sparse Networks

Input: Input sparse voxel features F, the number of blocks B, and voxel index V^s at the target scale s
Output: Semantic prediction P

$L = []$
$Loss = 0$
for $i = 1$ to B **do**
  $V \leftarrow \text{SFE}(F)$
  $F \leftarrow \text{SGFE}(\mathcal{V})$
  $O^s \leftarrow \Phi(F, \mathcal{V}^s)$
  $Loss$ += LossFunc($\mathcal{V}$)
  L.append(O^s)
end for
$L \leftarrow \text{Stack}(L)$
$P = \text{Softmax}(\text{MLP}(L))$
return P

4 Experiments

We conduct experiments on large-scale outdoor scenarios dataset SemanticKITTI [1] and Nuscenes [3] to show the effectiveness of our proposed approaches. Besides that, we also conduct ablation studies to validate proposed components.

4.1 Datasets

SemanticKITTI SemanticKITTI [1] generated from KITTI odometry dataset [9] contains 22 sequences which involve the most common scenes for autonomous driving. Each scan in the dataset has more than 100K points on average, with pointwise annotation labels for 20 classes. According to the official settings, sequences from 00 to 10 except 08 are the training split, sequence 08 is the validation split, and the rest sequences from 11 to 21 are treated as test split.
Nuscenes Nuscenes [3] has a total of 40,000 scans collected by 32 beams Lidar sensor. Compared with SemanticKITTI [1], it contains fewer points and annotations classes(16 classes).

For both datasets, they use **mIoU** as one evaluation metric, which is one of the most popular criteria in point cloud semantic segmentation. Apart from that, SemanticKITTI provides Acc, the average accuracy metric, and Nuscenes uses **fwIoU** as additional criteria, which is the weighted sum of IoU for each class based on the point-level frequency.
Network Details. We use the same settings for both datasets. We quantize the point cloud with a voxel scale $0.2m$ along xyz dimensions to generate the initial sparse voxel features. We design our Geometry-aware Sparse Networks with four blocks of Sparse Feature Encoder and Sparse Geometry Feature Enhancement. In the ablation study, we also evaluate the effect of block number choice between performance and efficiency. As for the multi-scale sparse projection layer, we adopt kernel sizes and strides with $[2, 4, 6, 8]$, which could cover the coarse and

Table 1. The per-class mIoU results on the SemanticKITTI test set.

Methods	road	sidewalk	parking	other ground	building	car	truck	bicycle	motorcycle	other vehicle	vegetation	trunk	terrain	person	bicyclist	motorcyclist	fence	pole	traffic sign	mIoU	speed (ms)
PointNet [29]	61.6	35.7	15.8	1.4	41.4	46.3	0.1	1.3	0.3	0.8	31.0	4.6	17.6	0.2	0.2	0.0	12.9	2.4	3.7	14.6	500
PointNet++ [30]	72.0	41.8	18.7	5.6	62.3	53.7	0.9	1.9	0.2	0.2	46.5	13.8	30.0	0.9	1.0	0.0	16.9	6.0	8.9	20.1	5900
KPConv [38]	88.8	72.7	61.3	31.6	90.5	96.0	33.4	30.2	42.5	44.3	84.8	69.2	69.1	61.5	61.6	11.8	64.2	56.4	47.4	58.8	-
RandLA [14]	90.7	73.7	60.2	20.4	86.9	94.2	40.1	26.0	25.8	38.9	81.4	66.8	49.2	49.2	48.2	7.2	56.3	47.7	38.1	53.9	880
SqueezeSegV3 [46]	91.7	74.8	63.4	26.4	89.0	92.5	29.6	38.7	36.5	33.0	82.0	58.7	65.4	45.6	46.2	20.1	59.4	49.6	58.9	55.9	238
RangeNet++ [27]	91.8	75.2	65.0	27.8	87.4	91.4	25.7	25.7	34.4	23.0	80.5	55.1	64.6	38.3	38.8	4.8	58.6	47.9	55.9	52.2	83.3
TangentConv [36]	83.9	63.9	33.4	15.4	83.4	90.8	15.2	2.7	16.5	12.1	79.5	49.3	58.1	23.0	28.4	8.1	49.0	35.8	28.5	35.9	3000
SPVCNN [35]	90.2	75.4	67.6	21.8	91.6	97.2	56.6	50.6	50.4	58.0	86.1	73.4	71.0	67.4	67.1	50.3	66.9	64.3	67.3	67.0	259
PolarNet [54]	90.8	74.4	61.7	21.7	90.0	93.8	22.9	40.3	30.1	28.5	84.0	65.5	67.8	43.2	40.2	5.6	61.3	51.8	57.5	54.3	62
DASS [39]	92.8	71.0	31.7	0.0	82.1	91.4	**66.7**	25.8	31.0	43.8	83.5	56.6	69.6	47.7	70.8	0.0	39.1	45.5	35.1	51.8	90
DRINet [52]	90.7	75.2	65.0	26.2	91.5	96.9	43.3	57.0	56.0	54.5	85.2	72.6	68.8	69.4	75.1	58.9	67.3	63.5	66.0	67.5	62
Cylinder3D [59]	92.0	70.0	65.0	32.3	90.7	97.1	50.8	67.6	63.8	58.5	85.6	72.5	69.8	73.7	69.2	48.0	66.5	62.4	66.2	68.9	131
AF2S3Net [6]	92.0	76.2	66.8	**45.8**	92.5	94.3	40.2	63.0	**81.4**	40.0	78.6	68.0	63.1	76.4	81.7	**77.7**	69.6	64.0	**73.3**	**70.8**	-
RPVNet [47]	**93.4**	**80.7**	**70.3**	33.3	**93.5**	**97.6**	44.2	**68.4**	68.7	**61.1**	86.5	**75.1**	71.7	75.9	74.4	43.4	**72.1**	64.8	61.4	70.3	168
Ours	89.8	74.6	66.2	30.1	92.3	96.9	59.3	65.8	58.0	61.0	**87.3**	73.0	**72.5**	**80.4**	**82.7**	46.3	69.6	**66.1**	71.6	70.7	**59**

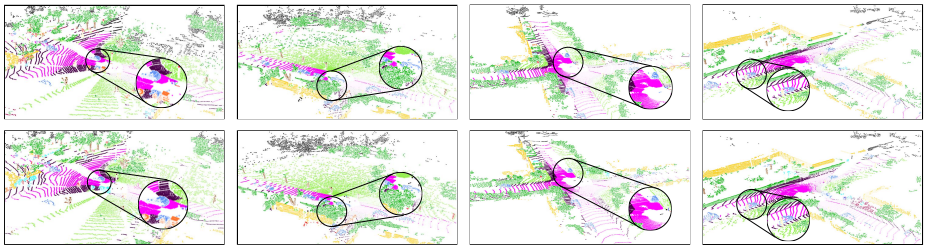

Fig. 4. The results on the SemanticKITTI validation set. The top row is the ground truth, and the bottom row is the predictions by our Geometry-aware Sparse Networks.

Table 2. Quantitative results of model complexity with performance. The performance is obtained from the leaderboard of SemanticKITTI. Statistics of the number of parameters and Macs are from the corresponding papers. The running times are all evaluated on a single Nvidia RTX 2080Ti GPU. $*$ donates the statistics from our reproduction.

Method	Param (M)	Macs (G)	Speed	Memory	mIoU
SPVCNN	12.5	73.8	120 ms	2.4 Gb	67.0
DRINet	3.5	14.58	62* ms	2.1* Gb	67.5
Cylind3D	53.3	64.3	131 ms	3.0 Gb	68.9
RPVNet	24.8	119.5	168* ms	2.7* Gb	70.3
Ours	**2.2**	**12.1**	**59 ms**	**1.6 Gb**	**70.7**

the fine pooling regions. Similar to previous works, we apply random flipping, random points dropout, random scale, and global rotation in the training stage. For the loss design, we combine the Lovasz loss [2] and cross-entropy loss as supervision. Adam optimizer [15] is employed with an initial learning rate of 0.0002 at batch size 4 for 50 epochs. The learning rate decays in a ratio of 0.1 for every 15 epochs.

4.2 Results on SemanticKITTI

SemanticKITTI Single Scan: We provide the detailed per-class quantitative results of our Geometry-aware Sparse Networks as well as the other state-of-the-art methods in Tab. 1. Compared with previous methods, our GASN achieves state-of-the-art performance while maintaining a real-time inference efficiency. Although our Geometry-aware Sparse Networks fails to achieve the best result for every class, it achieves balanced results among all the classes. Even compared to multiple representation fusion approaches [52, 35, 47], our method still surpasses by a considerable margin.

Furthermore, we also provide quantitative analysis for the model complexity and latency for some state-of-the-art methods and our GASN in Tab. 2 to illustrate the high performance-time ratio of our approach. Compared with previous methods, we achieve the best mIoU result with the least computation cost, demonstrating the efficiency and effectiveness of our approach. Some quantitative results on the SemanticKITTI validation set are shown in Fig. 4.

Table 3. Comparison to the state-of-the-art methods on the test set of SemanticKITTI multiple scans challenge. $*$ donates the moving category.

Approach	road	sidewalk	parking	other ground	building	car	car*	truck	truck*	bicycle	motorcycle	other vehicle	other vehicle*	vegetation	trunk	terrain	person	person*	bicyclist	bicyclist*	motorcyclist	motorcyclist*	fence	pole	traffic sign	mIoU
Cylinder3D	90.7	74.5	65.0	32.3	92.6	94.6	74.9	41.3	0.0	**67.6**	63.8	38.8	0.1	85.8	72.0	68.9	12.5	65.7	1.7	68.3	0.2	11.9	66.0	63.1	61.4	52.5
TemporalLidarSeg	91.8	75.8	59.6	23.2	89.8	92.1	68.2	39.2	2.1	47.7	40.9	35.0	12.4	82.3	62.5	64.7	14.4	40.4	0.0	42.8	0.0	12.9	63.8	52.6	60.4	47.0
KPConv	86.5	70.5	58.4	26.7	90.8	93.7	69.4	42.5	5.8	44.9	47.2	38.6	4.7	84.6	70.3	66.0	21.6	67.5	0.0	67.4	0.0	47.2	64.5	57.0	53.9	51.2
AF2S3Net	91.3	72.5	68.8	**53.5**	87.9	91.8	65.3	15.7	5.6	65.4	**86.8**	27.5	3.9	75.1	64.6	57.4	16.4	67.6	**15.1**	66.4	**67.1**	59.6	63.2	62.6	71.0	56.9
Ours	**92.3**	**79.1**	**69.6**	30.9	**93.7**	**97.2**	**85.4**	**46.8**	**15.9**	63.6	53.3	**64.6**	**26.3**	**86.8**	**75.8**	**71.2**	**30.5**	**84.8**	0.0	**73.1**	0.0	**76.9**	**72.8**	**68.0**	**73.5**	**61.3**

SemanticKITTI Multiple Scans: Meanwhile, we also conduct experiments on SemanticKITTI multiple scans challenge to verify the effectiveness of our GASN. In this task, we directly stack multiple aligned scans as input without any temporal fusion algorithm, then generate the pointwise output prediction according to Algo. 2. We do not apply any post-processing for refinement. As shown in Tab. 3, our Geometry-aware Sparse Network shows a significant improvement compared with previous methods in terms of both metrics. Compared with AF2S3Net [6], which is also a voxel-based approach, the proposed method brings a very competitive gain (about 4.4%). Our method surpasses the existing approaches in nearly all categories, demonstrating the effectiveness and generalization capability of our method. To this end, GASN can serve as an efficient and strong backbone for the large-scale point cloud semantic segmentation task.

4.3 Results on Nuscenes

In order to verify the generalization ability of our approach, we also report the results on Nuscenes-lidarSeg [3] task on the test set. As shown in Tab. 4,

Table 4. The per-class mIoU results on the Nuscenes test set.

Method	barrier	bicycle	bus	car	construction	motorcycle	pedestrian	traffic cone	trailer	truck	driveable	other_flat	sidewalk	terrain	manmade	vegetation	FW mIoU	mIoU
AF2S3Net [6]	78.9	**52.2**	89.9	84.2	**77.4**	74.3	77.3	72.0	83.9	73.8	97.1	66.5	77.5	74.0	87.7	86.8	88.5	78.3
Cylinder3D [59]	82.8	29.8	84.3	89.4	63.0	79.3	77.2	**73.4**	**84.6**	69.1	**97.7**	70.2	80.3	75.5	90.4	87.6	89.9	77.2
SPVCNN [35]	80.0	30.0	**91.9**	90.8	64.7	79.0	75.6	70.9	81.0	74.6	97.4	69.2	80.0	76.1	89.3	87.1	89.7	77.4
PolarNet [54]	72.2	16.8	77.0	86.5	51.1	69.7	64.8	54.1	69.7	63.5	96.6	67.1	77.7	72.1	87.1	84.5	87.4	69.4
Ours	**85.5**	43.2	90.5	**92.1**	64.7	**86.0**	**83.0**	73.3	83.9	**75.8**	97.0	**71.0**	**81.0**	**77.7**	**91.6**	**90.2**	**91.0**	**80.4**

GASN achieves highly strong results, with 10 out of 16 categories surpassing the other approaches. There is a noticeable improvement for classes with small sizes, such as pedestrians and motorcycles, demonstrating the effectiveness of our sparse geometry feature enhancement, which is designed to capture and integrate multi-scale context information through hierarchical feature learning and attentive scale selection.

4.4 Ablation Study

Components Study. Our baseline model, shown in Tab. 5, which only uses a sparse feature encoder, has achieved decent performance. It reveals that the voxel representation with sparse convolution has a strong capability for feature extraction and context information learning in the outdoor point cloud semantic segmentation task. The multi-scale sparse projection layer brings 1.8% improvement with its ability to capture hierarchical geometry information. Furtherly, we apply the attentive scale selection to re-distribute the importance of each scale for each channel. This strategy enables the network the focus on a more significant scale and has a performance gain of about 1.9%. Next, adding the deep sparse supervision in the training stage improves the mIoU to 69.4%, an increase of 1.7%. More importantly, deep sparse supervision will be disabled when inference, bringing no extra computation cost for deployment.

Table 5. Ablation study on the SemanticKITTI validation set. MSP refers to Multi-scale Sparse Projection. ASS refers to Attentive Scale Selection. DSS refers to Deep Sparse Supervision.

SFE	SGFE		DSS	mIoU (%)
	MSP	ASS		
✓				64.0
✓	✓			65.8
✓	✓	✓		67.7
✓	✓	✓	✓	69.4

Number of Blocks. The block number choice plays a crucial role in our network. Fewer blocks may lead to underfitting problems, while more blocks mean more computation cost and memory consumption, indicating inefficiency. In order to find a better block number, we conduct the experiment varying this parameter. With block number ranging from 1 to 5, the performance is 50.8%, 61.3%, 67.4%, 69.4% and 69.9%, while the speed will be $26ms$, $33ms$, $41ms$, $59ms$ and $70ms$ respectively. The performance increases from 50.8% to 70.2% when the block number rises to 5, while the runtime speed nearly doubles. Furthermore, adding one more block when the block number is 4 introduces extra $11ms$ latency while only improving the mIoU by about 0.5%. Therefore, we choose block number as 4 in our experiment.

Table 6. Ablation study for fusion strategy. TC refers to concatenation and TS refers to Tensor sum.

Method	mIoU (%)	Memory	Speed
ASS	**69.4**	1.6Gb	59ms
TS	67.7	**1.45Gb**	**57ms**
TC	68.7	1.48Gb	**57ms**

Scale Features Fusion Strategy. We analyze the effectiveness of the proposed ASS by comparing it with different fusion strategies. We use tensor concatenation and summation, which are commonly used in feature fusion. For this work, we enable all other proposed modules for a fair comparison. As shown in Tab. 6, the ASS layer serves as a better approach for fusing the multi-scale features, which leads to a 2.0% increase compared with the tensor summation while only bringing 2ms cost. Tab. 7 also shows that other models that utilize multiple representations could benefit from the ASS strategy. Compared with DRINet [52], the original SPVCNN [35] does not have hierarchical learning in the pointwise branches, and then its performance will improve when ASS is enabled.

Table 7. Ablation study for DSS and ASS on the SemanticKITTI [1] validation set with different models. The statistics are from our reproduction.

Method	Original (%)	DSS (%)	ASS (%)
SPVCNN [35]	64.7	66.1 (**+1.4**)	66.7 (**+2.0**)
DRINet [52]	67.3	68.1 (**+0.8**)	67.9 (**+0.6**)
Cylinder3D [59]	66.5	67.8 (**+1.3**)	-

Sparse or Dense Supervision. Deep Sparse supervision is another characteristic of our GASN. In this experiment, we compare the sparse and dense super-

vision in terms of memory cost and performance. We remove the deep auxiliary loss branch for simplicity, with only the main supervision left. The memory consumption for both sparse and dense supervision only includes prediction tensor and label tensor without the consumption used by gradient tensors. The memory footprint in sparse supervision (6Mb) only accounts for about one percent of that in dense supervision (552Mb), and the results are close between the methods, which are 69.4% and 69.5% respectively. By incorporating sparse supervision, we can use a larger batch size for training, leading to efficient training.

Deep Sparse Supervision. Our Deep Sparse Supervision can be a general component in the point cloud semantic segmentation task, and we incorporate this strategy with other popular models to verify its effectiveness. As shown in Tab. 7, Deep Sparse Supervision could help the performance of popular models without any extra computation cost for inference.

Table 8. Ablation study on SemanticKITTI validation set for voxel size choice.

Voxel Size (m)	mIoU (%)	Memory	Speed (ms)
0.2	**69.4**	1.6Gb	59
0.3	68.3	1.42Gb	47
0.4	67.1	1.31Gb	42
0.5	64.0	**1.25Gb**	**39**

Voxel Size. To choose the best voxel size for the experiments, we also conduct experiments to verify the effect. As shown in Tab. 8, the runtime speed, memory consumption, and performance drop significantly with a larger voxel size. As a result, we choose voxel size as $0.2m$ in our experiments.

5 Conclusion

In this paper, we propose our Geometry-aware Sparse Networks that serve as an efficient network architecture for point cloud segmentation. Our GASN deals with point cloud segmentation by fully utilizing the sparsity and geometry in a single sparse voxel representation to maintain performance and efficiency. GASN consists of Sparse Feature Encoder (SFE) and Sparse Geometry Feature Enhancement (SGFE). SFE helps extract local context information, while SGFE enhances the geometry with multi-scale sparse projection and attentive scale selection. Moreover, we apply deep sparse supervision to accelerate convergence with a lower memory cost. The experiments on large-scale outdoor scenarios datasets demonstrate that our approach achieves state-of-the-art performance with impressive runtime efficiency.

References

1. Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J.: Semantickitti: A dataset for semantic scene understanding of lidar sequences. In: ICCV. pp. 9297–9307 (2019)
2. Berman, M., Rannen Triki, A., Blaschko, M.B.: The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In: CVPR. pp. 4413–4421 (2018)
3. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: CVPR. pp. 11621–11631 (2020)
4. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence **40**(4), 834–848 (2017)
5. Chen, L.C., Papandreou, G., Schroff, F., Adam, H.: Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587 (2017)
6. Cheng, R., Razani, R., Taghavi, E., Li, E., Liu, B.: 2-s3net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network. In: CVPR. pp. 12547–12556 (2021)
7. Choy, C., Gwak, J., Savarese, S.: 4d spatio-temporal convnets: Minkowski convolutional neural networks. In: CVPR. pp. 3075–3084 (2019)
8. Cortinhal, T., Tzelepis, G., Aksoy, E.E.: Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds. In: International Symposium on Visual Computing. pp. 207–222. Springer (2020)
9. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. The International Journal of Robotics Research **32**(11), 1231–1237 (2013)
10. Graham, B., Engelcke, M., Maaten, L.V.D.: 3d semantic segmentation with submanifold sparse convolutional networks. In: CVPR (2018)
11. Guo, M.H., Cai, J.X., Liu, Z.N., Mu, T.J., Martin, R.R., Hu, S.M.: Pct: Point cloud transformer. arXiv preprint arXiv:2012.09688 (2020)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
13. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: CVPR. pp. 7132–7141 (2018)
14. Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., Markham, A.: Randla-net: Efficient semantic segmentation of large-scale point clouds. In: CVPR. pp. 11108–11117 (2020)
15. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
16. Kumawat, S., Raman, S.: Lp-3dcnn: Unveiling local phase in 3d convolutional neural networks. In: CVPR. pp. 4903–4912 (2019)
17. Le, T., Duan, Y.: Pointgrid: A deep network for 3d shape understanding. In: CVPR. pp. 9204–9214 (2018)
18. Li, X., Wang, W., Hu, X., Yang, J.: Selective kernel networks. In: CVPR. pp. 510–519 (2019)
19. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: Pointcnn: Convolution on x-transformed points. Advances in neural information processing systems **31**, 820–830 (2018)

20. Liu, Z., Hu, H., Cao, Y., Zhang, Z., Tong, X.: A closer look at local aggregation operators in point cloud analysis. In: ECCV. pp. 326–342. Springer (2020)
21. Liu, Z., Tang, H., Lin, Y., Han, S.: Point-voxel cnn for efficient 3d deep learning. In: Advances in Neural Information Processing Systems. pp. 965–975 (2019)
22. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR. pp. 3431–3440 (2015)
23. Loquercio, A., Dosovitskiy, A., Scaramuzza, D.: Learning depth with very sparse supervision. IEEE Robotics and Automation Letters **5**(4), 5542–5549 (2020)
24. Maas, A.L., Hannun, A.Y., Ng, A.Y., et al.: Rectifier nonlinearities improve neural network acoustic models. In: Proc. icml. vol. 30, p. 3. Citeseer (2013)
25. Maturana, D., Scherer, S.: Voxnet: A 3d convolutional neural network for real-time object recognition. In: IROS. pp. 922–928. IEEE (2015)
26. Milioto, A., Vizzo, I., Behley, J., Stachniss, C.: Rangenet++: Fast and accurate lidar semantic segmentation. In: IROS. pp. 4213–4220. IEEE (2019)
27. Milioto, A., Vizzo, I., Behley, J., Stachniss, C.: Rangenet++: Fast and accurate lidar semantic segmentation. In: IROS. pp. 4213–4220. IEEE (2019)
28. Pham, Q.H., Nguyen, T., Hua, B.S., Roig, G., Yeung, S.K.: Jsis3d: Joint semantic-instance segmentation of 3d point clouds with multi-task pointwise networks and multi-value conditional random fields. In: CVPR. pp. 8827–8836 (2019)
29. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: CVPR. pp. 652–660 (2017)
30. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In: Advances in Neural Information Processing Systems 30. vol. 30, pp. 5099–5108. Curran Associates, Inc. (2017)
31. Qi, X., Liao, R., Jia, J., Fidler, S., Urtasun, R.: 3d graph neural networks for rgbd semantic segmentation. In: ICCV. pp. 5199–5208 (2017)
32. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)
33. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.: Multi-view convolutional neural networks for 3d shape recognition. In: ICCV. pp. 945–953 (2015)
34. Sun, K., Xiao, B., Liu, D., Wang, J.: Deep high-resolution representation learning for human pose estimation. In: CVPR. pp. 5693–5703 (2019)
35. Tang, H., Liu, Z., Zhao, S., Lin, Y., Lin, J., Wang, H., Han, S.: Searching efficient 3d architectures with sparse point-voxel convolution. In: ECCV. pp. 685–702. Springer (2020)
36. Tatarchenko, M., Park, J., Koltun, V., Zhou, Q.Y.: Tangent convolutions for dense prediction in 3d. In: CVPR. pp. 3887–3896 (2018)
37. Te, G., Hu, W., Zheng, A., Guo, Z.: Rgcnn: Regularized graph cnn for point cloud segmentation. In: Proceedings of the 26th ACM international conference on Multimedia. pp. 746–754 (2018)
38. Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: Kpconv: Flexible and deformable convolution for point clouds. In: ICCV. pp. 6411–6420 (2019)
39. Unal, O., Van Gool, L., Dai, D.: Improving point cloud semantic segmentation by learning 3d object detection. In: WACV. pp. 2950–2959 (2021)
40. Wang, C., Samari, B., Siddiqi, K.: Local spectral graph convolution for point set feature learning. In: ECCV. pp. 52–66 (2018)
41. Wang, L., Huang, Y., Hou, Y., Zhang, S., Shan, J.: Graph attention convolution for point cloud semantic segmentation. In: CVPR. pp. 10296–10305 (2019)

42. Wang, P.S., Liu, Y., Guo, Y.X., Sun, C.Y., Tong, X.: O-cnn: Octree-based convolutional neural networks for 3d shape analysis. ACM Transactions On Graphics (TOG) **36**(4), 1–11 (2017)
43. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. Acm Transactions On Graphics (tog) **38**(5), 1–12 (2019)
44. Wu, B., Wan, A., Yue, X., Keutzer, K.: Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In: ICRA. pp. 1887–1893. IEEE (2018)
45. Wu, W., Qi, Z., Fuxin, L.: Pointconv: Deep convolutional networks on 3d point clouds. In: CVPR. pp. 9621–9630 (2019)
46. Xu, C., Wu, B., Wang, Z., Zhan, W., Vajda, P., Keutzer, K., Tomizuka, M.: Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation. In: ECCV. pp. 1–19. Springer (2020)
47. Xu, J., Zhang, R., Dou, J., Zhu, Y., Sun, J., Pu, S.: Rpvnet: A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation. arXiv preprint arXiv:2103.12978 (2021)
48. Xu, Q., Sun, X., Wu, C.Y., Wang, P., Neumann, U.: Grid-gcn for fast and scalable point cloud learning. In: CVPR. pp. 5661–5670 (2020)
49. Yan, X., Gao, J., Li, J., Zhang, R., Li, Z., Huang, R., Cui, S.: Sparse single sweep lidar point cloud segmentation via learning contextual shape priors from scene completion. arXiv preprint arXiv:2012.03762 (2020)
50. Yan, Y., Mao, Y., Li, B.: Second: Sparsely embedded convolutional detection. Sensors **18**(10), 3337 (2018)
51. Yang, M., Yu, K., Zhang, C., Li, Z., Yang, K.: Denseaspp for semantic segmentation in street scenes. In: CVPR. pp. 3684–3692 (2018)
52. Ye, M., Xu, S., Cao, T., Chen, Q.: Drinet: A dual-representation iterative learning network for point cloud segmentation. In: ICCV (2021)
53. Zhang, K., Hao, M., Wang, J., de Silva, C.W., Fu, C.: Linked dynamic graph cnn: Learning on point cloud via linking hierarchical features. arXiv preprint arXiv:1904.10014 (2019)
54. Zhang, Y., Zhou, Z., David, P., Yue, X., Xi, Z., Gong, B., Foroosh, H.: Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In: CVPR. pp. 9601–9610 (2020)
55. Zhang, Y., Rabbat, M.: A graph-cnn for 3d point cloud classification. In: 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 6279–6283. IEEE (2018)
56. Zhao, H., Jiang, L., Jia, J., Torr, P., Koltun, V.: Point transformer. arXiv preprint arXiv:2012.09164 (2020)
57. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: CVPR. pp. 2881–2890 (2017)
58. Zhou, Y., Sun, P., Zhang, Y., Anguelov, D., Gao, J., Ouyang, T., Guo, J., Ngiam, J., Vasudevan, V.: End-to-end multi-view fusion for 3d object detection in lidar point clouds. In: Conference on Robot Learning. pp. 923–932. PMLR (2020)
59. Zhu, X., Zhou, H., Wang, T., Hong, F., Ma, Y., Li, W., Li, H., Lin, D.: Cylindrical and asymmetrical 3d convolution networks for lidar segmentation. In: CVPR. pp. 9939–9948 (2021)