CPGNet: Cascade Point-Grid Fusion Network for Real-Time LiDAR Semantic Segmentation

Xiaoyan Li*,+, Gang Zhang*,×, Hongyu Pan×, Zhenhua Wang×

***Abstract*— LiDAR semantic segmentation essential for advanced autonomous driving is required to be accurate, fast, and easy-deployed on mobile platforms. Previous point-based or sparse voxel-based methods are far away from real-time applications since time-consuming neighbor searching or sparse 3D convolution are employed. Recent 2D projection-based methods, including range view and multi-view fusion, can run in real time, but suffer from lower accuracy due to information loss during the 2D projection. Besides, to improve the performance, previous methods usually adopt test time augmentation (TTA), which further slows down the inference process. To achieve a better speed-accuracy trade-off, we propose Cascade Point-Grid Fusion Network (CPGNet), which ensures both effectiveness and efficiency mainly by the following two techniques: 1) the novel Point-Grid (PG) fusion block extracts semantic features mainly on the 2D projected grid for efficiency, while summarizes both 2D and 3D features on 3D point for minimal information loss; 2) the proposed transformation consistency loss narrows the gap between the single-time model inference and TTA. The experiments on the SemanticKITTI and nuScenes benchmarks demonstrate that the CPGNet without ensemble models or TTA is comparable with the state-of-the-art RPVNet, while it runs** 4.7 **times faster.**

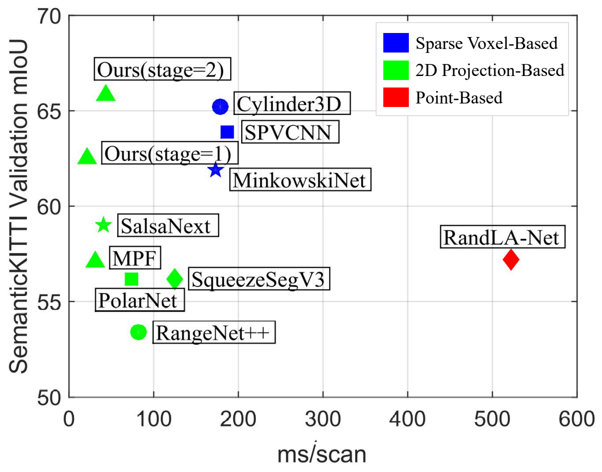


Fig. 1. Accuracy (mIoU) vs. runtime on the SemanticKITTI validation set. For fair comparison, all methods are trained based on the official code and evaluated without ensemble models or TTA. The experiments are conducted with PyTorch FP32 on NVIDIA RTX 2080Ti GPU.

I. INTRODUCTION

Light Detection and Ranging (LiDAR) sensors are widely used in autonomous driving and robotics. The 3D point cloud data they captured provide rich information about the surrounding scene. LiDAR semantic segmentation assigns semantic labels for these 3D point clouds, such as car, pedestrian, cyclist, road, building, and thus directly bears on driving accuracy and safety. In the last few years, various deep learning models have been proposed to process LiDAR 3D point cloud, but these methods can not guarantee accuracy and speed simultaneously, especially on mobile platforms (*e.g.* cars and robots).

Previous methods on LiDAR 3D point cloud can be grouped into three categories: point-based methods, sparse voxel-based methods, and 2D projection-based methods.

The point-based methods include PointNet [1], PointNet++ [2], PointCNN [3], RandLA-Net [4] and *etc*. They are directly applied on the raw unordered 3D point cloud without any information loss. However, these methods usually adopt time-consuming operations, namely farthest point sampling (FPS) for uniformly down-sampling, and k-nearest neighbor (kNN) or ball query for local neighbor searching.

+University of Chinese Academy of Sciences, Beijing 100049, China. xiaoyan.li@vipl.ict.ac.cn
×Damo Academy, Alibaba Group. zhanggang11021136@gmail.com hongyu.pan@alibaba-inc.com zhwang.me@gmail.com
* equal contribution

Since LiDAR 3D points are hugely sparse, the sparse voxel-based methods quantize the 3D points to voxels and then apply 3D convolution operation only on these sparse voxels. Although these methods have inevitable information loss due to quantization, they achieve the state-of-the-art performance. However, these methods are computationally expensive, and can not run in real-time.

The 2D projection-based methods apply mature 2D CNN on the 2D grid feature map projected by the 3D point cloud. Inspired by Fully Convolution Network (FCN) [5] and its variants [6], [7], [8], [9], [10] that dominate 2D image semantic segmentation, these 2D projection-based methods, usually considering bird's-eye view [11] or range view [12], can be easily designed and deployed on some efficient deep learning inference frameworks (*e.g.* TensorRT [13]). They can achieve almost 15 ms per LiDAR scan. However, these methods have lower accuracy due to serious 2D projection information loss. They still perform worse, even if the following RangeNet++ [14] attempts to use kNN as post-process, and MPF [15] combines both bird's-eye view and range view.

To this end, we propose the CPGNet for real-time accurate LiDAR semantic segmentation. The proposed PG fusion block in CPGNet first projects and extracts semantic features on the 2D grids of both bird's-eye view and range view, and then transmits and fuses these features onto the 3D point. As can be seen, the PG fusion block combines the advantages of both point-based methods with complete information and

2D projection-based methods with fast speed. The CPGNet applies PG fusion block repeatedly to further enhance point features. Besides, inspired by test time augmentation (TTA), the transformation consistency loss is proposed to ensure agreements between the results of original and augmented point clouds. Finally, we compare CPGNet with the open-source methods, as shown in Fig. 1, and CPGNet achieves the best mIoU (65.9) on SemanticKITTI [16] validation set, while it runs 43 ms with PyTorch FP32 on NVIDIA RTX 2080Ti GPU. The contributions can be listed as follows:

- We present an accurate, fast, and easy-deployed CPGNet for LiDAR semantic segmentation. It fuses point, bird's-eye view and range view features in a cascade framework.
- We propose transformation consistency loss inspired by test time augmentation (TTA), and enable higher performance with only single-time inference.
- The proposed CPGNet achieves the best speed-accuracy trade-off on SemanticKITTI and nuScenes benchmarks.

II. RELATED WORK

Unlike 2D images with dense grid structures, point clouds are unordered, sparse, and unstructured, which make it difficult to apply deep learning operations (*e.g.* convolution). Previous methods attempt to solve this problem in three ways.

A. Point-Based Methods

Point-based methods directly operate on the raw points. The PointNet [1] applies shared Multi-Layer Perceptron (MLP) on each point and max pooling among the whole points to acquire point-level features for further segmentation task. However, PointNet performs worse on the complicated scene for the lack of local context extraction. The following work [2], [3] propose ball query and χ-Conv to mimic 2D convolution, and achieve great results on the indoor scene. However, they cannot be applied on the LiDAR point cloud due to computation and memory cost. To accelerate the network inference, RandLA-Net [4] adopts random sampling and local feature aggregation, but it suffers from lower accuracy due to random sampling. KPConv [17] proposes a novel spatial kernel-based point convolution to extract local structure, and KPRNet [18] combines KPConv and ResNext [19] to achieve the best results of point-based methods. Although point-based methods are directly applied on the raw points without dropping the information out, it's less studied in autonomous driving due to its inefficient local structure extraction.

B. Sparse Voxel-Based Methods

Sparse voxel-based methods quantize the 3D points into sparse voxels, and then apply 3D convolution operation only on those non-empty voxels to reduce computation and memory cost. Minkowski CNN [20] is the first efficient sparse voxel framework, and it surpasses all point-based methods in both accuracy and speed. A possible reason is that sparse voxel is structured, which is convenient for convolution operation. SPVNAS [21] introduces neural architecture search (NAS) into [20], and achieves better results with lower computation cost. Recently, variants [22], [23], [24] of sparse voxel-based methods are proposed. Cylinder3D [22] quantizes the 3D points in the cylindrical coordinate system, and proves its efficiency. AF2S3Net [23] proposes the attentive feature fusion module (AF2M) and adaptive feature selection module (AFSM) to efficiently extract local and global structures simultaneously. RPVNet [24] fuses range view, point, and sparse voxel features in a single framework to alleviate quantization error, and achieves the best results on the SemanticKITTI and nuScenes benchmarks. Although these methods dominate the LiDAR semantic segmentation benchmarks, they have difficulty in deployment and cannot run in real-time on mobile platforms.

C. 2D Projection-Based Methods

Recently 2D projection-based methods attract more attentions because they are fast and easy-deployed. These methods utilize 2D FCN by projecting 3D points onto the 2D grid and primarily include range view and multi-view fusion. Range view projects 3D points onto the 2D spherical grid and has many variants. RangeNet++ [14] proposes an accelerated kNN for post-process, which is an indispensable module in the following range-based methods. SqueezeSegV3 [25] proves the superiority of the spatially adaptive convolution. SalsaNext [26] designs a novel encoder-decoder network based on SalsaNet [27], and adopts Lovász-Softmax loss [28] that can directly optimize the mean Intersection over Union (mIoU) metric. More recently, Lite-HDSeg [29] proposes harmonic dense convolutions and achieves the best results among range-based methods. Since the single view or 2D grid has inevitable 2D projection information loss, the following multi-view fusion projects the 3D points to two or more different types of 2D grid. MPF [15] and AMVNet [30] both combine bird's-eye view and range view. Different from the proposed method, they conduct semantic segmentation independently on each view and fuse the segmentation results from the two views by a late-fusion module.

III. PROPOSED METHOD

For accurate and fast LiDAR semantic segmentation, it needs to not only extract semantic features in an efficient way but also keep complete point cloud information. Therefore, we propose CPGNet which progressively extracts point features by the Point-Grid (PG) fusion block. As shown in Fig. 2, the PG fusion block consists of four steps: 1) the Point to Grid (**P2G**) operation projects the input point features onto bird's-eye view and range view feature maps respectively; 2) the 2D FCN is applied on the 2D feature maps to extract semantic features efficiently; 3) the Grid to Point (**G2P**) operation transmits the 2D grid features onto the 3D point; 4) the Point Fusion fuses features from the 3D point, bird's-eye view and range view branches to ensure complete point cloud information. As can be seen, CPGNet combines the advantages of both point-based and 2D projection-based

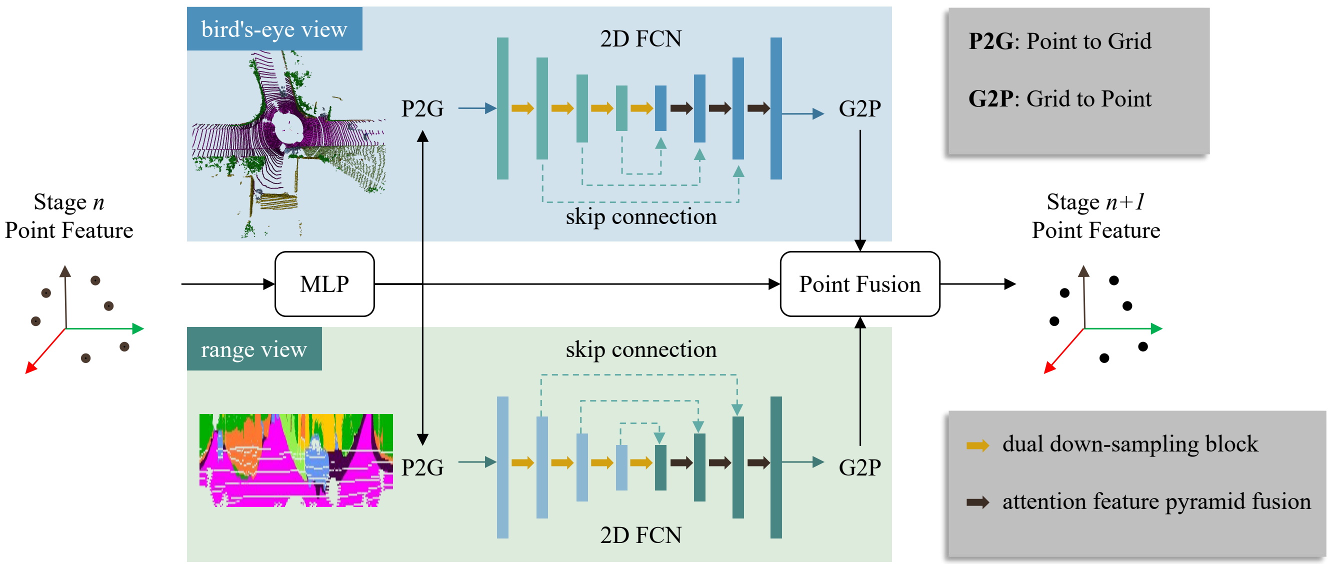


Fig. 2. The Point-Grid (PG) fusion block. It takes the point features from the last PG fusion block as input and undergoes the point, bird's-eye view and range view branches, respectively. The output point features are acquired by fusing features from the three branches.

methods. The components of CPGNet, except **P2G** and **G2P**, can be directly deployed on TensorRT. The **P2G** and **G2P** operations can be implemented via efficient CUDA code. Each PG fusion block of CPGNet shares the same network architecture, but does not share parameters. The details are described in the subsections below.

A. *Point to Grid*

The Point to Grid (**P2G**) operation aims to transform 3D point features to 2D grid feature maps. As shown in Fig. 3a, it first projects the k^{th} 3D point $\boldsymbol{p}_k^{3D} = (x_k, y_k, z_k)$ onto 2D grid to acquire the corresponding 2D coordinates $\boldsymbol{p}_k^{2D} = (u_k, v_k)$. The set $\mathcal{R}_{h,w}$ contains the indices of points that fall in the same 2D grid (h, w), namely $\mathcal{R}_{h,w} = \{k | \lfloor u_k \rfloor = h; \lfloor v_k \rfloor = w\}$. Then the features $\boldsymbol{\mathcal{F}}_k^{3D}$ of point in $\mathcal{R}_{h,w}$ are gathered through max pooling to form the corresponding 2D grid features $\boldsymbol{\mathcal{G}}_{h,w}^{2D}$. The formula is as follows:

$$\boldsymbol{\mathcal{G}}_{h,w,c}^{2D} = \max_{k \in \mathcal{R}_{h,w}} \boldsymbol{\mathcal{F}}_{k,c}^{3D}. \tag{1}$$

There may be more than one points falling in a 2D grid. To avoid parallel conflicts, while processing the same 2D grid, it utilizes CUDA atomicMax function.

Both bird's-eye view and range view are used in the proposed method. Bird's-eye view omits z dimension, and range view discards r dimension. Therefore, the two views are complementary to alleviate 2D projection information loss. Actually, the two views use a similar **P2G** operation. They just differs in the way of 2D projection. For bird's-eye view, it projects the 3D point onto $x - y$ plane that is discretized using a rectangular 2D grid $(x_{min}, y_{min}, x_{max}, y_{max})$ with the predefined width W_{bev} and height H_{bev}, as summarized in the following equation,

$$\begin{pmatrix} u_k \\ \\ v_k \end{pmatrix} = \begin{pmatrix} \frac{x_k - x_{min}}{x_{max} - x_{min}} \times W_{bev} \\ \\ \frac{y_k - y_{min}}{y_{max} - y_{min}} \times H_{bev} \end{pmatrix}. \tag{2}$$

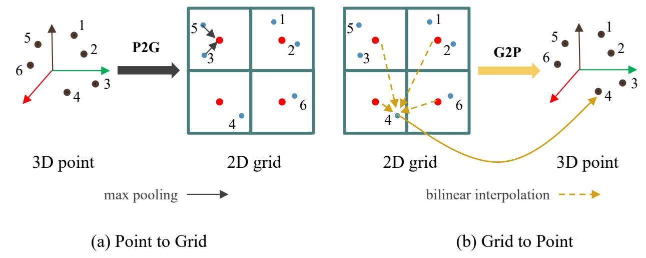


Fig. 3. The diagram of Point to Grid (a) and Grid to Point (b) operations.

For range view, the 3D point is mapped from the 3D cartesian space $\boldsymbol{p}_k^{3D} = (x_k, y_k, z_k)$ to the spherical space $\boldsymbol{p}_k^{sph} = (r_k, \theta_k, \phi_k)$ by applying

$$\begin{pmatrix} r_k \\ \\ \theta_k \\ \\ \phi_k \end{pmatrix} = \begin{pmatrix} \sqrt{{x_k}^2 + {y_k}^2 + {z_k}^2} \\ \\ \arcsin\left(\frac{z_k}{\sqrt{{x_k}^2 + {y_k}^2 + {z_k}^2}}\right) \\ \\ \arctan\left(y_k, x_k\right) \end{pmatrix}, \tag{3}$$

where r_k, θ_k, ϕ_k denote the distance, zenith and azimuth angle respectively. Subsequently, range view grid with the predefined width W_{rv} and height H_{rv} is acquired by discretizing θ_k and ϕ_k but ignoring r_k,

$$\begin{pmatrix} u_k \\ \\ v_k \end{pmatrix} = \begin{pmatrix} \frac{1}{2}[1 - \phi_k \pi^{-1}] W_{rv} \\ \\ [1 - (\theta_k + f_{up}) f^{-1}] H_{rv} \end{pmatrix}, \tag{4}$$

where $f = f_{up} + f_{down}$ is the LiDAR vertical field-of-view.

B. *2D FCN*

The 2D FCNs with the encoder and decoder architectures, are applied on the bird's-eye view and range view feature maps respectively to extract semantic features. They occupy more than 90% computation costs of the CPGNet. Therefore, the encoder network utilizes ResNet [31] with only 9 layers as a lightweight backbone, and has 128 maximum number of channels. To keep information during down-sampling, we propose the dual down-sampling block that uses 2D Convolution and 2D MaxPool for down-sampling in parallel, as shown in Fig. 4a. In the experiment, it demonstrates that the dual down-sampling block performs better with negligible latency.

The previous decoder architectures [10], [26] usually adopt feature pyramid fusion to fuse high-level and low-level feature maps. High-level feature maps contain more semantic information, while low-level feature maps show more details. For semantic segmentation, some parts (*e.g.* road, building) need high-level semantic features, and some parts (*e.g.* pedestrian, object boundary) require detailed features. Instead of using simple feature maps concatenation, we propose attention feature pyramid fusion to automatically select features from different levels, as shown in Fig. 4b.

C. Grid to Point

On the contrary to the Point to Grid (**P2G**) operation, the Grid to Point (**G2P**) transfers features from 2D grid to 3D point, according to the corresponding 2D coordinates $\boldsymbol{p}_k^{2D} = (u_k, v_k)$. As shown in Fig. 3b, it applies bilinear interpolation within the four neighbor grids. The formula is as follows:

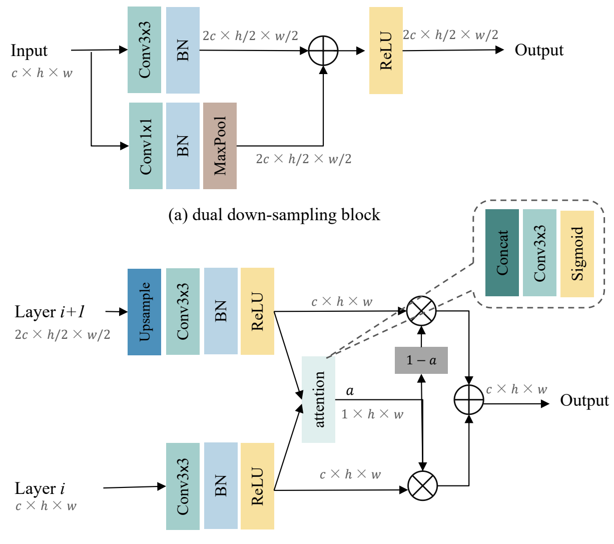
$$\mathcal{F}_{k,c}^{3D} = \sum_{i=0}^{1}\sum_{j=0}^{1} w_{i,j,k}\mathcal{G}_{\lfloor u_k \rfloor + i, \lfloor v_k \rfloor + j, c}^{2D}, \tag{5}$$

where $w_{i,j,k} = (1 - |u_k - (\lfloor u_k \rfloor + i)|)(1 - |v_k - (\lfloor v_k \rfloor + j)|)$ denotes the bilinear interpolation weight. Note that the neighbor grids beyond the 2D grid range are regarded as all zeros. As can be seen, each point and each feature channel are calculated independently, which is much suitable for the CUDA parallel computing.

D. Point Fusion

The Point Fusion module is responsible for fusing the features from the point, bird's-eye view and range view. For efficiency, it only adopts feature concatenation and two MLP layers. Different from MPF [15] and AMVNet [30], the Point Fusion does not conduct in post-process but serves as a mid-fusion module that is an important part of the proposed end-to-end CPGNet. The end-to-end framework has two advantages: 1) it is convenient to be deployed with rare post-process; 2) it can narrow the gap between the training and evaluation phases. The experiment section proves its superiority.

Though the features of the point that is projected out of the 2D grid range are regarded as all zeros in a certain view, it can pass information from the other view. For instance, the point that is beyond the range of bird's-eye view, but in the range of range view, has meaningful features from range view. In the experiment, we analyze the point distributions on both views and find that almost all points fall in the range of at least one view.



(a) dual down-sampling block

(b) attention feature pyramid fusion

Fig. 4. (a) dual down-sampling block. (b) attention feature pyramid fusion. Note that c, h and w denotes channels, height and width of the 2D feature map, respectively.

E. Loss Function

The segmentation predictions are acquired by applying a Fully Connection (FC) layer to the output features of the PG fusion block. The LiDAR semantic segmentation dataset (*e.g.* SemanticKITTI, nuScenes) has highly unbalanced categories. For instance, the proportions of $road$, $sidewalk$ and $building$ are hundreds times than that of $person$ and $motorcyclist$. To this end, we adopt weighted cross entropy (WCE) loss to manually emphasize rare categories. The WCE loss can be formulated as

$$\begin{aligned}\alpha_c &= \frac{1}{F_c + \epsilon}\\ \mathcal{L}_{wce} &= -\sum_{c=1}^{C} \alpha_c y_c \log(\hat{y}_c),\end{aligned} \tag{6}$$

where y_c defines the ground-truth label, $\hat{y}_c$ is the predicted probability, F_c is the frequency, and α_c is the weight of the c^{th} class. C is the category number of the dataset. In the experiment, ϵ is set as 0.001. We also adopt Lovász-Softmax loss [28] that can optimize the mean Intersection over Union (mIoU) metric as the second loss term $\mathcal{L}_{ls}$. As shown in [27], [29], it really improves the mIoU metric of segmentation task. More details can be seen in [28].

Previous methods [21], [22] adopt test time augmentation (TTA) to improve the performance, which needs model inference for multiple times. For both effectiveness and efficiency, we propose transformation consistency loss $\mathcal{L}_{tc}$ to reduce the differences between raw points and augmented points. The formula is as follows:

$$\mathcal{L}_{tc} = \sum_{c=1}^{C} |\hat{y}_c^{raw} - \hat{y}_c^{aug}|, \tag{7}$$

where $\hat{y}_c^{raw}$, $\hat{y}_c^{aug}$ are the predicted probability respectively from raw points and augmented points. The total loss $\mathcal{L}_{total}$ is the sum of the three loss terms and is defined as

$$\mathcal{L}_{total} = \mathcal{L}_{wce} + 2\mathcal{L}_{ls} + \mathcal{L}_{tc}. \tag{8}$$

IV. EXPERIMENTAL RESULTS

We evaluate the performance of the proposed CPGNet on SemanticKITTI [16] and nuScenes [32] benchmarks.

SemanticKITTI. It contains 43,552 360° LiDAR scans from 22 sequences collected in a city of Germany. Equipped with a Velodyne HDL-64E rotating LiDAR with 64 beams vertically, each LiDAR scan has approximately 130k points. The training set (19,130 scans) consists of sequence from 00 to 10 except 08, and the sequence 08 (4,071 scans) is used for validation. The rest sequences (20,351 scans) from 11 to 21 are only provided with LiDAR point clouds, and are used for online leaderboard. This dataset is labeled with 28 classes, but a high-level label set of 19 classes is used for single-scan LiDAR semantic segmentation.

nuScenes. It is a newly released dataset for LiDAR semantic segmentation with 1,000 scenes collected from different areas of Boston and Singapore. Each scene is 20s long collected by a Velodyne HDL-32E rotating LiDAR with 32 beams vertically. It splits 28,130 samples for training, 6,019 for validation and 6,008 for testing. It is annotated with 32 classes, and just 16 classes are used for official evaluation after merging some similar classes and removing rare classes.

Evaluation Metric. We adopt the most popular metric, mean Intersection over Union (mIoU), to evaluate the proposed CPGNet and its competitors. It can be formulated as

$$mIoU = \frac{1}{C}\sum_{c=1}^{C}\frac{TP_c}{TP_c + FP_c + FN_c}, \tag{9}$$

where TP_c, FP_c, FN_c are the true positive, false positive and false negative of c^{th} class respectively. C is the total number of classes.

A. Experimental Setup

Network Setup. As shown in Fig. 2, each PG fusion block of the CPGNet has a similar network architecture but different parameters. In the experiment, we adopt two cascade PG fusion blocks. The numbers of the input point feature channels for these two blocks are 9, 64, respectively. The input 9 channels of the first block refer to x, y, z, $intensity$, r, Δx, Δy, $\Delta\theta$, $\Delta\phi$, where Δx, Δy, $\Delta\theta$, $\Delta\phi$ denote the offsets from the corresponding 2D grid center. The first MLP layer of each block outputs 64 feature channels, which the following **P2G** operation transforms to bird's-eye view and range view feature maps. The two views utilize a similar 2D FCN network with three down-sampling and three up-sampling stages, but the range view does not apply down-sampling along the height dimension. The feature channels of each stage in the 2D FCN are 64, 32, 64, 128, 128, 96, 64, 64, respectively. Therefore, the inputs of Point Fusion are 64×3 feature channels from the three branches. The output channels of these two PG fusion blocks are 64, 96, respectively.

TABLE I
THE PROPORTIONS OF POINTS IN THE 2D GRID RANGE ON SEMANTICKITTI.

BEV	RV	BEV and RV	BEV or RV
98.76%	98.60%	97.36%	99.99%

For SemanticKITTI, the bird's-eye view branch accepts a 2D feature map with the shape $(W_{bev} = 600, H_{bev} = 600)$ and the range $(x_{min} = -50, y_{min} = -50, x_{max} = 50, y_{max} = 50)$. And the range view branch sets the input shape as $(W_{rv} = 2048, H_{rv} = 64)$. The nuScenes adopts the same configuration except $H_{rv} = 32$. Based on these hyper parameters, we find that 99.99% of the whole points fall in at least one view on SemanticKITTI, as shown in Table I.

Training Details. All experiments are conducted with PyTorch FP32 on NVIDIA RTX 2080Ti GPU. The proposed CPGNet is trained from scratch for 30 epochs with the batch size of 16. The training process takes around 15 hours on 8 GPUs. The optimizer utilizes stochastic gradient descent (SGD) with the initial learning rate of 0.02, which is decayed by 0.1 every 6 epochs. Other methods are trained with the official code. Besides, we apply data augmentation during training, including random rotation around z axis, random global scale sampled from $[0.95, 1.05]$, random flipping along x and y axis, and random gaussian noise $\mathcal{N}(0, 0.02)$.

B. Results

As shown in Table II, we compare the proposed CPGNet with the state-of-the-arts on SemanticKITTI test set. The methods are grouped as point-based, 2D projection-based and sparse voxel-based methods from top to bottom. We find that CPGNet outperforms all point-based and 2D projection-based methods, and it can be comparable with the first ranking RPVNet [24] in most categories except $motorcyclist$. This category has less training samples and is confused with $bicyclist$ and $motorcycle$, which can be solved by LiDAR and image fusion. On the category of $truck$ and $traffic$-$sign$, CPGNet surpasses RPVNet by a large margin.

Besides, CPGNet runs much faster than the top ranking methods, including SPVCNN [21], Cylinder3D [22], DRINet [33] and RPVNet [24]. Note that we also test the speed of CPGNet on NVIDIA Tesla V100 GPU (marked by *) for fair comparison with RPVNet.

We report results on nuScenes validation set. As shown in Table III, CPGNet still outperforms the 2D projection-based methods and can be comparable with the first ranking RPVNet.

C. Ablation Study

In order to figure out the effectiveness of the proposed components, we make ablation study on SemanticKITTI validation set with the same experimental setup.

First, we make ablative analysis on the Point-Grid (PG) fusion block. As shown in Table IV, the baseline (first row) is the reproduced MPF [15], which adopts our 2D

TABLE II

CLASS-WISE AND MEAN IOU OF THE PROPOSED CPGNET AND THE COMPETITORS ON SEMANTICKITTI SINGLE SCAN LEADERBOARD. NOTE THAT SPEED MEASUREMENTS ARE TAKEN ON A SINGLE NVIDIA RTX 2080TI GPU, WHILE $*$ MEANS THAT IT USES NVIDIA TESLA V100 GPU.

Methods	**mIoU**	**speed (ms)**	Car	Bicycle	Motorcycle	Truck	Other-vehicle	Person	Bicyclist	Motorcyclist	Road	Parking	Sidewalk	Other-ground	Building	Fence	Vegetation	Trunk	Terrain	Pole	Traffic-sign
PointNet [1]	14.6	-	46.3	1.3	0.3	0.1	0.8	0.2	0.2	0.0	61.6	15.8	35.7	1.4	41.4	12.9	31.0	4.6	17.6	2.4	3.7
PointNet++ [2]	20.1	-	53.7	1.9	0.2	0.9	0.2	0.9	1.0	0.0	72.0	18.7	41.8	5.6	62.3	16.9	46.5	13.8	30.0	6.0	8.9
RandLA-Net [4]	53.9	521.8	94.2	26.0	25.8	40.1	38.9	49.2	48.2	7.2	90.7	60.3	73.7	20.4	86.9	56.3	81.4	61.3	66.8	49.2	47.7
KPConv [17]	58.8	-	96.0	30.2	42.5	33.4	44.3	61.5	61.6	11.8	88.8	61.3	72.7	31.6	90.5	64.2	84.8	69.2	69.1	56.4	47.4
RangeNet++ [14]	52.2	82.3	91.4	25.7	34.4	25.7	23.0	38.3	38.8	4.8	91.8	65.0	75.2	27.8	87.4	58.6	80.5	55.1	64.6	47.9	55.9
SqueezeSegv3 [25]	55.9	124.3	92.5	38.7	36.5	29.6	33.0	45.6	46.2	20.1	91.7	63.4	74.8	26.4	89.0	59.4	82.0	58.7	65.4	49.6	58.9
SalsaNext [26]	59.5	40.7	91.9	48.3	38.6	38.9	31.9	60.2	59.0	19.4	91.7	63.7	75.8	29.1	90.2	64.2	81.8	63.6	66.5	54.3	62.1
Lite-HDSeg [29]	63.8	-	92.3	40.0	55.4	37.7	39.6	59.2	71.6	54.1	93.0	68.2	78.3	29.3	91.5	65.0	78.2	65.8	65.1	59.5	67.7
MPF [15]	55.5	31	93.4	30.2	38.3	26.1	28.5	48.1	46.1	18.1	90.6	62.3	74.5	30.6	88.5	59.7	83.5	59.7	69.2	49.7	58.1
AMVNet [30]	65.3	-	96.2	59.9	54.2	48.8	45.7	71.0	65.7	11.0	90.1	71.0	75.8	32.4	92.4	69.1	85.6	71.7	69.6	62.7	67.2
SPVCNN [21]	63.8	187	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SPVNAS [21]	67.0	-	97.2	50.6	50.4	56.6	58.0	67.4	67.1	50.3	90.2	67.6	75.4	21.8	91.6	66.9	86.1	73.4	71.0	64.3	67.3
Cylinder3D [22]	67.8	178	97.1	67.6	64.0	59.0	58.6	73.9	67.9	36.0	91.4	65.1	75.5	32.3	91.0	66.5	85.4	71.8	68.5	62.6	65.6
DRINet [33]	67.5	62	96.9	57.0	56.0	43.3	54.5	69.4	75.1	58.9	90.7	65.0	75.2	26.2	91.5	67.3	85.2	72.6	68.8	63.5	66.0
AF2S3Net [23]	69.7	-	94.5	65.4	86.8	39.2	41.1	80.7	80.4	74.3	91.3	68.8	72.5	53.5	87.9	63.2	70.2	68.5	53.7	61.5	71.0
RPVNet [24]	70.3	168*	97.6	68.4	68.7	44.2	61.1	75.9	74.4	73.4	93.4	70.3	80.7	33.3	93.5	72.1	86.5	75.1	71.7	64.8	61.4
CPGNet [ours]	68.3	43/35.6*	96.7	62.9	61.1	56.7	55.3	72.1	73.9	27.9	92.9	68.0	78.1	24.6	92.7	71.1	84.6	72.9	70.2	64.5	71.9

TABLE III

RESULTS ON NUSCENES VALIDATION SET.

Methods	**mIoU**
RangeNet++ [14]	65.5
SalsaNext [26]	72.2
AMVNet [30]	76.1
Cylinder3D [22]	76.1
RPVNet [24]	77.6
CPGNet [ours]	76.9

TABLE IV

POINT-GRID (PG) FUSION BLOCK ANALYSIS ON SEMANTICKITTI VALIDATION SET.

Point Fusion	**Point Feature**	**Blocks**	**mIoU**	**speed (ms)**
×	×	1	59.5	19.2
✓	×	1	60.4	18.6
✓	✓	1	62.5	21.7
✓	✓	2	65.9	43

TABLE V

EFFECTS OF 2D FCN ARCHITECTURE AND TRANSFORMATION CONSISTENCY LOSS ON SEMANTICKITTI VALIDATION SET. DDB: DUAL DOWN-SAMPLING BLOCK. AFPN: ATTENTION FEATURE PYRAMID FUSION. TTA: TEST TIME AUGMENTATION. TRT16: TENSORRT FP16

DDB	**AFPN**	**TTA**	$\mathcal{L}_{tc}$	**TRT16**	**mIoU**	**speed (ms)**
×	×	×	×	×	63.5	38.4
✓	×	×	×	×	63.8	39.2
✓	✓	×	×	×	64.5	43
✓	✓	✓	×	×	65.7	172
✓	✓	✓	✓	×	66.1	172
✓	✓	×	✓	×	65.9	43
✓	✓	×	✓	✓	65.8	26.8

FCN architecture and transformation consistency loss. From the latter rows, we can discover that: 1) the Point Fusion outperforms by +0.9 mIoU compared with the post-process of MPF; 2) it achieves 2.1 gains of mIoU when introducing point feature; 3) the CPGNet with 2 PG fusion blocks leads to the biggest mIoU improvement, compared with the single-block version.

Subsequently, we analyze the effects of other components, including 2D FCN architecture, transformation consistency loss, and TensorRT FP16 deployment, as shown in Table V. For 2D FCN architecture, dual down-sampling block (DDB) and attention pyramid feature fusion (APFN) improve the mIoU by 0.3, 0.7, respectively, demonstrating their effectiveness. In the experiment, TTA augments point cloud for three times, namely flipping along x axis, flipping along y axis, and flipping along the both. As can be seen, TTA improves the performance (+1.2 mIoU) but needs model inference for four times. When we add transformation consistency loss during training and remove TTA during inference, it performs slightly better (+0.2 mIoU) than TTA with model inference for once. Besides, CPGNet can be easily deployed on TensorRT FP16 inference mode and runs 26.8 ms per scan with negligible performance drop (-0.1 mIoU).

V. CONCLUSIONS

In this paper, we present the accurate, fast, and easy-deployed CPGNet for LiDAR semantic segmentation, where point, bird's-eye view and range view features are fused in a cascade framework. The transformation consistency loss is proposed to save the inference time without performance drop compared with TTA. Besides, we find that 3D point features are beneficial for keeping complete point cloud information, while 2D grid features are suitable for efficient semantic feature extraction.

References

[1] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[2] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 5105–5114.

[3] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on x-transformed points," *Advances in neural information processing systems*, vol. 31, pp. 820–830, 2018.

[4] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "Randla-net: Efficient semantic segmentation of large-scale point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 108–11 117.

[5] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[6] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1520–1528.

[7] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2881–2890.

[8] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[9] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[10] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–818.

[11] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 697–12 705.

[12] Y. Wang, T. Shi, P. Yun, L. Tai, and M. Liu, "Pointseg: Real-time semantic segmentation based on 3d lidar point cloud," *arXiv preprint arXiv:1807.06288*, 2018.

[13] H. Vanholder, "Efficient inference with tensorrt," 2016.

[14] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "Rangenet++: Fast and accurate lidar semantic segmentation," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 4213–4220.

[15] Y. A. Alnaggar, M. Afifi, K. Amer, and M. ElHelw, "Multi projection fusion for real-time semantic segmentation of 3d lidar point clouds," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 1800–1809.

[16] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "Semantickitti: A dataset for semantic scene understanding of lidar sequences," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9297–9307.

[17] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6411–6420.

[18] D. Kochanov, F. K. Nejadasl, and O. Booij, "Kprnet: Improving projection-based lidar semantic segmentation," *arXiv preprint arXiv:2007.12668*, 2020.

[19] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[20] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3075–3084.

[21] H. Tang, Z. Liu, S. Zhao, Y. Lin, J. Lin, H. Wang, and S. Han, "Searching efficient 3d architectures with sparse point-voxel convolution," in *European Conference on Computer Vision*. Springer, 2020, pp. 685–702.

[22] X. Zhu, H. Zhou, T. Wang, F. Hong, Y. Ma, W. Li, H. Li, and D. Lin, "Cylindrical and asymmetrical 3d convolution networks for lidar segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9939–9948.

[23] R. Cheng, R. Razani, E. Taghavi, E. Li, and B. Liu, "2-s3net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12 547–12 556.

[24] J. Xu, R. Zhang, J. Dou, Y. Zhu, J. Sun, and S. Pu, "Rpvnet: A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 16 024–16 033.

[25] C. Xu, B. Wu, Z. Wang, W. Zhan, P. Vajda, K. Keutzer, and M. Tomizuka, "Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation," in *European Conference on Computer Vision*. Springer, 2020, pp. 1–19.

[26] T. Cortinhal, G. Tzelepis, and E. E. Aksoy, "Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds," in *International Symposium on Visual Computing*. Springer, 2020, pp. 207–222.

[27] E. E. Aksoy, S. Baci, and S. Cavdar, "Salsanet: Fast road and vehicle segmentation in lidar point clouds for autonomous driving," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 926–932.

[28] M. Berman, A. R. Triki, and M. B. Blaschko, "The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4413–4421.

[29] R. Razani, R. Cheng, E. Taghavi, and L. Bingbing, "Lite-hdseg: Lidar semantic segmentation using lite harmonic dense convolutions," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 9550–9556.

[30] V. E. Liong, T. N. T. Nguyen, S. Widjaja, D. Sharma, and Z. J. Chong, "Amvnet: Assertion-based multi-view fusion network for lidar semantic segmentation," *arXiv preprint arXiv:2012.04934*, 2020.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[32] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[33] M. Ye, S. Xu, T. Cao, and Q. Chen, "Drinet: A dual-representation iterative learning network for point cloud segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 7447–7456.