PVGNet: A Bottom-Up One-Stage 3D Object Detector with Integrated Multi-Level Features

Zhenwei Miao[1]*, Jikai Chen[1], Hongyu Pan[1], Ruiwen Zhang[1,2], Kaixuan Liu[1],
Peihan Hao[1], Jun Zhu[1], Yang Wang[1], Xin Zhan[1]
[1]Alibaba Group, [2]Tsinghua University

Abstract

Quantization-based methods are widely used in LiDAR points 3D object detection for its efficiency in extracting context information. Unlike image where the context information is distributed evenly over the object, most LiDAR points are distributed along the object boundary, which means the boundary features are more critical in LiDAR points 3D detection. However, quantization inevitably introduces ambiguity during both the training and inference stages. To alleviate this problem, we propose a one-stage and voting-based 3D detector, named Point-Voxel-Grid Network (PVGNet). In particular, PVGNet extracts point, voxel and grid-level features in a unified backbone architecture and produces point-wise fusion features. It segments LiDAR points into foreground and background, predicts a 3D bounding box for each foreground point, and performs group voting to get the final detection results. Moreover, we observe that instance-level point imbalance due to occlusion and observation distance also degrades the detection performance. A novel instance-aware focal loss is proposed to alleviate this problem and further improve the detection ability. We conduct experiments on the KITTI and Waymo datasets. Our proposed PVGNet outperforms previous state-of-the-art methods and ranks at the top of KITTI 3D/BEV detection leaderboards.

1. Introduction

In autonomous driving, 3D object detection is a critical task that has received significant attention from both industry and academia [8, 10, 11, 20, 24]. LiDAR sensors are the most widely used 3D sensors. The generated point clouds provide accurate distance measurements and geometric information for environment understanding. Different from 2D images, 3D point clouds are irregular, sparse, and unevenly distributed. As shown in Fig. 1, the LiDAR points are mainly located along the object boundary. Therefore,

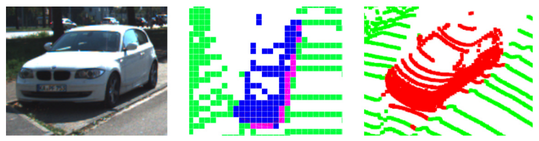

Figure 1. An example of an object with representation of (a) image, (b) grid-quantization of LiDAR points, (c) raw LiDAR points.

it is important to extract the boundary feature. Moreover, compared with images, the boundary of an object in LiDAR points is crystally clear with its background without ambiguity. Under such LiDAR points merits, effectively extracting the object boundary features is critical for 3D detection.

According to the type of input point cloud representation, 3D object detectors can be divided into two categories: point-based and quantization-based methods. Point-based detectors [18, 19] extract local features by searching for local neighborhoods from a set of specifically selected key points. This local search process is time-consuming, preventing point-based detectors from being widely adopted in real-time autonomous driving systems. Quantization-based methods include voxel-based and grid-based ones. Voxel-based detectors [32] split point clouds into evenly spaced voxels and use sparse 3D convolution [7] for computational acceleration. Grid-based detectors [10] project point clouds onto 2D grids and utilize 2D convolution for feature extraction. The quantization inevitably loses detailed geometric information and leads to an ambiguous object boundary. This ambiguity will introduce training uncertainty and degrade the discrimination of object boundary representation.

To alleviate the above quantization problem and take advantage of the efficient feature representation, we propose a unified architecture Point-Voxel-Grid Network (PVGNet). As illustrated in Fig. 2, PVGNet extracts these multi-level features simultaneously and applies a point-wise supervision signal on the point-level fusion feature. As a result, detailed geometric information and rich semantic information are integrated into a single feature vector. The crystallized object boundary in LiDAR points is preserved in the PVGNet from the supervision signal. A bottom-up detection head is used to predict a 3D bounding box for each

*corresponding author: zhenwei.mzw@alibaba-inc.com

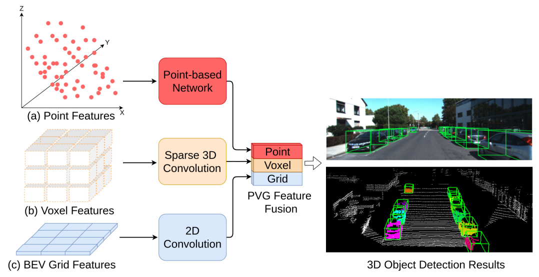

Figure 2. Our proposed PVGNet is a one-stage 3D object detector using LiDAR point clouds, which integrates point, voxel and grid-level features in a unified backbone network.

foreground point, followed by a group voting algorithm to generate the final detection results.

As a one-stage bottom-up detector, the training of PVGNet faces severe sample imbalance problems. In addition to the imbalance of foreground/background points, we observe that the imbalance of instance-level points also exists. Specifically, an unoccluded and closer instance usually contains more points than an occluded and farther instance. Therefore, we propose a novel instance-aware focal loss (IAFL) to alleviate the imbalance problem and further boost the detection performance.

Our contributions are summarized as follows:

- We propose PVGNet framework by extracting point, voxel, and grid-level features in a unified backbone and detecting 3D objects in a bottom-up one-stage manner. This framework enables learning integrated point-wise features for accurate object bounding box prediction.
- We propose a group voting-based method to generate the final detection results, which is more efficient and accurate than NMS based post-processing.
- We propose a new instance-aware focal loss for dealing with instance-level foreground point number imbalance. Extensive experiments are carried out to explore its effectiveness.

Unlike the grid feature map-based method such as PVR-CNN [20] and HVNet [29], the proposed PVGNet is a bottom-up 3D detector with point-level supervision, which segments point clouds and generates bounding boxes simultaneously in one-stage manner. Different from the Hough voting in VoteNet [16] that is used to generate votes for a refinement deep network module, the group voting in our algorithm is proposed to directly generate the final detection results from the point-level 3D bounding box predictions.

2. Related Work

2.1. Two-stage 3D Object Detection

A typical two-stage 3D object detector consists of Stage-I for proposal generation and Stage-II for fine-grained box refinement. PointRCNN [21] designs a bottom-up strategy to segment the input point clouds and generates proposals from the segmented foreground points. Part-A^{2}net [22] develops an encoder-decoder backbone network and uses 3D intra-object part locations to improve detection performance in Stage-II. STD [28] proposes the sparse to dense strategy for multi-stage proposal refinement. PV-RCNN [20] uses anchor-based 3D voxel CNN backbone for proposal generation and applies keypoint-to-grid ROI feature abstraction for proposal refinement.

2.2. One-stage 3D Object Detection

A set of 3D boxes is classified and regressed to improve computational efficiency in a one-stage framework. VoxelNet [32] designs voxel feature encoding (VFE) layers to obtain the feature representation of each voxel and uses convolutional middle layers to generate 2D feature maps. SECOND [25] proposes an improved sparse 3D convolution to speed up inference and optimize GPU usage. PointPillars [11] extracts features from stacked pillars and represents the feature maps as 2D pseudo-images for convolution. HVNet [29] designs a hybrid voxel network to enhance the voxel-wise feature. SA-SSD [9] employs an auxiliary network to learn point-level structure information and exhibits better object localization performance.

2.3. Feature Extraction on Point Clouds

We group most previous point cloud representations into point, voxel and grid-based categories. PointNet [18] is a pioneer work to extract features directly on points. PointNet++ [19] is an evolved version of PointNet and is widely used as the backbone of point-based detectors [21, 28, 16, 17, 30]. VoxelNet [32] introduces VFE layers, which are also adopted in [25, 3], to extract voxel-wise features. Voxel-based methods use 3D convolution to achieve high-quality detection results, but voxelization still leads to loss of geometric details. Grid-based methods [2, 26, 11] project the point clouds onto regular grids and apply 2D convolution. They are usually efficient but have limited detection accuracy. In comparison, our PVGNet extracts features at point, voxel and grid-level simultaneously, and integrates them into point-wise representations.

3. Proposed PVGNet

In this section, we present our proposed one-stage 3D object detection framework PVGNet. Fig. 3 illustrates the overall architecture, consisting of the Point-Voxel-Grid (PVG) feature fusion, the point-wise prediction and group voting parts. These three parts are described in Sec. 3.2, Sec. 3.3 and Sec. 3.4 respectively. Then we detail how to use the instance-aware focal loss for imbalance problem alleviation in Sec. 3.5 and define the total loss function in Sec. 3.6.

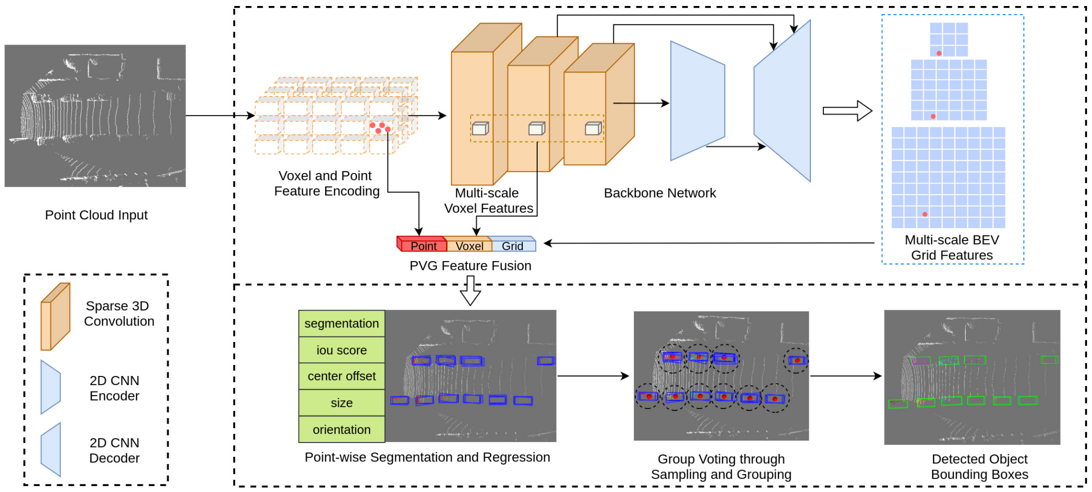
Point Cloud Input
Voxel and Point Feature Encoding
Multi-scale Voxel Features
Backbone Network
Multi-scale BEV Grid Features
Point
Voxel
Grid
PVG Feature Fusion
Sparse 3D Convolution
2D CNN Encoder
2D CNN Decoder
segmentation
iou score
center offset
size
orientation
Point-wise Segmentation and Regression
Group Voting through Sampling and Grouping
Detected Object Bounding Boxes

Figure 3. Illustration of the PVGNet architecture for 3D object detection. First, we voxelize the point cloud and apply the voxel and point feature encoding to each non-empty voxel. Second, a backbone network composed of sparse 3D convolution and 2D convolution is adopted to produce multi-scale voxel-wise features and grid-wise BEV features. By concatenating these different-level features, point-wise PVG fusion features are obtained. Then we use PVG features to segment the point clouds and regress a 3D bounding box for each foreground point. Finally, a group voting-based method is used to generate the final detection results by merging redundant boxes.

Grid Size	Ambiguous Grid Ratio
1m*1m	80.7%
0.5m*0.5m	65.9%
0.2m*0.2m	45.4%
0.05m*0.05m	20.9%
0.02m*0.02m	8.9%
$\rightarrow 0$	$\rightarrow 0$

Table 1. Statistical results of the ambiguous grid ratio of the *Car* category from the KITTI train set.

3.1. Motivation

Fig. 1 shows a vanilla example of object appearance representation. For an object from an image, detailed context information appeared on both the inner and boundary in Fig. 1(a). In contrast, LiDAR points of an instance are mainly located along the object boundary, while the inner region contains only quite a few points as shown in Fig. 1(b). Therefore, effectively extracting the object boundary features is the critical issue for 3D detection. However, the quantization-based LiDAR point projection usually introduces the uncertainty of grid/voxel if multiple points from different classes are projected into the same grid/voxel (named as ambiguous grid/voxel). The statistical results of the ambiguous grid ratio of the *Car* category from the KITTI train set are given in Table 1. It is seen that the ambiguous grid ratio is above 45% for the grid size large than 0.2m, which deteriorates both the feature extraction and supervision signal generation. Fig. 1(c) is the point-level representation, which has a clear boundary between the object and the background. The point-level representation is equivalent to a grid size with zero, which means zero ambiguous grid ratio. This analysis motivates us to design the architecture from the point-level feature representation.

3.2. PVG Feature Fusion

3.2.1 Voxel and Point Feature Encoding

Sparse 3D voxel CNN is widely used in many state-of-the-art 3D detectors due to its high efficiency and accuracy. Point cloud $\mathcal{P} = \{p_i\}_{i=1}^{N}$ is firstly divided into evenly spaced voxels with the size of (v_d, v_h, v_w) along the Z, Y and X axes. The points residing in the same voxel are grouped based on their spatial coordinates, then the sampling stage sub-samples a fixed number of T points from each voxel. The sub-sampled point set of a specified voxel $\tilde{v}$ is denoted as $\mathcal{P}_s$, which will be fed into the VFE layers to extract voxel-wise features $f^{(v_0)} = \text{VFE}(\mathcal{P}_s)$. More details about the VFE layer can be found in [32].

Previous works such as [20] and [9] revealed the importance of leveraging fine-grained spatial information of points to achieve accurate bounding box localization. In our work, we introduce a voxel and point feature encoding (VPFE) layer, as shown in Fig. 4, to extract the point-wise features of all observed points as well as the voxel-wise features. The VPFE layer is an extension of the VFE layer. First, in the VFE stage, the voxel-wise feature $f^{(v_0)}$ is encoded using the sampled point set $\mathcal{P}_s$. Then all of the raw points $\{p_i\}$ in voxel $\tilde{v}$, including the points retained and dropped during the VFE process, are fed into a MLP network to generate features $\{f_i^{(raw)}\}$. Finally, we augment each $f_i^{(raw)}$ with $f^{(v_0)}$ via concatenation and produce the output point-wise encoded feature set $\{f_i^{(p)}\}$ using MLP:

$$f_i^{(p)} = \mathcal{M}\left(\left[f_i^{(raw)}, f^{(v_0)}\right]\right), \quad (1)$$

where $[\cdot]$ denotes concatenation and $\mathcal{M}(\cdot)$ denotes a MLP network.

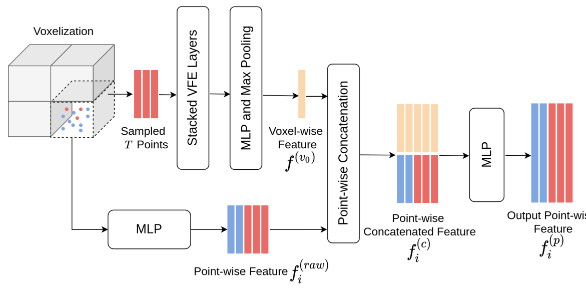
Voxelization
Sampled T Points
Stacked VFE Layers
MLP and Max Pooling
Voxel-wise Feature $f^{(v_0)}$
Point-wise Concatenation
MLP
Point-wise Concatenated Feature $f_i^{(c)}$
MLP
Output Point-wise Feature $f_i^{(p)}$
Point-wise Feature $f_i^{(raw)}$

Figure 4. Voxel and point feature encoding layer. For each non-empty voxel, the stacked VFE layers are used to generate the voxel-wise feature using the sampled T points. Then the point-wise features for all observed points in the voxel are extracted by additional concatenation and MLP layers.

The voxel-wise feature set $\{f^{(v_0)}\}$ will be fed into the backbone network, and the point-wise feature set $\{f_i^{(p)}\}$ will be used for further PVG feature fusion described later.

3.2.2 Backbone Network

We show the details of our backbone network in Fig. 5. The input is the encoded voxel-wise feature map $\{f^{(v_0)}\}$. The first part consists of several sparse 3D convolutional layers. We insert sparse 3D convolutions with a stride of 2 to gradually produce $2\times$, $4\times$, $8\times$ downscaled sparse volumes. The corresponding multi-scale voxel-wise feature vector sets are respectively denoted as $\{f^{(v_1)}\}$, $\{f^{(v_2)}\}$ and $\{f^{(v_3)}\}$, as shown in Fig. 5(a). Then, by concatenating the voxel-wise feature vectors along the Z axis into one channel, the downscaled sparse volumes are reshaped into 2D BEV feature maps. 2D convolutions are applied for further feature abstraction and enlarging the receptive field.

Feature pyramid network (FPN) [12] has been proven to be very effective in 2D image detection and segmentation tasks. Following the design of FPN, we apply the transposed convolutional layers to up-sample the BEV feature maps. We concatenate the higher-level features containing stronger semantic information with the lower-level features containing richer location information. Before concatenation, we reshape the lower-level sparse 3D volumes into 2D BEV features with dimensionality reduction. The resulting multi-scale BEV feature map set is denoted as $\{f^{(bev_k)}\}_{k=0,1,2,3}$, as illustrated in Fig. 5(c).

3.2.3 Point-wise PVG Feature Fusion

In 3D detection tasks, since context and location information are both important, it is best to integrate these multi-level features together for subsequent classification and regression. PVG feature fusion is designed for this purpose. The feature set $\{f_i^{(p)}\}$ encodes point interactions within a voxel and describes the fine-grained structure of the point cloud. However, the receptive field of $\{f_i^{(p)}\}$ is extremely limited. The voxel-wise feature set $\{f^{(v_k)}\}$ encodes voxel interactions in 3D space. In this way, the feature discrimination ability along the Z axis can be better preserved. The grid-wise feature set $\{f^{(bev_k)}\}$ encodes grid interactions under BEV. Compared with $\{f_i^{(p)}\}$ and $\{f^{(v_k)}\}$, $\{f^{(bev_k)}\}$ has a larger receptive field and captures richer contextual information. For a specific point p_i, we denote $f_i^{(v_k)}$ and $f_i^{(bev_k)}$ as the k-th level feature vector of the voxel and grid where p_i is located, respectively. Then the fused PVG feature is generated by integrating $f_i^{(p)}$, $f_i^{(v_k)}$ and $f_i^{(bev_k)}$:

$$f_i^{(PVG)} = \left[f_i^{(p)}, \mathcal{M}_v\left(\left[f_i^{(v_k)}\right]\right), \mathcal{M}_{bev}\left(\left[f_i^{(bev_k)}\right]\right)\right], \tag{2}$$

where $[f_i^{(v_k)}]$ and $[f_i^{(bev_k)}]$ are the concatenated voxel-wise and grid-wise feature vectors. The PVG fusion feature set $\{f_i^{(PVG)}\}_{i=1}^N$ are generated for all points $\{p_i\}_{i=1}^N$.

3.3. Point-wise Segmentation and Regression

Given the per-point features $\{f_i^{(PVG)}\}_{i=1}^N$, we predict foreground confidence s_i, bounding box $\mathbf{b}_i$ and 3D Intersection-over-Union (IoU) score s_{iou_i} by three MLP headers. For point segmentation, the ground-truth semantic classes of points can be easily generated by the 3D ground-truth boxes. A segmentation branch with MLP layers is applied to calculate the foreground confidence of each point.

For each segmented foreground point, a regression branch is used to predict the corresponding 3D bounding box encoded as

$$\mathbf{b}_i = (c_x, c_y, c_z, l, w, h, \theta). \tag{3}$$

Instead of regressing the center value directly, we predict the offset, $(\Delta x, \Delta y, \Delta z)$, from the center of the instance where p_i resides to the position of p_i. The estimation of size (l, w, h) and orientation θ are similar to those in [20]. More details can be found in Sec. 3.6.

Besides, for each predicted box, the 3D IoU score is estimated for localization quality evaluation. Previous works such as [28] and [20] have shown that the quality-aware IoU score achieves better performance than the traditional classification score. In the training phase, all the regression components described above are predicted point-wisely from the network, but only the foreground points are contributed to the loss computation.

3.4. Group Voting

Our network predicts a 3D bounding box for every foreground point. For inference, it is necessary to merge these boxes to generate the final detection results. NMS is widely used in many detectors for this purpose. However, in NMS, only the box with the highest score is selected and the other overlapping boxes are suppressed. To better perform box merging, we propose a method based on group voting,

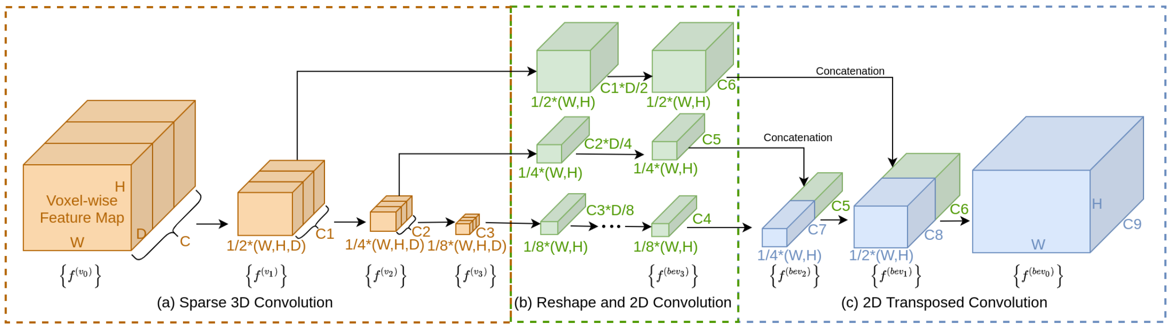

Figure 5. Backbone network composed of sparse 3D convolution, 2D convolution and lateral connections.

which improves efficiency and effectiveness compared with NMS.

In the group voting algorithm, all segmented foreground points are translated by predicted displacement to the corresponding centroids to form the votes $\{v_i\}_{i=1}^{N_{fg}}$. Then the vote clusters are generated through sampling and grouping, and each cluster corresponds to an object. Specifically, we use farthest point sampling (FPS) to sample a subset of K key votes $\{o_k\}_{k=1}^{K}$ from $\{v_i\}_{i=1}^{N_{fg}}$. For each sampled key vote o_k, a vote cluster G_k is formed by finding all votes that are within a radius r to o_k:

$$G_k = \{v_i | \, \|v_i - o_k\| \leq r\}, \text{for } k = 1, \ldots, K. \tag{4}$$

After clustering, we average the attributes (center, size, orientation, and score) of the predicted boxes belonging to the same cluster.

3.5. Instance-aware Focal Loss

For the point-wise segmentation task, the extreme foreground/background class imbalance overwhelms the training process and degrades model performance. Moreover, in 3D detection, we are more concerned about instance-level segmentation/detection performance than point-level. In a frame, different instances have different numbers of points. An unoccluded and closer instance usually contains more points than an occluded and farther instance. As a result, "easy" instances with more points comprise the majority of the point-wise foreground segmentation loss and dominate the gradient, limiting overall performance. To this end, we propose an instance-aware focal loss (IAFL) to alleviate the instance-level imbalance problem.

For each foreground point p_i, we can find the instance $\mathcal{I}(p_i)$ to which p_i belongs. We propose to add a modulating factor $\text{IA}(p_i)$ to the focal loss:

$$\text{IA}(p_i) = \beta \left(1 - \frac{1}{N_{\mathcal{I}(p_i)}} \sum_k s_k \cdot \mathbb{1}[p_k \text{ in } \mathcal{I}(p_i)] \right)^{\tau}, \tag{5}$$

where $\mathbb{1}[p_k \text{ in } \mathcal{I}(p_i)]$ indicates whether a point p_k belongs to instance $\mathcal{I}(p_i)$, s_k is the predicted foreground probability of point p_k, and $N_{\mathcal{I}(p_i)}$ is the count of total number of points in $\mathcal{I}(p_i)$. The term $\frac{1}{N_{\mathcal{I}(p_i)}} \sum_k s_k \cdot \mathbb{1}[p_k \text{ in } \mathcal{I}(p_i)]$, denoted as $s_{\mathcal{I}(p_i)}$ for brevity, is used to evaluate the classification accuracy of instance $\mathcal{I}(p_i)$. β and τ are tunable hyperparameters.

By adding the factor $\text{IA}(p_i)$ to the original focal loss $\text{FL}(p_i, s_t)$ [13], we get the IAFL

$$\text{IAFL}(p_i, s_t) = \begin{cases} (1 + \text{IA}(p_i))\text{FL}(p_i, s_t) & \text{if } y = 1, \\ \text{FL}(p_i, s_t) & \text{otherwise.} \end{cases} \tag{6}$$

When most points in $\mathcal{I}(p_i)$ are well classified, the factor $\text{IA}(p_i)$ is near 0 and IAFL is equivalent to FL. As $s_{\mathcal{I}(p_i)} \to 0$, the factor $\text{IA}(p_i)$ rises and the loss of all points in $\mathcal{I}(p_i)$ is up-weighted. In this way, IAFL reduces the loss contribution from "easy" instances that may contain many points and makes the training process more balanced.

3.6. Total Loss Function

The overall multi-task loss consists of point-wise segmentation loss, bounding box regression loss and 3D IoU score prediction loss.

Point-wise segmentation loss L_{seg} is computed by the IAFL described in Sec. 3.5, as

$$L_{seg} = \frac{1}{N} \sum_{i=1}^{N} \text{IAFL}(p_i), \tag{7}$$

where N is the total number of points.

Smooth-L1 loss is uses for point-wise bounding box regression task. We define the vector $\mathbf{b}_i^* \in R^7$ containing the regression training targets corresponding to center location $(c_{x_t}, c_{y_t}, c_{z_t})$, box size (l_t, w_t, h_t) and the orientation θ_t, as

$$\begin{aligned} & c_{x_t} = x - c_{x_g}, c_{y_t} = y - c_{y_g}, c_{z_t} = z - c_{z_g}, \\ & l_t = \log(\frac{l_g}{l_a}), w_t = \log(\frac{w_g}{w_a}), h_t = \log(\frac{h_g}{h_a}), \\ & \theta_t = \sin(\theta_g), \end{aligned} \tag{8}$$

where $(c_{x_g}, c_{y_g}, c_{z_g}, l_g, w_g, h_g, \theta_g)$ is the 3D ground-truth bounding box, (x, y, z) is the point position coordinate, and (l_a, w_a, h_a) is the predefined constant anchor size. We then average the regression loss of all foreground points as

$$L_{reg} = \frac{1}{N_{fg}} \sum_{i=1}^{N_{fg}} \mathcal{L}_{smooth_{L_1}}(\mathbf{b}_i, \mathbf{b}_i^*), \tag{9}$$

where N_{fg} is the number of foreground points.

For the confidence prediction branch, similar to [20], we use 3D IoU between detected bounding boxes and corresponding groud-truth boxes as the training target

$$s_{iou}^* = \min(1, \max(0, \frac{1}{b_u - b_l}(\mathrm{IoU} - b_l))), \tag{10}$$

where the lower bound b_l and the upper bound b_u are used to map IoU to the target probability score. The IoU prediction loss is defined as a smooth-L1 loss of

$$L_{iou} = \frac{1}{N_{fg}} \sum_{i=1}^{N_{fg}} \mathcal{L}_{smooth_{L_1}}(s_{iou_i}, s_{iou_i}^*), \tag{11}$$

where s_{iou_i} denotes the estimated score of the model.

Our total loss is the weighted sum of the above three losses, expressed as

$$L = L_{seg} + \lambda_{iou} L_{iou} + \lambda_{reg} L_{reg}, \tag{12}$$

where λ_{iou} and λ_{reg} are constant weights to balance different tasks.

4. Experiments

We conduct experiments on the KITTI [6] dataset and compare our method with previous state-of-the-art in terms of 3D object and BEV detection. Mean average precision (mAP) with an IoU threshold of 0.7 is used as the evaluation metric. KITTI dataset contains 7,481 training samples and 7,518 test samples. The training set is divided into *train* split with 3,712 samples and *val* split with 3,769 samples as suggested in [32]. The labeled instances are further divided into easy, moderate, and hard categories based on their height in the image, occlusion ratios, etc.

4.1. Implementation Details

4.1.1 Data Augmentation

We conduct the cut-and-paste augmentation strategy [25, 4] to randomly sample additional ground-truth boxes from other scenes into the current scene to simulate objects in various environments. We also apply the widely used global augmentation strategies [20], including random flipping, global rotation, global scaling, and global translation. The angle of global rotation is randomly sampled from $\mathcal{U}(-\frac{\pi}{4}, \frac{\pi}{4})$, and the ratio of global scaling is sampled from $\mathcal{U}(0.95, 1.05)$. Global translation is applied by randomly adding a displacement sampled from $\mathcal{N}(-0.2, 0.2)$. Besides, we perform instance-level augmentation strategies on individual annotations. The noise of instance rotation is uniformly drawn from $\mathcal{U}(-\frac{\pi}{20}, \frac{\pi}{20})$, and the ratio of scaling is drawn from $\mathcal{U}(0.95, 1.05)$.

4.1.2 Network Architecture and Training Details

On the KITTI dataset, the detection range is within $[0, 71.68]m$ for the X axis, $[-40.32, 40.32]m$ for the Y axis and $[-3, 1]m$ for the Z axis. The voxel size is set to $(0.07m, 0.07m, 0.2m)$. The VPFE module extracts 64-dim features for each non-empty voxel and 80-dim features for each observed point. The backbone network utilizes nine $3\times3\times3$ sparse 3D convolutional layers to gradually convert the voxel features into feature volumes with $2\times$, $4\times$, $8\times$ downscaled sizes. These sparse feature volumes are converted into dense tensors, and a series of 2D transposed convolutional layers are used to generate full-resolution BEV feature maps by upsampling spatially coarser features. At last, the concatenated multi-scale voxel-wise and grid-wise features are further processed by MLP with output sizes of 96 and 96. The final fused PVG feature is a 272-dim vector. The point segmentation and box regression branches are both realized through MLP layers.

We train the entire network with batch size 16 and learning rate 0.01 on 8 RTX 2080 Ti GPUs. ADAM optimizer is adopted to train our PVGNet from scratch in an end-to-end manner with cosine annealing learning rate. The hyperparameters β and τ defined in IAFL are empirically set to 0.4 and 2.0, the lower bound b_l and upper bound b_u defined in Eq. (10) are set to 0.55 and 0.85, and λ_{iou} and λ_{reg} in total loss Eq. (12) are set to 2 and 4, respectively. A fixed number of 64 key votes are sampled for group voting of the KITTI dataset. A key vote represents an instance after clustering. The maximum number of instances in a frame is 22 in KITTI *val* split and 64 is to guarantee to be sufficient in both *val* split and *test* set. Since the number of targets in a frame may be less than 64, NMS with threshold 0.01 is applied on key votes to further remove the redundant bounding boxes.

4.2. 3D Object Detection on KITTI

4.2.1 Evaluation on KITTI *Test* Set

We evaluate our PVGNet detector on KITTI *test* set with the official online test server. The mAP results on the most important car category calculated with 40 recall positions are listed in Table 2 test set column. In both 3D object and BEV detection tasks, our method achieves the best performance among all competitors. In the context of one-stage detector, PVGNet outperforms SA-SSD [9] with remarkable margins. Specifically, it increases the mAP of 3D detection by 1.19%, 2.02%, 2.93% on easy, moderate and hard difficulty levels, respectively. Compared with the state-of-the-art two-stage detector PVRCNN [20], our one-stage model achieves

	Test Set						Validation Set					
	3D Detection			BEV Detection			3D Detection			BEV Detection		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
Two-stage:												
AVOD-FPN [10]	83.07	71.76	65.73	90.99	84.82	79.62	84.41	74.44	68.65	–	–	–
PointRCNN [21]	86.96	75.64	70.70	92.13	87.39	82.72	88.88	78.63	77.38	–	–	–
F-ConvNet [23]	87.36	76.39	66.69	91.51	85.84	76.11	89.02	78.80	77.09	90.23	88.79	86.84
SERCNN [30]	87.74	78.96	74.30	94.11	88.10	83.43	89.50	79.21	78.16	90.23	87.53	86.45
STD [28]	87.95	79.71	75.09	94.74	89.19	86.42	89.70	79.80	**79.30**	90.50	88.50	88.10
PVRCNN [20]	**90.25**	81.43	76.82	94.98	90.65	86.14	–	83.90	–	–	–	–
One-stage:												
VoxelNet [32]	77.47	65.11	57.73	89.35	79.26	77.39	81.97	65.46	62.85	89.60	84.81	78.57
PointPillars [11]	82.58	74.31	68.99	90.07	86.56	82.81	87.29	76.99	70.84	90.07	87.06	83.81
TANet [14]	84.39	75.94	68.82	91.58	86.54	81.19	88.21	77.85	75.62	90.17	87.55	87.14
SECOND [25]	84.65	75.96	68.71	91.81	86.37	81.04	87.43	76.48	69.10	89.96	87.07	79.66
3DSSD [27]	88.36	79.57	74.55	92.66	89.02	85.86	89.71	79.45	78.67	–	–	–
SA-SSD [9]	88.75	79.79	74.16	**95.03**	91.03	85.96	**90.15**	79.99	78.78	–	–	–
PVGNet	89.94	**81.81**	**77.09**	94.36	**91.26**	**86.63**	89.40	**85.05**	79.00	**90.55**	**88.88**	**88.30**

Table 2. Performance comparison on the KITTI *test* and *val* set. The results are evaluated by the mean average precision with 40 recall positions for the *test* set and 11 recall positions for the *val* set.

	3D Detection			BEV Detection		
	Easy	Mod.	Hard	Easy	Mod.	Hard
PVRCNN [20]	92.57	84.83	82.69	95.76	91.11	88.93
Proposed PVGNet	**92.75**	**85.51**	**83.04**	**96.57**	**91.88**	**89.56**

Table 3. Performance comparison on the KITTI *val* set. The results are evaluated by the mAP with 40 recall positions.

(0.38%, 0.27%) improvement in 3D detection and (0.61%, 0.49%) improvement in BEV detection on the moderate and hard difficulty levels, respectively.

As of Aug. 10th, 2020, the proposed PVGNet ranks 1^{st} on both the car 3D and BEV detection leaderboards among all published papers, including one-stage and two-stage approaches. This leaderboard demonstrates the effectiveness of our method and proves that a well-designed one-stage detector has excellent potential in 3D object detection tasks.

4.2.2 Evaluation on KITTI *Val* Set

We compare our PVGNet with other state-of-the-art approaches on KITTI *val* set. Results with R40 are reported in Table 3. The proposed PVGNet leads PVRCNN by (0.18%, 0.68%, 0.35%) in 3D detection and (0.81%, 0.77%, 0.63%) in BEV detection. Table 2 validation set column shows our results evaluated by the mAP with R11. PVGNet achieves a significant improvement of 5.06% compared with SA-SSD and 1.15% compared with PVRCNN in 3D detection on the most important moderate subset.

4.3. Ablation Studies

Extensive ablation experiments are carried out to analyze the individual components of our proposed method. All the results are evaluated on the KITTI *val* set by the 3D mAP with 40 recall positions.

Grid	Point	Voxel	Easy	Mod.	Hard
✓			87.90	79.99	78.73
		✓	89.83	77.22	73.11
✓	✓		92.41	83.23	80.62
✓	✓	✓	92.75	85.51	83.04

Table 4. Effects of PVG feature fusion.

4.3.1 Effects of PVG Feature Fusion

We validate the effectiveness of PVG feature fusion strategy by comparing different feature combinations. As shown in Table 7, fusing multi-level features contributes significantly to the performance on all difficulty levels. Due to the lack of information along Z axis, using only grid-level features results in the lowest mAP. Voxel-wise and point-wise features are more refined to be more beneficial for object localization. Since context and location information are both critical in 3D detection, integrating these multi-level features increases the mAP performance.

	Easy	Mod.	Hard
PVGNet with Rotated NMS	92.41	83.74	81.72
PVGNet with Group Voting	92.75	85.51	83.04

Table 5. Effects of group voting.

4.3.2 Effects of Group Voting

We compare the detection results of using group voting and rotated NMS for box merging. Table 5 shows that group

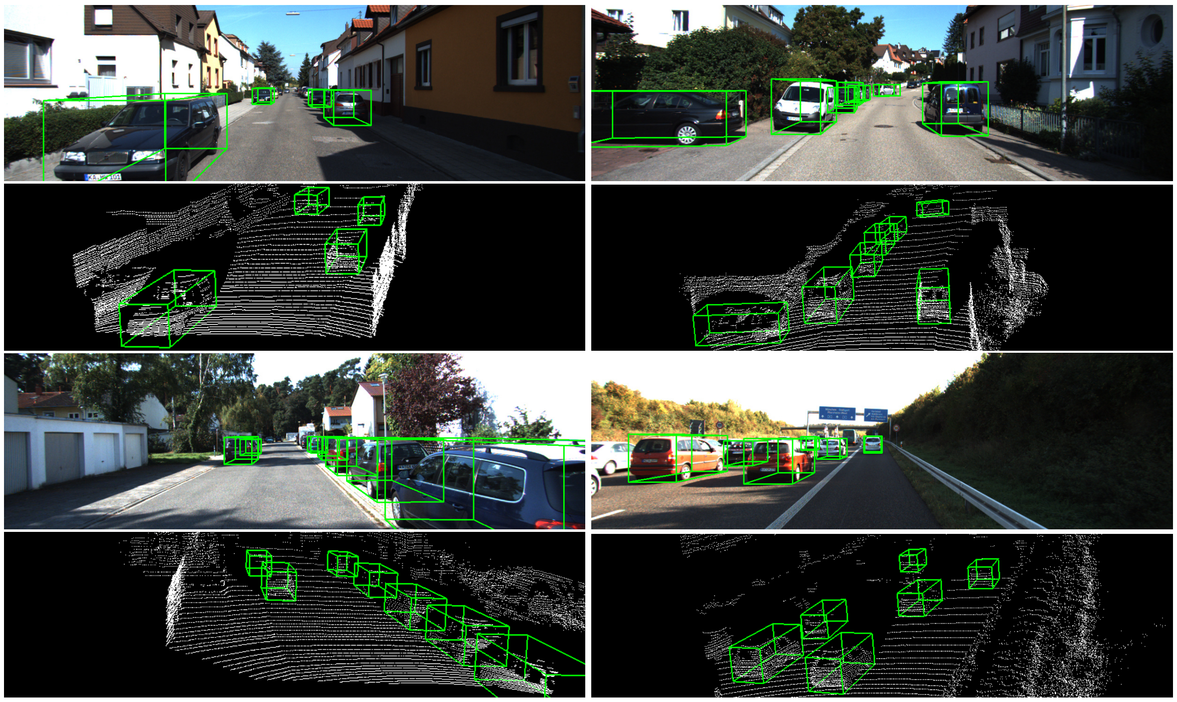
Figure 6. The results on KITTI *val* set. The predicted bounding boxes are shown in green. The predictions are projected onto the RGB images (upper row) for better visualization.

voting achieves (0.34%, 1.77%, 1.32%) mAP improvement than rotated NMS on different difficulty levels.

Grid	Point	Voxel	without IAFL	with IAFL
✓			79.02	79.99
✓	✓		82.19	83.23
✓	✓	✓	85.12	85.51

Table 6. Effects of instance-aware focal loss. The results are evaluated on the moderate difficulty level.

4.3.3 Effects of Instance-aware Focal Loss

The 3D detection results with and without IAFL are summarized in Table 6. For different feature fusion strategies, using the proposed IAFL consistently achieves superior performance. It demonstrates that reducing the imbalance of instance-level points is beneficial for the LiDAR-based bottom-up 3D detectors.

4.4. Qualitative Results

Visualization of detection results by PVGNet is shown in Fig. 6. It demonstrates that the proposed one-stage 3D object detector predicates high-quality 3D bounding boxes.

4.5. Evaluation on Waymo *Val* Set

For the Waymo Open dataset [15, 31], the detection range is within $[-75.2, 75.2]m$ for the X and Y axes and $[-2, 4]m$ for the Z axis. We report the whole *val* set result using the training dataset, as shown in Table 7. We outperforms the state-of-the-art with a large margin on the Vehicle, Pedestrian and Cyclist categories.

	Vehicle	Ped	Cyc
MVF [31]	62.9	65.3	–
AFDet [5]	63.7	–	–
RCD [1]	69.6	–	–
PVRCNN [20]	70.3	–	–
PVGNet	74.0	69.5	72.3

Table 7. Results on Waymo by official LEVEL_1 3D mAP.

5. Conclusion

This work introduces a novel one-stage bottom-up 3D object detection framework named PVGNet, which extracts point, voxel and grid-level features in a unified backbone architecture. A box merging method based on group voting is proposed to generate the detection results. Furthermore, an instance-aware focal loss is designed to alleviate the instance-level imbalance in point-wise segmentation. Experiments conducted on the well-known KITTI and Waymo datasets demonstrate that our one-stage PVGNet outperforms previous one-stage and two-stage state-of-the-art algorithms with remarkable margins.

6. Acknowledgement

This work was supported by the funding of "Leading Innovation Team of the Zhejiang Province" (2018R01017), and by Alibaba Group through Alibaba Innovative Research Program and Alibaba Research Intern Program.

References

[1] Alex Bewley, Pei Sun, Thomas Mensink, Drago Anguelov, and Cristian Sminchisescu. Range conditioned dilated convolutions for scale invariant 3d object detection. In *Conference on Robot Learning*, 2020.

[2] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

[3] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast point r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9775–9784, 2019.

[4] Shuyang Cheng, Zhaoqi Leng, Ekin Dogus Cubuk, Barret Zoph, Chunyan Bai, Jiquan Ngiam, Yang Song, Benjamin Caine, Vijay Vasudevan, Congcong Li, et al. Improving 3d object detection through progressive population based augmentation. *arXiv preprint arXiv:2004.00831*, 2020.

[5] Runzhou Ge, Zhuangzhuang Ding, Yihan Hu, Yu Wang, Sijia Chen, Li Huang, and Yuan Li. Afdet: Anchor free one stage 3d object detection. 2020.

[6] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[7] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9224–9232, 2018.

[8] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[9] Chenhang He, Hui Zeng, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. Structure aware single-stage 3d object detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11873–11882, 2020.

[10] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pages 1–8, 2018.

[11] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.

[12] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[13] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.

[14] Zhe Liu, Xin Zhao, Tengteng Huang, Ruolan Hu, Yu Zhou, and Xiang Bai. Tanet: Robust 3d object detection from point clouds with triple attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 11677–11684, 2020.

[15] Jiquan Ngiam, Benjamin Caine, Wei Han, Brandon Yang, Yuning Chai, Pei Sun, Yin Zhou, Xi Yi, Ouais Alsharif, Patrick Nguyen, et al. Starnet: Targeted computation for object detection in point clouds. *arXiv preprint arXiv:1908.11069*, 2019.

[16] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9277–9286, 2019.

[17] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018.

[18] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas 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*, 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. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.

[20] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020.

[21] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019.

[22] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[23] Zhixin Wang and Kui Jia. Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pages 1742–1749, 2019.

[24] Liang Xie, Chao Xiang, Zhengxu Yu, Guodong Xu, Zheng Yang, Deng Cai, and Xiaofei He. Pi-rcnn: An efficient multi-sensor 3d object detector with point-based attentive cont-conv fusion module. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 12460–12467, 2020.

[25] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

[26] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018.

[27] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11040–11048, 2020.

[28] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std: Sparse-to-dense 3d object detector for point cloud. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1951–1960, 2019.

[29] Maosheng Ye, Shuangjie Xu, and Tongyi Cao. Hvnet: Hybrid voxel network for lidar based 3d object detection. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1631–1640, 2020.

[30] Dingfu Zhou, Jin Fang, Xibin Song, Liu Liu, Junbo Yin, Yuchao Dai, Hongdong Li, and Ruigang Yang. Joint 3d instance segmentation and object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1839–1849, 2020.

[31] Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay Vasudevan. End-to-end multi-view fusion for 3d object detection in lidar point clouds. In *Conference on Robot Learning*, pages 923–932, 2020.

[32] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.