PV-RCNN++: Point-Voxel Feature Set Abstraction With Local Vector Representation for 3D Object Detection

Shaoshuai Shi, Li Jiang, Jiajun Deng, Zhe Wang, Chaoxu Guo,
Jianping Shi, Xiaogang Wang, Hongsheng Li

Abstract—3D object detection is receiving increasing attention from both industry and academia thanks to its wide applications in various fields. In this paper, we propose the Point-Voxel Region-based Convolution Neural Networks (PV-RCNNs) for 3D object detection from point clouds. First, we propose a novel 3D detector, PV-RCNN, which consists of two steps: the voxel-to-keypoint scene encoding and keypoint-to-grid RoI feature abstraction. These two steps deeply integrate the 3D voxel CNN with the PointNet-based set abstraction for extracting discriminative features. Second, we propose an advanced framework, PV-RCNN++, for more efficient and accurate 3D object detection. It consists of two major improvements: the sectorized proposal-centric strategy for efficiently producing more representative keypoints, and the VectorPool aggregation for better aggregating local point features with much less resource consumption. With these two strategies, our PV-RCNN++ is more than 2x faster than PV-RCNN, while also achieving better performance on the large-scale Waymo Open Dataset with $150m \times 150m$ detection range. Also, our proposed PV-RCNNs achieve state-of-the-art 3D detection performance on both the Waymo Open Dataset and the highly-competitive KITTI benchmark. The source code is available at https://github.com/open-mmlab/OpenPCDet.

Index Terms—3D object detection, point clouds, LiDAR, autonomous driving, sparse convolution, Waymo Open Dataset.

1 Introduction

3D object detection is indispensable to lots of real-world applications like autonomous driving, intelligent traffic system and robotics. The most commonly-used data representation for 3D object detection is point cloud, which is captured by 3D sensors (*e.g.*, LiDAR sensors) to depict the 3D scene. The sparsity and irregularity of point cloud make it challenging to extend 2D detection methods [1], [2], [3], [4], [5], [6], [7] to 3D object detection from point clouds.

To learn discriminative features from the sparse and irregular points, some 3D detection methods [8], [9], [10], [11], [12], [13], [14], [15], [16], [17] voxelize the points, then process the regular voxels by conventional Convolution Neural Networks (CNNs) and well-studied 2D detection heads [2], [3]. But the voxelization inevitably brings information loss, thus degrading the fine-grained localization accuracy of these voxel-based methods. In contrast, powered by the pioneer PointNet [18], [19], some other methods directly learn effective features from raw point cloud and predict 3D boxes around the foreground points [20], [21], [22], [23]. These point-based methods preserve accurate point positions and have flexible receptive field due to the radius-based local feature aggregation operations (*e.g.*, set abstraction [19]), but are generally computationally intensive.

We observe that, the voxel representation with dense bird's eye view heads generally produce better 3D box proposals with higher recall rate [24], while the point-wise features benefit the proposal refinement with fine-grained and accurate point locations. Motivated by these observations, we propose a unified framework, namely, Point-Voxel Region-based Convolutional Neural Networks (PV-RCNNs), to take the best of both voxel and point representations. The principle lies in the fact that the voxel-based operations can efficiently encode multi-scale features and generate high-quality 3D proposals, while the point-based operation can preserve accurate location information with flexible receptive fields for proposal refinement.

In this paper, we first introduce a novel two-stage detector, **PV-RCNN**, for accurate 3D object detection through a two-step strategy of point-voxel feature aggregation. The first step is voxel-to-keypoint scene encoding, where a voxel CNN with sparse convolutions is adopted for voxel feature learning and proposal generation. The multi-scale voxel features are then summarized into a small set of keypoints by set abstraction [19]. The keypoints with accurate point locations are sampled by farthest point sampling (FPS) from the raw point cloud. The second step is keypoint-to-grid RoI feature abstraction, where we propose the RoI-grid pooling module to aggregate the above keypoint features to the RoI grids of each proposal. It encodes multi-scale contextual information to form regular grid features for the following proposal refinement.

We then propose an advanced two-stage detection network, **PV-RCNN++**, on top of PV-RCNN, for achieving more accurate, efficient and practical 3D object detection. The improvements of PV-RCNN++ over PV-RCNN lie in two aspects. First, we propose a sectorized proposal-centric keypoint sampling strategy, where the limited number of

- *Shaoshuai Shi, Xiaogang Wang, Hongsheng Li are with the Department of Electronic Engineering at The Chinese University of Hong Kong, Hong Kong. Li Jiang is with the Department of Computer Science and Engineering at The Chinese University of Hong Kong, Hong Kong. Jiajun Deng is with the University of Science and Technology of China, China. Zhe Wang, Chaoxu Guo, and Jianping Shi are with SenseTime Research.*
- *Email: shaoshuaics@gmail.com, hsli@ee.cuhk.edu.hk*

keypoints is concentrated in and around the 3D proposals to encode more effective features for proposal refinement. Meanwhile, the sectorized farthest point sampling is conducted to parallelly sample keypoints in different sectors, which accelerates the keypoint sampling process, while ensuring the uniform distribution of keypoints. Our proposed keypoint sampling strategy is much faster and more effective than the vanilla farthest point sampling that has a quadratic complexity. The efficiency of the whole framework is thus greatly improved, which is particularly important for large-scale 3D scenes with millions of points. Second, we propose a novel local feature aggregation module, VectorPool aggregation, for more effective and efficient local feature encoding on sparse and irregular point cloud. We argue that the relative point locations in a local region are robust, effective and discriminative information for describing local geometry. We propose to split the 3D local space into regular and compact voxels, the features of which are sequentially concatenated to form a hyper feature vector. The voxel features in different locations are encoded with separate kernels to generate position-sensitive local features. In this way, different feature channels of the hyper feature vector are incorporated with different local location information. Compared with previous set abstraction operation for local feature aggregation, our proposed VectorPool aggregation can efficiently handle a very large number of centric points due to the compact local feature representation. Equipped with the VectorPool aggregation in both voxel-based backbone and RoI-grid pooling module, our proposed PV-RCNN++ is more memory-friendly and faster than previous counterparts with comparable or even better performance, which helps in establishing a practical 3D detector for resource-limited devices.

In summary, our proposed PV-RCNNs have three major contributions. (1) Our proposed PV-RCNN adopts two novel operations, voxel-to-keypoint scene encoding and keypoint-to-grid RoI feature abstraction, to deeply integrate the advantages of both point-based and voxel-based feature learning strategies. (2) Our proposed PV-RCNN++ takes a step in more practical 3D detection system with better performance, less resource consumption and faster running speed. This is enabled by our proposed sectorized proposal-centric keypoint sampling strategy to obtain more representative keypoints with faster speed, and is also powered by our novel VectorPool aggregation for achieving local aggregation on a very large number of central points with less resource consumption and more effective representation. (3) Our proposed 3D detectors surpass all published methods with remarkable margins on multiple challenging 3D detection benchmarks. In particular, our PV-RCNN++ achieves state-of-the-art results on the Waymo Open dataset with 10 FPS inference speed for the $150m \times 150m$ detection range. The source code is available at https://github.com/open-mmlab/OpenPCDet [25].

2 Related Work

2D Object Detection with RGB Images. We summarize the 2D object detectors into anchor-based and anchor-free directions. The approaches following the anchor-based paradigm advocate the empirically pre-defined anchor boxes to perform detection, where object detectors are further divided into two-stage [2], [26], [27], [28], [29] and one-stage [3], [7], [30], [31] categories. Two-stage approaches generally extract the proposal-align features for box refinement, and one-stage ones directly perform detection on feature maps. On the other hand, studies of the anchor-free direction mainly fall into keypoint-based [32], [33], [34], [35] and center-based [36], [37], [38], [39] paradigms. The keypoint-based methods represent bounding boxes as keypoints, *i.e.*, corner/extreme points, grid points and a set of bounded points, and the center-based approaches predict the bounding box from foreground points inside objects. Besides, the recently proposed DETR [40] leverages widely adopted transformers to detect objects with attention mechanism and self-learned object queries, which also gets rid of anchor boxes.

3D Object Detection with RGB images. Image-based 3D object detection aims to estimate 3D bounding boxes from a monocular image or stereo images. Mono3D [41] generates 3D region proposals with ground-plane assumption, which are scored by exploiting the semantic knowledge from images. The following works [42], [43] incorporate the relations between 2D and 3D boxes as geometric constraint. M3D-RPN [44] introduces an end-to-end 3D region proposal network with depth-aware convolutions. [45], [46], [47], [48], [49] predict 3D boxes based on a wire-frame template obtained from CAD models. RTM3D [50] performs coarse keypoints detection to localize 3d objects in real-time. On the stereo side, Stereo R-CNN [51], [52] capitalizes on a stereo RPN to associate proposals from left and right images. DSGN [53] introduces the differentiable 3D geometric volume to simultaneously learn depth information and semantic cues in an end-to-end optimized pipelines. Pseudo-LiDARs [52], [54], [55] propose to covert the image pixels to artificial point clouds, where the LiDAR-based detectors can operate on them for 3D box estimation. These image-based 3D detection methods suffer from inaccurate depth estimation and can only generate coarse 3D bounding boxes.

Representation Learning on Point Clouds. Recently representation learning on point clouds has drawn lots of attention for improving the performance of point cloud classification and segmentation [10], [18], [19], [56], [57], [58], [59], [60], [61], [62], [63], [64], [65], [66], [67]. In terms of 3D detection, previous methods generally project the point clouds to regular bird view grids [9], [12] or 3D voxels [**?**], [10] for processing point clouds with 2D/3D CNN. 3D sparse convolution [68], [69] are adopted in [16], [24] to effectively learn sparse voxel-wise features from the point clouds. Qi et al. [18], [19] proposes the PointNet to directly learn point-wise features from the raw points, where set abstraction operation enables flexible receptive fields by setting different search radii. [70] combines both voxel-based CNN and point-based multi-layer percetron (MLP) network for efficient point feature learning. In comparison, our PV-RCNNs take advantages from both the voxel-based (*i.e.*, 3D sparse convolution) and PointNet-based (*i.e.*, set abstraction operation) strategies to enable both high-quality 3D proposal generation and flexible receptive fields for improving the 3D detection performance.

3D Object Detection with Grid-based Representation on Point Clouds. To tackle the irregular data format of point

clouds, most existing works project the point clouds to regular grids. The pioneer work MV3D [9] projects the point clouds to 2D bird view grids and places lots of predefined 3D anchors for generating 3D boxes, and the following works [11], [13], [17], [71], [72], [73] develop better strategies for multi-sensor fusion. [12], [14], [15] introduce more efficient frameworks with bird-eye view representation while [74] proposes to fuse grid features of multiple scales. MVF [75] integrates 2D features from bird-eye view and perspective view before projecting points into pillar representations [15]. Some other works [8], [10] divide the points into 3D voxels to be processed by 3D CNN. 3D sparse convolution [69] is introduced by [16] for efficient 3D voxel processing. [76], [77] utilize multiple detection heads for detecting 3D objects with different scales. In addition, [78], [79] predicts bounding box parameters following the anchor-free paradigm. These grid-based methods are generally efficient for accurate 3D proposal generation but the receptive fields are constraint by the kernel size of 2D/3D convolutions.

3D Object Detection with Point-based Representation on Point Clouds. F-PointNet [20] first proposes to apply PointNet [18], [19] for 3D detection from the cropped points based on the 2D image boxes. PointRCNN [21] generates 3D proposals directly from 3D raw points for 3D detection with point clouds only. [80] proposes the hough voting strategy for better object feature grouping. 3DSSD [81] introduces F-FPS as a complement of D-FPS and develops a one stage object detector operating on raw points. These point-based methods are mostly based on the PointNet series, especially the set abstraction operation [19], which enables flexible receptive fields for point cloud feature learning.

3D Object Detection with Hybrid Representation on Point Clouds. There are also some works that utilize both the point-based and grid-based point representations. STD [23] proposes the PointsPool to transform point-wise features to voxel features for refining the proposals. Fast Point R-CNN [82] fuses the deep voxel features with raw point data for 3D detection. Part-A2-Net [24] aggregates the point-wise part locations by the voxel-based RoI-aware pooling to improve the 3D detection performance. However, these methods do not fuse the deeper features from both two representations and also not fully explore the advantages of these representations. In contrast, our proposed PV-RCNN frameworks explore on how to deeply aggregate the features by learning with both point-based (*i.e.*, set abstraction) and voxel-based (*i.e.*, sparse convolution) feature learning operations to boost the performance of 3D detection.

3 PRELIMINARIES

State-of-the-art 3D object detectors mostly adopt two-stage frameworks that generally achieve higher performance by splitting the complex detection problem into two stages, including the region proposal generation and per-proposal refinement. In this section, we briefly introduce our chosen strategy for the fundamental feature extraction and proposal generation stage, and then discuss the challenges of straightforward methods for the 3D proposal refinement.

3D Voxel CNN and Proposal Generation. Voxel CNN with 3D sparse convolution [68], [69] is a popular choice of state-of-the-art 3D detectors [16], [24], [83] thanks to its efficiency of converting irregular points into 3D sparse feature volumes. The input points $\mathcal{P}$ are first divided into small voxels with spatial resolution of $L \times W \times H$, where non-empty voxel features are directly calculated by averaging the features of inside points (*e.g.*, 3D coordinates and reflectance intensities). The network utilizes a series of 3D sparse convolution to gradually convert the points into feature volumes with $1\times, 2\times, 4\times, 8\times$ downsampled sizes. Such sparse feature volumes at each level can be viewed as a set of sparse voxel-wise feature vectors. The Voxel CNN backbone can be naturally combined with the 2D detection heads [2], [3], [38] by converting the encoded $8\times$ downsampled 3D feature volumes into 2D bird-view feature maps. Specifically, we follow [16] to stack the 3D feature volumes along Z axis to obtain the $\frac{L}{8} \times \frac{W}{8}$ bird-view feature maps, which can be combined with both the anchor-based head [3] and the center head [84] for high quality 3D proposal generation.

Discussions on 3D Proposal Refinement. In the proposal refinement stage, the proposal-specific features are required to be extracted from the 3D feature volumes or 2D maps. Intuitively, the feature extraction should be conducted in the 3D space instead of the 2D feature maps to learn more fine-grained features. However, these 3D feature volumes from the 3D voxel CNN have major limitations in the following aspects. (i) These feature volumes are generally of low spatial resolution as they are downsampled by up to 8 times, which hinders accurate localization of objects. (ii) Even if one can upsample to obtain feature volumes/maps with larger spatial sizes, they are generally still quite sparse. The commonly used trilinear or bilinear interpolation in the RoIPool/RoIAlign [6] operations can only extract features from very small neighborhoods (*i.e.*, 4 and 8 for bilinear and trilinear interpolation respectively), which would therefore obtain features with mostly zeros and waste much computations and memory for proposal refinement.

On the other hand, the point-based local feature aggregation methods [19] have shown strong capability of encoding sparse features from local neighborhoods with arbitrary scales. We therefore propose to incorporate a 3D voxel CNN with the point-based local feature aggregation operation for conducting accurate and robust proposal refinement.

4 PV-RCNN: POINT-VOXEL FEATURE SET ABSTRACTION FOR 3D OBJECT DETECTION

To learn effective features from sparse points, state-of-the-art 3D detectors are based on either 3D voxel CNNs with sparse convolution or PointNet-based operators. Generally, the 3D voxel CNNs with sparse convolutional layers are more efficient and are able to generate high-quality 3D proposals [24], [84], while the PointNet-based operators naturally preserve accurate point locations and can capture rich context information with flexible receptive fields [19].

We propose a novel two-stage 3D detection framework, PV-RCNN, to deeply integrate the advantages of two types of operators for 3D object detection from point clouds. As shown in Fig. 1, PV-RCNN consists of a 3D voxel CNN with sparse convolution as the backbone for efficient feature encoding and proposal generation. Given each 3D proposal, we propose to generate the proposal-aligned features in two novel steps for proposal refinement, which consists of

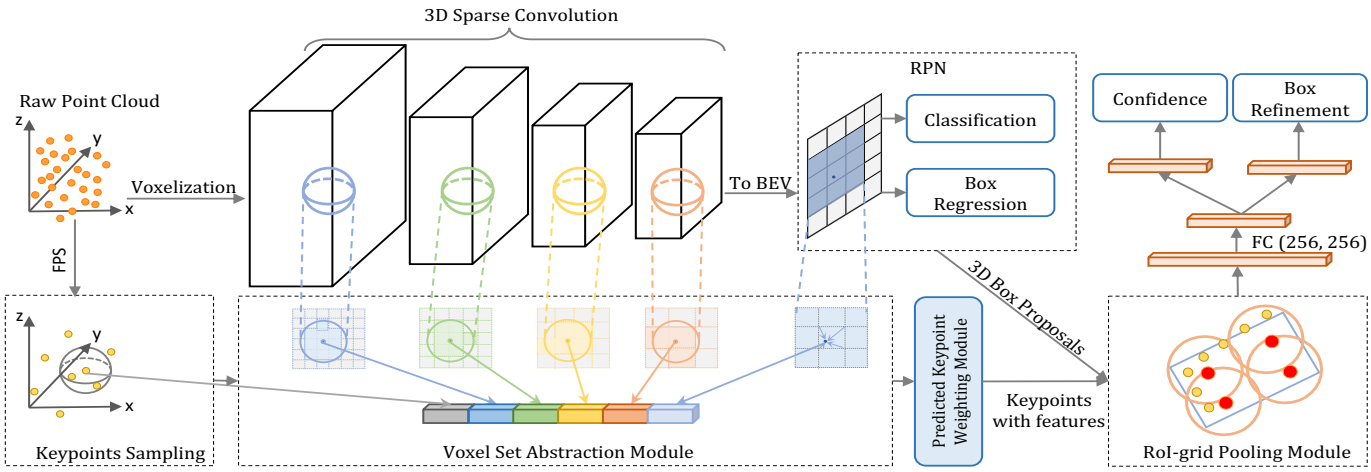

Fig. 1. The overall architecture of our proposed PV-RCNN. The raw point clouds are first voxelized to feed into the 3D sparse convolution based encoder to learn multi-scale semantic features and generate 3D object proposals. Then the learned voxel-wise feature volumes at multiple neural layers are summarized into a small set of key points via the novel voxel set abstraction module. Finally the keypoint features are aggregated to the RoI-grid points to learn proposal specific features for fine-grained proposal refinement and confidence prediction.

the voxel-to-keypoint scene encoding and keypoint-to-grid RoI feature abstraction. They are introduced in Sec. 4.1 and Sec. 4.2, respectively.

4.1 Voxel-to-Keypoint Scene Encoding

Our proposed PV-RCNN first aggregates the voxel-wise scene features at multiple neural layers of 3D voxel CNN into a small number of keypoints, which bridge the 3D voxel CNN feature encoder and the proposal refinement network.

Keypoints Sampling. We simply adopt the farthest point sampling (FPS) algorithm to sample a small number of keypoints $\mathcal{K} = \{p_1, \cdots, p_n\}$ from the point clouds $\mathcal{P}$, where n is a hyper-parameter (n=4,096 for Waymo Open Dataset [85] and n=2,048 for KITTI dataset [86]). It encourages that the keypoints are uniformly distributed around non-empty voxels and can be representative to the overall scene.

Voxel Set Abstraction Module. We propose the *Voxel Set Abstraction* (VSA) module to encode multi-scale semantic features from 3D feature volumes to the keypoints. The set abstraction [19] is adopted for aggregating voxel-wise feature volumes. Different with the original set abstraction, the surrounding local points are now regular voxel-wise semantic features from 3D voxel CNN, instead of the neighboring raw points with features learned by PointNet [18].

Specifically, denote $\mathcal{F}^{(l_k)} = \{f_1^{(l_k)}, \ldots, f_{N_k}^{(l_k)}\}$ as the set of voxel-wise features in the k-th level of 3D voxel CNN, $\mathcal{V}^{(l_k)} = \{v_1^{(l_k)}, \ldots, v_{N_k}^{(l_k)}\}$ as their corresponding 3D coordinates in the uniform 3D metric space, where N_k is the number of non-empty voxels in the k-th level. For each keypoint p_i, to retrieve the set of neighboring voxel-wise feature vectors, we first identify its neighboring non-empty voxels at the k-th level within a radius r_k as

$$S_i^{(l_k)} = \left\{ \left(f_j^{(l_k)}, v_j^{(l_k)} - p_i \right) \middle| \begin{array}{l} \left\| v_j^{(l_k)} - p_i \right\| < r_k, \\ \forall v_j^{(l_k)} \in \mathcal{V}^{(l_k)}, \\ \forall f_j^{(l_k)} \in \mathcal{F}^{(l_k)} \end{array} \right\}, \quad (1)$$

where the local relative position $v_j^{(l_k)} - p_i$ is concatenated to indicate the relative location of $f_j^{(l_k)}$. The features within neighboring set $S_i^{(l_k)}$ are then aggregated by a PointNet-block [18] to generate the keypoint feature as

$$f_i^{(pv_k)} = \max \left\{ G \left(\mathcal{M} \left(S_i^{(l_k)} \right) \right) \right\}, \quad (2)$$

where $\mathcal{M}(\cdot)$ denotes randomly sampling at most T_k voxels from the neighboring set $S_i^{(l_k)}$ for saving computations, $G(\cdot)$ denotes a multi-layer perceptron (MLP) network to encode voxel-wise features and relative locations. The operation $\max(\cdot)$ maps diverse number of neighboring voxel features to a single keypoint feature $f_i^{(pv_k)}$. Here multiple radii are utilized to capture richer contextual information.

The above voxel feature aggregation is performed at the outputs of different scales of the 3D voxel CNN, and the aggregated features from different scales are concatenated to obtain the multi-scale semantic feature for keypoint p_i as

$$f_i^{(pv)} = \left[f_i^{(pv_1)}, f_i^{(pv_2)}, f_i^{(pv_3)}, f_i^{(pv_4)} \right], \text{ for } i = 1, \ldots, n, \quad (3)$$

where the generated feature $f_i^{(pv)}$ incorporates both the voxel-wise feature $f_j^{(l_k)}$ from 3D CNN and the PointNet-based features from Eq. (2). Moreover, the 3D coordinates of p_i also naturally preserves accurate location information.

Further Enriching Keypoint Features. We further enrich the keypoint features with the raw points $\mathcal{P}$ and with the $8\times$ downsampled 2D bird-view feature maps, where the raw points can partially make up the quantization loss of the point voxelization while the 2D bird-view maps have larger receptive fields along the Z axis. Specifically, the raw point feature $f_i^{(raw)}$ is also aggregated as that in Eq. (2), while the bird-view features $f_i^{(bev)}$ are obtained by performing bilinear interpolation with projected 2D keypoints on the 2D feature maps. Hence, the keypoint features for p_i is further enriched by concatenating all its associated features as

$$f_i^{(p)} = \left[f_i^{(pv)}, f_i^{(raw)}, f_i^{(bev)} \right], \text{ for } i = 1, \ldots, n, \quad (4)$$

which have the strong capacity of preserving 3D structural information of the entire scene for the following fine-grained proposal refinement step.

Predicted Keypoint Weighting. As mentioned before, the keypoints are chosen by farthest point sampling and

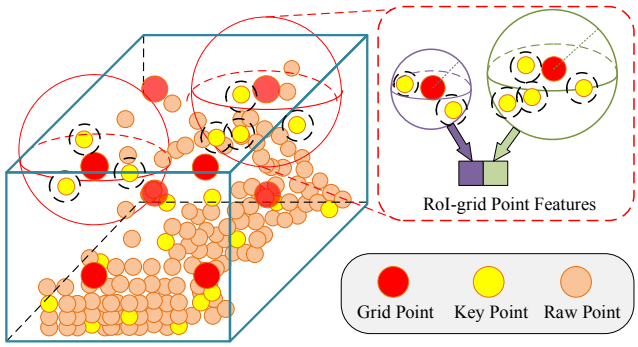

Fig. 2. Illustration of RoI-grid pooling module. Rich context information of each 3D RoI is aggregated by the set abstraction operation with multiple receptive fields.

some of them might only represent the background regions. Intuitively, keypoints belonging to the foreground objects should contribute more to the proposal refinement, while the ones from the background regions should contribute less. Hence, we propose a *Predicted Keypoint Weighting* (PKW) module to re-weight the keypoint features with extra supervisions from point segmentation. The segmentation labels are free-of-charge and can be generated from the 3D box annotations, *i.e.*, by checking whether each keypoint is inside or outside of a ground-truth 3D box, since the 3D objects in autonomous driving scenes are naturally separated in the 3D space. This module can be formulated as

$$\tilde{f}_i^{(p)} = \mathcal{A}(f_i^{(p)}) \cdot f_i^{(p)}, \tag{5}$$

where $\mathcal{A}(\cdot)$ is a three-layer MLP network with a sigmoid function to predict foreground confidence. It is trained with focal loss [7] (default hyper-parameters) to handle the imbalanced foreground/background points of the training set.

4.2 Keypoint-to-Grid RoI Feature Abstraction

After summarizing the multi-scale semantic features into a small number of keypoints, in this step, we propose keypoint-to-grid RoI feature abstraction to generate accurate proposal-aligned features from the keypoint features $\tilde{\mathcal{F}} = \{\tilde{f}_1^{(p)}, \cdots, \tilde{f}_n^{(p)}\}$ for fine-grained proposal refinement.

RoI-grid Pooling via Set Abstraction. Given each 3D proposal, as shown in Fig. 2, we propose the *RoI-grid pooling* module to aggregate the keypoint features to the RoI-grid points with multiple receptive fields. We uniformly sample $6\times6\times6$ grid points within each 3D proposal, which are then flattened and denoted as $\mathcal{G} = \{g_1, \cdots, g_{216}\}$. We utilize set abstraction to obtain features of grid points via aggregating the keypoint features. Specifically, we firstly identify the neighboring keypoints of grid point g_i as

$$\tilde{\Psi} = \left\{ \left(\tilde{f}_j^{(p)}, p_j - g_i\right) \,\middle|\, \begin{array}{l} \|p_j - g_i\| < \tilde{r}, \\ \forall p_j \in \mathcal{K}, \forall \tilde{f}_j^{(p)} \in \tilde{\mathcal{F}} \end{array} \right\}, \tag{6}$$

where $p_j - g_i$ is appended to indicate the local relative location within the ball of radius $\tilde{r}$. A PointNet-block [18] is then adopted to aggregate the neighboring keypoint feature set $\tilde{\Psi}$ to obtain the feature for grid point g_i as

$$\tilde{f}_i^{(g)} = \max\left\{G\left(\mathcal{M}\left(\tilde{\Psi}\right)\right)\right\}, \tag{7}$$

where $\mathcal{M}(\cdot)$ and $G(\cdot)$ are defined in the same way as Eq. (2). We set multiple radii $\tilde{r}$ and aggregate keypoint features with different receptive fields, which are concatenated together for capturing richer multi-scale contextual information.

Next, all RoI-grid features of the same RoI can be vectorized and transformed by a two-layer MLP with 256 feature dimensions to represent the overall features of proposal.

Our proposed RoI-grid pooling operation can aggregate much richer contextual information than the previous RoI-pooling/RoI-align operation [21], [23], [24]. It is because a single keypoint can contribute to multiple RoI-grid points due to the overlapped neighboring balls of RoI-grid points, and their receptive fields are even beyond the RoI boundaries by capturing the contextual keypoint features outside the 3D RoI. In contrast, the previous state-of-the-art methods either simply average all point-wise features within the proposal as the RoI feature [21], or pool many uninformative zeros as the RoI features because of the very sparse point-wise features [23], [24].

Proposal Refinement and Confidence Prediction. Given the RoI feature extracted by the above RoI-grid pooling module, the refinement network learns to predict the size and location (*i.e.* center, size and orientation) residuals relative to the 3D proposal box. Two sibling sub-networks are employed for confidence prediction and proposal refinement. Each sub-network consists of a two-layer MLP and a linear prediction layer. We follow [24] to conduct the IoU-based confidence prediction. The binary cross-entropy loss is adopted to optimize the IoU branch while the box residuals are optimized with smooth-L1 loss.

4.3 Training Losses

The proposed PV-RCNN framework is trained end-to-end with the region proposal loss L_{rpn}, keypoint segmentation loss L_{seg} and the proposal refinement loss L_{rcnn}. (i) We adopt the same region proposal loss L_{rpn} as that in [16],

$$L_{\text{rpn}} = L_{\text{cls}} + \beta \sum_{\text{r}\in O} \mathcal{L}_{\text{smooth-L1}}(\widehat{\Delta \text{r}^a}, \Delta \text{r}^a) + \alpha L_{dir}, \tag{8}$$

where $O = \{x, y, z, l, h, w, \theta\}$, the anchor classification loss L_{cls} is calculated with the focal loss [7], L_{dir} is a binary classification loss for orientation to eliminate the ambiguity of $\Delta\theta^a$ as in [16], and smooth-L1 loss is for anchor box regression with the predicted residual $\widehat{\Delta \text{r}^a}$ and regression target Δr^a. Loss weights are set as $\alpha = 0.2$ and $\beta = 2.0$ in the training process. (ii) The keypoint segmentation loss L_{seg} is also conducted with the focal loss as mentioned in Sec. 4.1. (iii) The proposal refinement loss L_{rcnn} includes the IoU-guided confidence prediction loss L_{iou} and the box refinement loss as

$$L_{\text{rcnn}} = L_{\text{iou}} + \sum_{\text{r}\in O} \mathcal{L}_{\text{smooth-L1}}(\widehat{\Delta \text{r}^p}, \Delta \text{r}^p), \tag{9}$$

where $\widehat{\Delta \text{r}^p}$ is the predicted box residual and Δr^p is the proposal regression target that is encoded same with Δr^a. The overall training loss are then the sum of these three losses with equal loss weights.

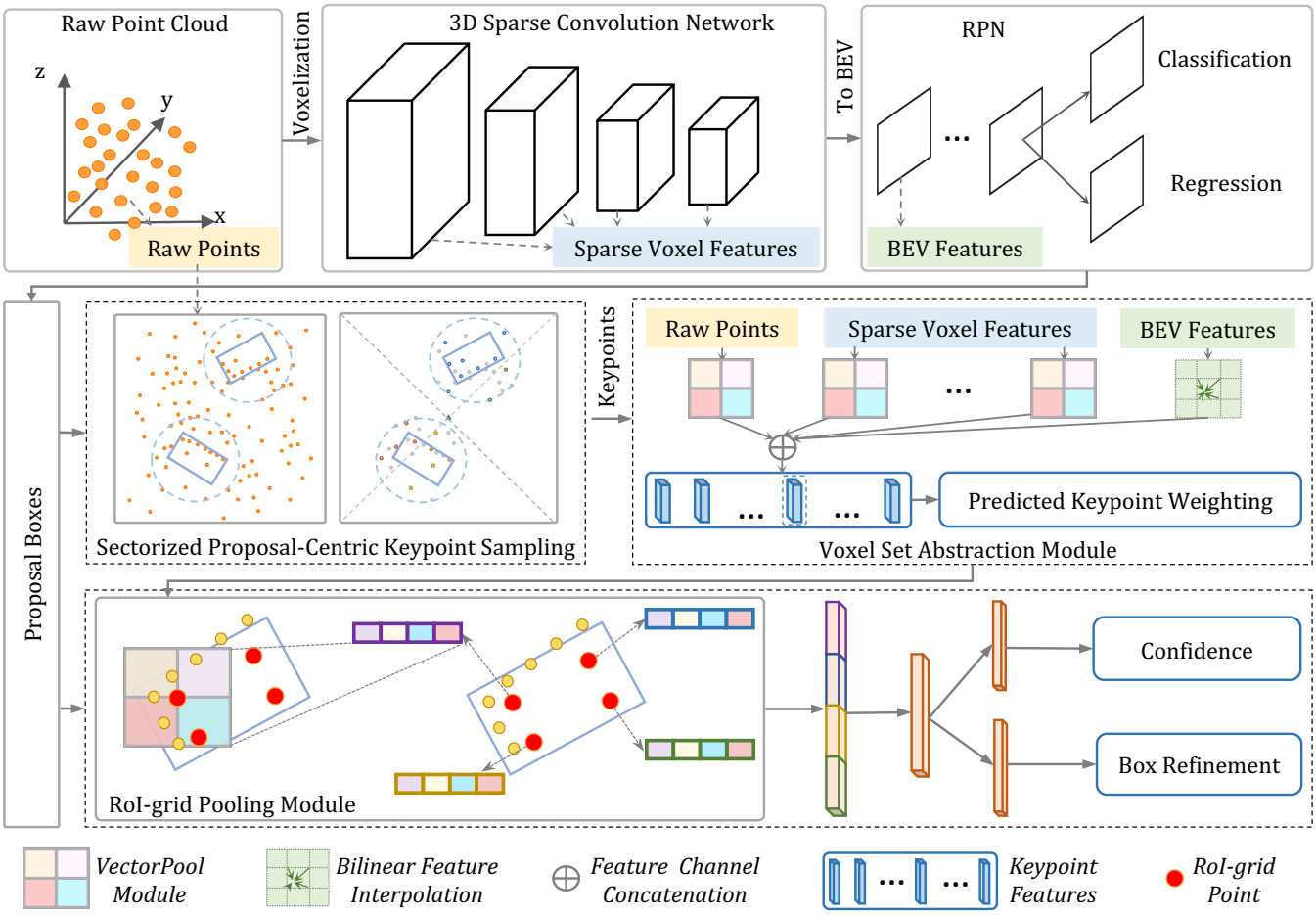

Fig. 3. The overall architecture of our proposed PV-RCNN++. We propose the sectorized proposal-centric keypoint sampling module to concentrate keypoints to the neighborhoods of 3D proposals while it can also accelerate the process with sectorized farthest point sampling. Moreover, our proposed VectorPool module is utilized in both the voxel set abstraction module and the RoI-grid pooling module to improve the local feature aggregation and save memory/computation resources

5 PV-RCNN++: Faster and Better 3D Detection With PV-RCNN Framework

To make our PV-RCNN framework more practical for real-world applications, we propose an improved version of PV-RCNN, *i.e.*, PV-RCNN++ framework, for more accurate and efficient 3D object detection with less resource consumption.

As shown in Fig. 3, we present two novel modules to improve both the accuracy and efficiency of the PV-RCNN framework. One is the sectorized proposal-centric strategy for much faster and better keypoint sampling, and the other one is the VectorPool aggregation module for more effective and efficient local feature aggregation from large-scale point clouds. These two novel modules are adopted to replace their counterparts in the PV-RCNN framework, which are introduced in Sec. 5.1 and Sec. 5.2, respectively.

5.1 Sectorized Proposal-Centric Sampling for Efficient and Representative Keypoint Sampling

The keypoint sampling is critical for our PV-RCNN framework since keypoints bridge the point-voxel representations and influence the performance of proposal refinement. However, the previous keypoint sampling algorithm has two main drawbacks. (i) Farthest point sampling is time-consuming due to its quadratic complexity, which hinders the training and inference speed of PV-RCNN, especially for keypoint sampling of large-scale points. (ii) Previous algorithm would generate a large number of background keypoints that are generally useless to the proposal refinement, since only the keypoints around the proposals can be retrieved by the RoI-grid pooling module. To mitigate these drawbacks, we propose a more efficient and effective keypoint sampling algorithm for 3D object detection.

Sectorized Proposal-Centric (SPC) Keypoint Sampling. As discussed above, the number of keypoints is limited and vanilla farthest point sampling algorithm would generate wasteful keypoints in the background regions, which decrease the capability of keypoints to well represent objects for proposal refinement. Hence, as shown in Fig. 4, we propose the *Sectorized Proposal-Centric* (SPC) keypoint sampling to uniformly sample keypoints from more concentrated neighboring regions of proposals, while also being much faster than the vanilla farthest point sampling algorithm,

Specifically, we denote the raw point clouds as $\mathcal{P}$, and denote the centers and sizes of 3D proposal as $\mathcal{C}$ and $\mathcal{D}$, respectively. To better generate the set of restricted keypoints, we first restrict the keypoint candidates $\mathcal{P}'$ to the neighboring point sets of all proposals as

$$\mathcal{P}' = \left\{ p_i \;\middle|\; \begin{array}{c} \|p_i - c_j\| < \frac{\max(dx_j, dy_j, dz_j)}{2} + r^{(s)}, \\ (dx_j, dy_j, dz_j) \in \mathcal{D} \subset \mathbb{R}^3, \\ p_i \in \mathcal{P} \subset \mathbb{R}^3, c_j \in \mathcal{C} \subset \mathbb{R}^3, \end{array} \right\}, \quad (10)$$

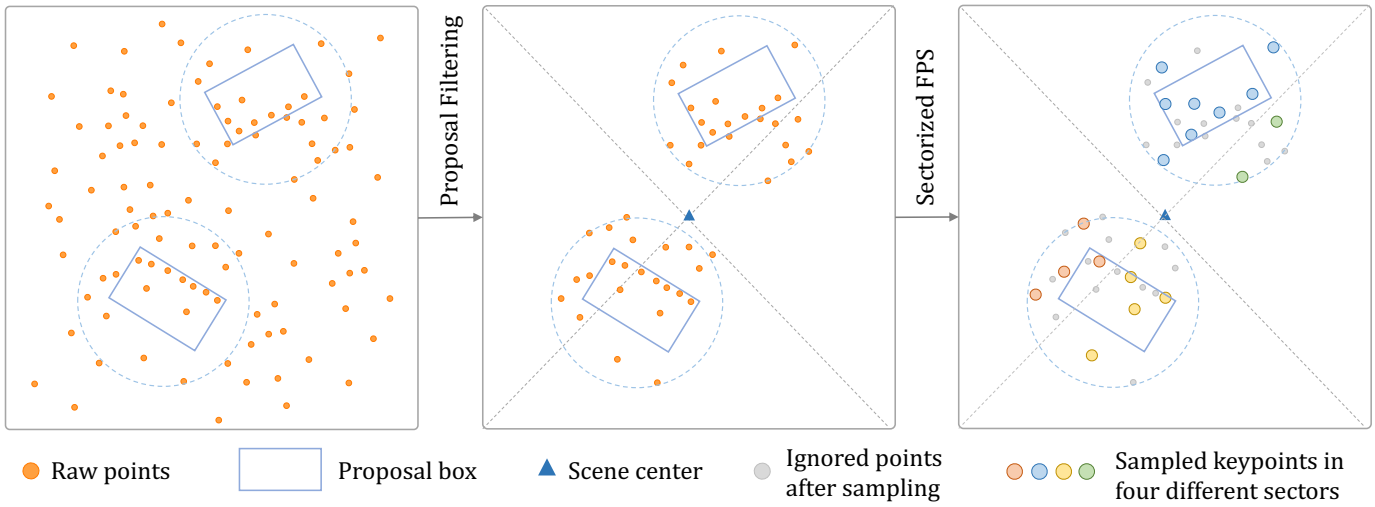

Fig. 4. Illustration of Sectorized Proposal-Centric (SPC) keypoint sampling. It contains two steps, where the first proposal filter step concentrates the limited number of keypoints to the neighborhoods of proposals, and the following sectorized-FPS step divides the whole scene into several sectors for accelerating the keypoint sampling process while also keeping the keypoints uniformly distributed.

where $r^{(s)}$ is a hyperparameter indicating the maximum extended radius of the proposals, and dx_j, dy_j, dz_j are the sizes of the 3D proposal. Through this proposal-centric filtering process, the number of candidate keypoints for sampling is greatly reduced from $|\mathcal{P}|$ to $|\mathcal{P}'|$, which not only reduces the time complexity of the follow-up keypoint sampling, but also concentrates the limited number of keypoints to better encode the neighboring regions of proposals.

To further parallelize the keypoint sampling process for acceleration, as shown in Fig. 4, we divide the proposal-centric point set $\mathcal{P}'$ into s sectors centered at the scene center, and the point set of the k-th sector can be represented as

$$S'_k = \left\{ p_i \;\middle|\; \begin{matrix} \lfloor (\arctan(py_i, px_i) + \pi) \times \frac{s}{2\pi} \rfloor = k - 1, \\ p_i = (px_i, py_i, pz_i) \in \mathcal{P}' \subset \mathbb{R}^3 \end{matrix} \right\}, \quad (11)$$

where $k \in \{1, \ldots, s\}$, and $\arctan(py_i, px_i) \in (-\pi, \pi]$ is to indicate the angle between the positive X axis and the ray ended with (px_i, py_i). Through this process, we divide the task of sampling n keypoints into s subtasks of sampling local keypoints, where the k-th sector samples $\left\lfloor \frac{|S'_k|}{|\mathcal{P}'|} \times n \right\rfloor$ keypoints from S'_k. These subtasks are eligible to be executed in parallel on GPUs, while the scale of keypoint sampling (*i.e.*, the time complexity) is further reduced from $|\mathcal{P}'|$ to $\max_{k \in \{1,\ldots,s\}} |S'_k|$. Note that here we adopt the farthest point sampling algorithm in each subtask since we find that farthest point sampling can generate more uniformly distributed keypoints to better cover the whole regions, which is critical for the final detection performance.

Therefore, our proposed sectorized proposal-centric keypoint sampling greatly reduces the scale of keypoint sampling from $|\mathcal{P}|$ to the much smaller $\max_{k \in \{1,\ldots,s\}} |S'_k|$, which not only effectively accelerates the keypoint sampling process, but also increases the capability of keypoint feature representation by concentrating the keypoints to the more important neighboring regions of 3D proposals.

5.2 Local Vector Representation for Structure-Preserved Local Feature Learning from Point Clouds

How to aggregate informative features from local point clouds is critical in our proposed point-voxel-based object detection system. As discussed in Sec. 4, PV-RCNN adopts the set abstraction for local feature aggregation in both voxel set abstraction module and RoI grid-pooling. However, we observe that the set abstraction operation can be extremely time- and resource-consuming on large-scale point clouds. Hence, in this section, we first analyze the limitations of set abstraction, and then propose the VectorPool aggregation module for local feature aggregation on the large-scale point clouds, which is integrated into our PV-RCNN++ framework for more accurate and efficient 3D object detection.

Limitations of Set Abstraction. As shown in Eqs. (2) and (7), the set abstraction operation samples T_k point-wise features from each local neighborhood, which are encoded separately by several shared-parameter MLPs. Suppose that there are a total number of N local point-cloud neighborhoods and the feature dimensions of each point is C_{in}, then $N \times T_k$ point-wise features with C_{in} channels should be encoded by the shared-parameter MLP $G(\cdot)$. It generates point-wise features of size $N \times T_k \times C_{out}$ (generally multiple MLPs are utilized and each MLP will generate such a tensor). Both the space complexity and computations would be significant when either the number of local neighborhoods or the number of MLPs or T_k are quite large.

For instance, in our RoI-grid pooling module, the number of RoI-grid points can be very large ($N = 100 \times 6 \times 6 \times 6 = 21,600$) with 100 proposals and grid size 6. This module is therefore slow and also consumes much GPU memory in PV-RCNN, which restricts its capability to be run on lightweight devices with limited computation and memory resources. Moreover, the max-pooling operation in set abstraction abandons the spatial distribution information of local points and harms the representation capability of locally aggregated features from point clouds.

VectorPool Aggregation for Structure-Preserved Local Feature Learning. To extract more informative features from local point-cloud neighborhoods, we propose a novel local feature aggregation operation, *VectorPool aggregation*, which preserves spatial point distributions of local neighborhoods and also costs less memory/computation resources than the commonly used set abstraction. We propose to generate

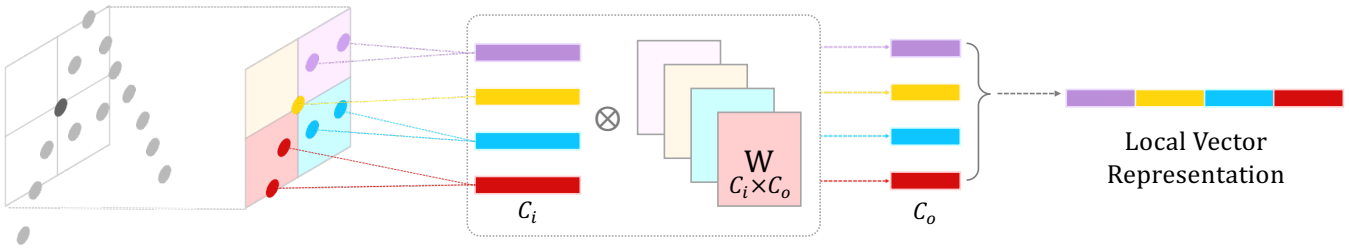

Fig. 5. Illustration of VectorPool aggregation for local feature aggregation from point clouds. The local space around a center point is divided into dense voxels, where the inside point-wise features are generated by interpolating from three nearest neighbors. The features of each volume are encoded with position-specific kernels to generate position-sensitive features, that are sequentially concatenated to generate the local vector representation to explicitly encode the spatial structure information.

position-sensitive features in different local neighborhoods by encoding them with separate kernel weights and separate feature channels, which are then concatenated to explicitly represent the spatial structures of local point features.

Specifically, denote the input point coordinates and features as $\mathcal{P} = \{(f_i, p_i) \mid f_i \in \mathbb{R}^{C_{in}}, p_i \in \mathbb{R}^3, i \in \{1, \dots, M\}\}$, and the centers of N local neighborhoods as $\mathcal{Q} = \{q_i \mid q_i \in \mathbb{R}^3, i \in \{1, \dots, N\}\}$. We are going to extract N local point-wise features with C_{out} channels for each point in $\mathcal{Q}$.

To reduce the parameter size, computational and memory consumption, motivated by [87], we first summarize the point-wise features $f_i \in \mathbb{R}^{C_{in}}$ to more compact representations $\hat{f}_i \in \mathbb{R}^{C_{r_1}}$ with a parameter-free scheme as

$$\hat{f}_i(k) = \sum_{j=0}^{n_r - 1} f_i(jC_{r_1} + k), \; k \in \{0, \dots, C_{r_1} - 1\}, \tag{12}$$

where C_{r_1} is the number of output feature channels and $C_{in} = n_r \times C_{r_1}$. Eq. (12) sums up every n_r input feature channels into one output feature channel to reduce the feature channels by n_r times, which can effectively reduce the resource consumptions of the follow-up processing.

To generate position-sensitive features for a local cubic neighborhood centered at q_k, we split its neighboring cubic space (denote the half length of this cubic space as l) into $n_x \times n_y \times n_z$ small local voxels. The features of each local voxel are generated by aggregating the above point-wise features from its neighborhood. Specifically, this feature aggregation process depends on the set of neighboring points of each local neighborhood q_k, hence we first identify these neighboring points as follows:

$$\mathcal{Y}_k = \left\{ \left(\hat{f}_j, p_j\right) \;\middle|\; \begin{array}{l} \max(|r_x|, |r_y|, |r_z|) < 2 \times l, \\ (r_x, r_y, r_z) = (p_j - q_k) \in \mathbb{R}^3, \\ \forall (p_j, f_j) \in \mathcal{P} \end{array} \right\}. \tag{13}$$

Note that we double the half length (*e.g.*, $2 \times l$) of the original cubic space to contain more neighboring points for feature aggregation of the small local voxels.

For calculating the features of local voxel with index (i_x, i_y, i_z), inspired by [19], we utilize the inverse distance weighted strategy to interpolate its features based on 3 nearest neighbors from $\mathcal{Y}_k$:

$$f^{(v)}_{i_x, i_y, i_z} = \frac{\sum_{j=1}^{3} \left(w_{\sigma(j)} \hat{f}_{\sigma(j)}\right)}{\sum_{j=1}^{3} w_{\sigma(j)}}, \quad \left(\hat{f}_{\sigma(j)}, p_{\sigma(j)}\right) \in \mathcal{Y}_k, \tag{14}$$

where $w_{\sigma(j)} = (||p_{\sigma(j)} - v_{i_x, i_y, i_z}||)^{-1}$ with v_{i_x, i_y, i_z} as the center position of this local voxel, and $\sigma(j)$ finds the index of its three nearest neighbor in $\mathcal{Y}_k$. Here we have $i_x \in \{1, \dots, n_x\}, i_y \in \{1, \dots, n_y\}, i_z \in \{1, \dots, n_z\}$. The resulted features $f^{(v)}_{i_x, i_y, i_z}$ encodes the local features of the specific local voxel (i_x, i_y, i_z) in this local cubic.

There are also two other alternative strategies to aggregate the features of local voxels by simply averaging the features within each local voxel or by randomly choosing one point within each local voxel. Both of them generate lots of empty features in the empty local voxels, which may degrade the performance. In contrast, our interpolation based strategy can generate more effective features even on empty local voxels.

Those features in different local voxels may represent very different local features. Hence, instead of encoding the local features with a shared-parameter MLP as in [19], we propose to encode different local voxels with separate local kernel weights for capturing position-sensitive features as

$$\hat{U}(i_x, i_y, i_z) = E\left(\hat{r}_{i_x, i_y, i_z}, f^{(v)}_{i_x, i_y, i_z}\right) \times W_v, \tag{15}$$

where $\hat{U}(i_x, i_y, i_z) \in \mathbb{R}^{C_{r_2}}$, $\hat{r}_{i_x, i_y, i_z} \in \mathbb{R}^{(3 \times 3)}$ indicating the relative positions of its three nearest neighbors, and $E(\cdot)$ is an operation fusing the relative position and features (by default, we use concatenation). $W_v \in \mathbb{R}^{(9 + C_{r_1}) \times C_{r_2}}$ is the learnable kernel weights for encoding the specific features of local voxel (i_x, i_y, i_z) with channel C_{r_2}, and different positions have different kernel weights for encoding position-sensitive local features.

Finally, we directly sort the local voxel features $\hat{U}(i_x, i_y, i_z)$ according to their spatial order (i_x, i_y, i_z), and their features are sequentially concatenated to generate the final local vector representation as

$$\mathcal{U} = [\hat{U}(1, 1, 1), \dots, \hat{U}(i_x, i_y, i_z), \dots, \hat{U}(n_x, n_y, n_z)], \tag{16}$$

where $\mathcal{U} \in \mathbb{R}^{n_x \times n_y \times n_z \times C_{r_2}}$ encodes the structure-preserved local features by simply assigning the features of different locations to their corresponding feature channels, which naturally preserves the spatial structures of local features in the neighboring space centered at q_k, This local vector representation would be finally processed with another MLPs to encode the local features to C_{out} feature channels for follow-up processing.

Note that compared with set abstraction, our proposed VectorPool aggregation can greatly reduce the needed computations and memory resources by adopting channel summation and utilizing our local vector representation before MLPs. Moreover, instead of conducting max-pooling

on local point-wise features as in the set abstraction, our proposed spatial-structure-preserved local vector representation can encode the position-sensitive local features with different feature channels, to provide more effective representation for local feature learning.

PV-RCNN++ with VectorPool Aggregation for Local Feature Aggregation. Our proposed VectorPool aggregation is integrated in our PV-RCNN++ detection framework, to replace the set abstraction operation in both the voxel set abstraction layer and the RoI-grid pooling module. Thanks to our VectorPool aggregation operation, compared with PV-RCNN framework, our PV-RCNN++ not only consumes much less memory and computation resources, but also achieves better 3D detection performance.

6 Experiments

In this section, we evaluate our proposed framework in the large-scale Waymo Open Dataset [85] and the highly-competitive KITTI dataset [86]. In Sec. 6.1, we first introduce our experimental setup and implementation details. In Sec. 6.2, we conduct extensive ablation experiments and analysis to investigate the individual components of both our PV-RCNN and PV-RCNN++ frameworks. In Sec. 6.3, we present the main results of our PV-RCNN/PV-RCNN++ frameworks and compare with state-of-the-art methods on both the Waymo dataset and the KITTI dataset.

6.1 Experimental Setup

Datasets and Evaluation Metrics. Our methods are evaluated on the following two datasets.

Waymo Open Dataset [85] is currently the largest dataset with LiDAR point clouds for autonomous driving. There are totally 798 training sequences with around 160k LiDAR samples, 202 validation sequences with 40k LiDAR samples and 150 testing sequences with 30k LiDAR samples. It annotated the objects in the full 360° field instead of annotating the front view only as in KITTI dataset. The evaluation metrics are calculated by the official evaluation tools, where the mean average precision (mAP) and the mean average precision weighted by heading (mAPH) are used for evaluation. The 3D IoU threshold is set as 0.7 for vehicle detection and 0.5 for pedestrian/cyclist detection. We present the comparison in terms of two ways. The first way is based on objects' different distances to the sensor: $0-30m$, $30-50m$ and $>50m$. The second way is to split the data into two difficulty levels, where the LEVEL 1 denotes the ground-truth objects with at least 5 inside points while the LEVEL 2 denotes the ground-truth objects with at least 1 inside points. As utilized by the official Waymo evaluation server, the mAPH of LEVEL 2 difficulty is the most important evaluate metric for all experiments.

KITTI Dataset [86] is one of the most popular 3D detection datasets for autonomous driving. There are $7,481$ training samples and $7,518$ test samples. We compare PV-RCNNs with state-of-the-art methods on this highly-competitive 3D detection learderboard [88]. The evaluation metrics are calculated by the official evaluation tools, where the mean average precision (mAP) is calculated with 40 recall positions on three difficulty levels.

Network Architecture. For the PV-RCNN framework, the 3D voxel CNN has four levels (see Fig. 1) with feature dimensions 16, 32, 64, 64, respectively. Their two neighboring radii r_k of each level in the voxel set abstraction module are set as $(0.4\text{m}, 0.8\text{m})$, $(0.8\text{m}, 1.2\text{m})$, $(1.2\text{m}, 2.4\text{m})$, $(2.4\text{m}, 4.8\text{m})$, and the neighborhood radii of set abstraction for raw points are $(0.4\text{m}, 0.8\text{m})$. For the proposed RoI-grid pooling operation, we uniformly sample $6\times6\times6$ grid points in each 3D proposal and the two neighboring radii $\tilde{r}$ of each grid point are $(0.8\text{m}, 1.6\text{m})$.

For the PV-RCNN++ framework, we set the maximum extended radius $r^{(s)} = 1.6m$ for the proposal-centric filtering, and each scene is split into 6 sectors for parallel keypoint sampling. Two VectorPool aggregation operations are adopted to the $4\times$ and $8\times$ feature volumes of the voxel set abstraction module with the half length $l = (1.2m, 2.4m)$ and $l = (2.4m, 4.8m)$ respectively, and both of them have local voxels $n_x = n_y = n_z = 3$ and channel reduced factor $n_r = 2$. The VectorPool aggregation operation on the raw points is set with local voxels $n_x = n_y = n_z = 2$ and without channel reduction. All VectorPool aggregation utilize the concatenation operation as $E(\cdot)$ for encoding relative positions and point-wise features. For RoI-grid pooling, we adopt the same number of RoI-grid points ($6 \times 6 \times 6$) with PV-RCNN, and the utilized VectorPool aggregation has local voxels $n_x = n_y = n_z = 3$, channel reduced factor $n_r = 3$ and half length $l = (0.8m, 1.6m)$.

Training and Inference Details. Both PV-RCNN and PV-RCNN++ frameworks are trained from scratch in an end-to-end manner with ADAM optimizer, learning rate 0.01 and cosine annealing learning rate decay strategy. To train the proposal refinement stage, we randomly sample 128 proposals with 1:1 ratio for positive and negative proposals, where a proposal is considered as a positive sample if it has at least 0.55 3D IoU with the ground-truth boxes, otherwise it is treated as a negative sample.

During training, we adopt the widely used data augmentation strategies for 3D object detection, including the random scene flipping, global scaling with a random scaling factor sampled from $[0.95, 1.05]$, global rotation around Z axis with a random angle sampled from $[-\frac{\pi}{4}, \frac{\pi}{4}]$, and the ground-truth sampling augmentation [16] to randomly "paste" some new objects from other scenes to current training scene for simulating objects in various environments.

For the Waymo Open dataset, the detection range is set as $[-75.2, 75.2]m$ for both the X and Y axes, and $[-2, 4]m$ for the Z axis, while the voxel size is set as $(0.1m, 0.1m, 0.15m)$. For the KITTI dataset, the detection range is set as $[0, 70.4]m$ for X axis, $[-40, 40]m$ for Y axis and $[-3, 1]m$ for the Z axis, which is voxelized with voxel size $(0.05m, 0.05m, 0.1m)$ in each axis.

For the inference speed, our final PV-RCNN++ framework can achieve state-of-the-art performance with 10 FPS for $150m \times 150m$ detection range on the Waymo Open Dataset (three times faster than PV-RCNN framework), while also achieving state-of-the-art performance with 16 FPS for $70m \times 80m$ detection region on the KITTI dataset. Both of them are profiling on a single TITAN RTX GPU card.

Point Feature Extraction	RoI Pooling Module	LEVEL 1 (3D) mAP	LEVEL 1 (3D) mAPH	LEVEL 2 (3D) mAP	LEVEL 2 (3D) mAPH
UNet-decoder	RoI-aware Pooling [24]	71.82	71.29	64.33	63.82
UNet-decoder	RoI-grid Pooling	73.84	73.18	64.76	64.15
VSA	RoI-aware Pooling [24]	69.89	69.25	61.03	60.46
VSA	RoI-grid Pooling	74.06	73.38	64.99	64.38

TABLE 1
Effects of voxel set abstraction (VSA) and RoI-grid pooling modules. We adopt same UNet-decoder with [24]. All experiments are based on our PV-RCNN framework with an anchor-based head for proposal generation, and only the above two modules are changed during the ablation experiments.

VSA Input Feature				LEVEL 1 (3D)		LEVEL 2 (3D)	
$f_i^{(pv_1),(pv_2)}$	$f_i^{(pv_3),(pv_4)}$	$f_i^{(bev)}$	$f_i^{(raw)}$	mAP	mAPH	mAP	mAPH
		✓		71.55	70.94	64.04	63.46
			✓	71.97	71.36	64.44	63.86
	✓			72.15	71.54	64.66	64.07
	✓	✓		73.77	73.09	64.72	64.11
	✓	✓	✓	74.06	73.38	64.99	64.38
✓	✓	✓	✓	74.10	73.42	65.04	64.43

TABLE 2
Effects of different feature components for voxel set abstraction module. All experiments are based on our PV-RCNN framework with an anchor-based head for proposal generation.

Use PKW	LEVEL 1 (3D) mAP	LEVEL 1 (3D) mAPH	LEVEL 2 (3D) mAP	LEVEL 2 (3D) mAPH
✗	73.90	73.29	64.75	64.23
✓	74.06	73.38	64.99	64.38

TABLE 3
Effects of predicted keypoint weighting (PKW) module. The experiments are based on our PV-RCNN framework with an anchor-based head for proposal generation.

6.2 Ablation Studies for PV-RCNN Framework

We investigate the individual components of our proposed PV-RCNN/PV-RCNN++ frameworks with extensive ablation experiments. We conduct all experiments on the large Waymo Open Dataset [85] with detection range $150m \times 150m$ for more comprehensive evaluation. For efficiently conducting the ablation experiments, we generate a small representative training set by uniformly sampling 20% frames (about $32k$ frames) from the training set as in [25], and all results are evaluated on the full validation set (about $40k$ frames) with the official evaluation tool. All models are trained with 30 epochs and batch size 2 on each GPU.

6.2.1 The Component Analysis of PV-RCNN

In this section, all models are equipped with the anchor-based RPN head as in [16], [24], and all experiments are conducted on the vehicle category of Waymo Open Dataset [85].

Effects of Voxel-to-Keypoint Scene Encoding. In Sec. 4.1, we propose the voxel-to-keypoint scene encoding strategy to encode the global scene features to a small set of keypoints, which serves as a bridge between the backbone network and the proposal refinement network. As shown in the 2^{nd} and 4^{th} rows of Table 1, our proposed voxel-to-keypoint encoding strategy achieves comparable performance with the UNet-based decoder by summarizing the scene features to much less point-wise features than the UNet-based decoder. For instance, our voxel set abstraction module encodes the whole scene to around $4k$ keypoints for feeding into the RoI-grid pooling module, while the UNet-based decoder network needs to summarize the scene features to around $80k$ point-wise features, which validates the effectiveness of our proposed voxel-to-keypoint scene encoding strategy. We consider that it might benefit from the fact that the keypoint features are aggregated from multi-scale feature volumes and raw point clouds with large receptive fields, while also keeping the accurate point locations. Besides that, the feature dimension of UNet-based decoder is generally smaller than the feature dimensions of our keypoints since it is limited to its large memory consumption on large-scale point clouds, which may degrade its performance.

We also notice that our voxel set abstraction module achieves worse performance (the 1^{st} and 3^{rd} rows of Table 1) than the UNet-decoder when it is combined with RoI-aware pooling [24]. This is to be expected since RoI-aware pooling module generates lots of empty voxels in each proposal by taking only $4k$ keypoints, which may degrade the performance. In contrast, our voxel set abstraction module can be ideally combined with RoI-grid pooling module and they can benefit each other by taking a small number of keypoints as the intermediate connection.

Effects of Different Features for Voxel Set Abstraction. Our proposed voxel set abstraction module incorporates multiple feature components (see Sec. 4.1), and their effects are explored in Table 2. We can summarize the observations as follows: (i) The performance drops a lot if we only aggregate features from high level bird-view semantic features ($f_i^{(bev)}$) or accurate point locations ($f_i^{(raw)}$), since neither 2D-semantic-only nor point-only are enough for the proposal refinement. (ii) As shown in 5^{th} row of Table 2, $f_i^{(pv_3)}$ and $f_i^{(pv_4)}$ contain both 3D structure information and high level semantic features, which can improve the performance a lot by combining with the bird-view semantic features $f_i^{(bev)}$ and the raw point locations $f_i^{(raw)}$. (iii) The shallow semantic features $f_i^{(pv_1)}$ and $f_i^{(pv_2)}$ can slightly improve the performance and the best performance is achieved by taking all the feature components as the keypoint features.

Effects of Predicted Keypoint Weighting Module. We propose the predicted keypoint weighting (PKW) module in Sec. 4.1 to re-weight the point-wise features of keypoints with extra keypoint segmentation supervision. As shown in Table 3, the experiments show that the performance slightly drops after removing this module, which demonstrates that the predicted keypoint weighting module enables better multi-scale feature aggregation by focusing more on the foreground keypoints, since they are more important for the succeeding proposal refinement network.

Effects of RoI-grid pooling module. RoI-grid pooling module is proposed in Sec. 4.2 for aggregating RoI features from very sparse keypoints. Here we investigate the effects of RoI-grid pooling module by replacing it with the RoI-aware pooling [24] and keeping other modules consistent. As shown in Table 1, the experiments show that the performance drops significantly when replacing RoI-grid pooling module. It validates that our proposed RoI-grid pooling module can aggregate much richer contextual information to generate more discriminative RoI features.

Compared with the previous RoI-aware pooling module, our proposed RoI-grid pooling module [24] can generate

Keypoint Sampling Algorithm	Running Time	LEVEL 1 (3D) mAP	LEVEL 1 (3D) mAPH	LEVEL 2 (3D) mAP	LEVEL 2 (3D) mAPH	Coverage Rate under Different Radii 0.1m	0.2m	0.3m	0.4m	0.5m	Average Coverage Rate
FPS	133ms	74.06	73.38	64.99	64.38	-	-	-	-	-	-
PC-Filter + FPS	27ms	75.05	74.41	65.98	65.40	43.22	82.78	97.97	99.95	99.99	84.78
PC-Filter + Random Sampling	<1ms	69.85	69.23	60.97	60.42	51.44	77.06	87.32	92.16	94.74	80.54
PC-Filter + Voxelized-FPS-Voxel	17ms	74.12	73.46	65.06	64.47	6.52	52.00	89.93	98.89	99.92	69.45
PC-Filter + Voxelized-FPS-Point	17ms	74.38	73.71	65.36	64.76	34.14	75.84	96.28	99.77	99.99	81.20
PC-Filter + RandomParallel-FPS	2ms	73.94	73.28	64.90	64.31	38.12	64.47	81.72	91.82	96.90	74.61
PC-Filter + Sectorized-FPS	9ms	74.94	74.27	65.81	65.21	47.63	82.38	95.20	98.87	99.72	84.76

TABLE 4
Effects of different keypoint sampling algorithms. The running time is the average running time of keypoint sampling process on the validation set of the Waymo Open Dataset. The coverage rate is calculated by averaging the coverage rate of each scene on the validation set of the Waymo Open Dataset. "FPS" indicates the farthest point sampling and "PC-Filter" indicates our proposal-centric filtering strategy. All experiments are conducted by adding different keypoint sampling algorithms to our PV-RCNN framework with an anchor-based head.

denser grid-wise feature representation by supporting different overlapped ball areas among different grid points, while RoI-aware pooling module may generate lots of zeros due to the sparse inside points of RoIs. That means our proposed RoI-grid pooling module is especially effective for aggregating local features from very sparse point-wise features, such as in our PV-RCNN framework to aggregate features from a very small number of keypoints.

6.2.2 The Component Analysis of PV-RCNN++

In this section, the experiments in Table 4 adopt the same setting with the experiments in Sec. 6.2.1. Except for that, all other models are equipped with the center-based RPN head as in [84], and all experiments are jointly conducted on three categories (vehicle, pedestrian and cyclist) of Waymo Open Dataset [85], where the mAPH of LEVEL 2 difficulty is adopted as the evaluation metric as used by the official Waymo Open Dataset [85].

Effects of Proposal-Centric Filtering for Keypoint Sampling: In the 1^{st} and 2^{nd} rows of Table 4, we investigate the effectiveness of our proposed proposal-centric keypoint filtering (see Sec. 5.1), where we can see that compared with the strong baseline PV-RCNN, our proposal-centric keypoint filtering can further improve the detection performance by about 1.0 mAP/mAPH in both LEVEL 1 and LEVEL 2 difficulties. It validates our argument that our proposed proposal-centric keypoint sampling strategy can generate more representative keypoints by concentrating the small number of keypoints to the more informative neighboring regions of proposals. Moreover, improved by our proposal-centric keypoint filtering, our keypoint sampling algorithm is about five times (133ms vs. 27ms) faster than the baseline farthest point sampling algorithm by reducing the number of candidate keypoints.

Comparison of Different Strategies for Local Keypoint Sampling. In Sec. 5.1, we propose the sectorized farthest point sampling algorithm to further speed up the local keypoint sampling after the above proposal-centric filtering. Besides our proposed algorithm, we further explore four alternative strategies for accelerating the keypoint sampling strategy, which are as follows: (i) Random Sampling: the keypoints are randomly chosen from raw points. (ii) Voxelized-FPS-Voxel: the raw points are firstly voxelized to reduce the number of points (*i.e.,* voxels), then the farthest point sampling is applied to sample keypoints from voxels by taking the voxel centers. (iii) Voxelized-FPS-Point: unlike Voxelized-FPS-Voxel, here a raw point is randomly selected within the selected voxels as keypoints. (iv) RandomParallel-FPS: the raw points are randomly split into several groups, then farthest point sampling is utilized to these groups in parallel for faster keypoint sampling. As shown in Table 4, compared with the vanilla farthest point sampling (2^{nd} row) algorithm, the detection performances of all four alternative strategies drop a lot. In contrast, the performance of our proposed sectorized farthest point sampling algorithm is on par with the vanilla farthest point sampling algorithm while being three times (27ms vs. 9ms) faster than the vanilla farthest point sampling algorithm.

We argue that the uniformly distributed keypoints are important for the following proposal refinement, where a better keypoint distribution should cover more input points to benefit the scene feature aggregation. Hence, to better demonstrate the quality of different keypoint distributions in terms of statistics, we propose an evaluation metric, *coverage rate*, which is defined as the ratio of input points that are within the coverage region of any keypoints. Specifically, for a set of input points $\mathcal{P} = \{p_1, p_2, \cdots, p_m\}$ and a set of sampled keypoints $\mathcal{K} = \{p'_1, p'_2, ..., p'_n\}$, the coverage rate $\mathbf{C}$ can be formulated as follows:

$$\mathbf{C} = \frac{\sum_{i=1}^{m} \min\left(1.0, \quad \sum_{j=1}^{n} \mathbb{1}\left(||p_i - p'_j|| < R_c\right)\right)}{m}, \quad (17)$$

where R_c is a scalar that denotes the coverage radius of each keypoint, and $\mathbb{1}(\cdot)$ is the indicator function to indicate whether p_i is covered by p'_j.

As shown in Table 4, we evaluate the coverage rate of different keypoint sampling algorithms in terms of multiple coverage radii. Our proposed sectorized farthest point sampling achieves similar average coverage rate (84.76%) with the vanilla farthest point sampling algorithm (84.78%), which is much better than other sampling algorithms. The higher average coverage rate demonstrates that our proposed sectorized farthest point sampling can sample more uniformly distributed keypoints to better cover the input points, which is consistent with the qualitative visualization of different keypoint sampling strategies as in Fig. 6.

Besides that, the coverage rate (Table 4) and qualitative visualization (Fig. 6) also demonstrate some properties of different sampling algorithms: (i) The random sampling can achieve higher coverage rate with small radius (51.44% with 0.1m radius) since it can probably sample more keypoints on the clustered region, but it can not guarantee the coverage rate with large radius (94.74% with 0.5m radius) and may lose some important points such as the distant points. (ii) Compared with the Voxelized-FPS-Voxel, the Voxelized-FPS-Point achieves much better coverage rate with small

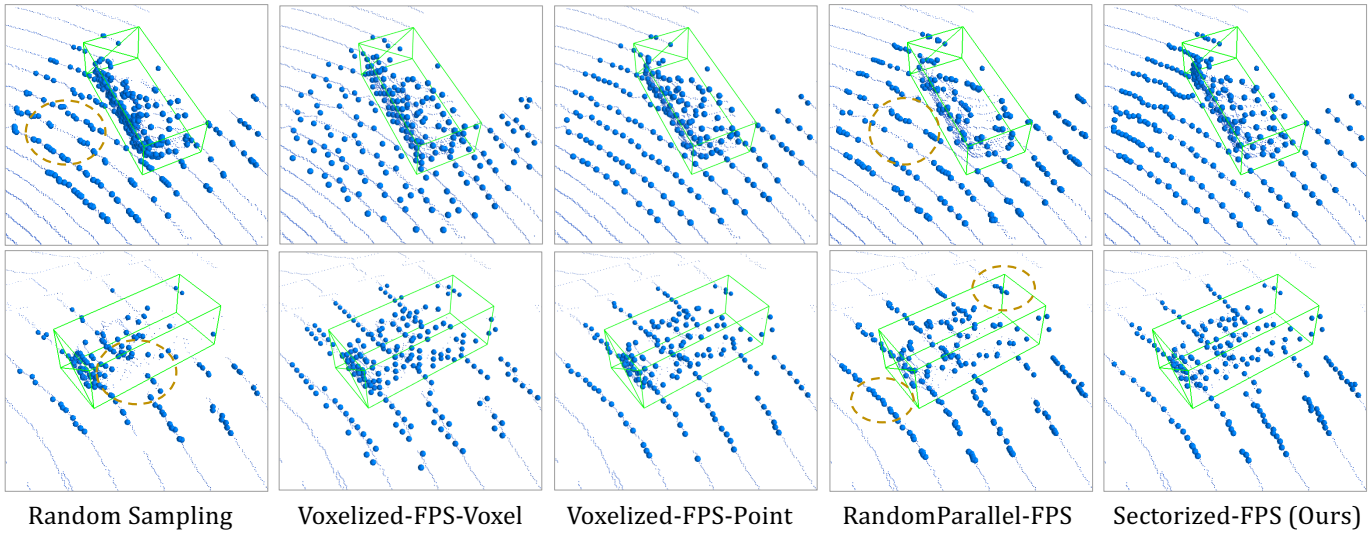

Fig. 6. Illustration of the keypoint distributions from different keypoint sampling strategies. Some dashed circles are utilized to highlight the missing parts and the clustered keypoints after using these keypoint sampling strategies. We can see that our proposed Sectorized-FPS generates better uniformly distributed keypoints that cover more input points to better encode the scene features for proposal refinement, while other strategies may miss some important regions or generate some clustered keypoints.

Setting	Keypoint Sampling	Point Feature Extraction	RoI Feature Aggregation	Vehicle		Pedestrian		Cyclist		Average		GFLOPS	Memory (MB)
				L1	L2	L1	L2	L1	L2	L1	L2		
PV-RCNN	FPS	VSA (SA)	RoI-grid pool (SA)	75.43	67.54	69.40	61.62	68.98	66.57	71.27	65.24	-0	10617
-	SPC-FPS	VSA (SA)	RoI-grid pool (SA)	76.44	68.03	70.52	62.43	69.84	67.29	72.27	65.92	-0	10617
-	SPC-FPS	VSA (VP)	RoI-grid pool (SA)	77.41	68.73	70.98	63.30	69.63	67.19	72.67	66.41	-1.712	9453
-	SPC-FPS	VSA (SA)	RoI-grid pool (VP)	77.12	68.43	70.82	63.15	70.11	67.66	72.68	66.41	-2.967	8565
PV-RCNN++	SPC-FPS	VSA (VP)	RoI-grid pool (VP)	77.32	68.62	71.36	63.74	70.71	68.26	73.13	66.87	-4.679	7583

TABLE 5
Effects of different components in our proposed PV-RCNN++ frameworks. All models are trained with 20% frames from the training set and are evaluated on the full validation set of the Waymo Open Dataset, and the evaluation metric is the mAPH in terms of LEVEL_1 (L1) and LEVEL_2 (L2) difficulties as used in [85]. "FPS" denotes farthest point sampling, "SPC-FPS" denotes our proposed sectorized proposal-centric keypoint sampling strategy, "VSA" denotes the voxel set abstraction module, "SA" denotes the set abstraction operation and "VP" denotes our proposed VectorPool aggregation. All models adopt the center-based head for proposal generation.

radii and also achieves better performance by taking raw points as keypoints. However, it is still about twice slower than our method and its performance is also lower than our method. (*iii*) The RandomParallel-FPS generates small clustered keypoints since the nearby raw points can be divided into different groups, and all of them can be sampled as keypoints from different groups.

In short, our sectorized farthest point sampling can generate uniformly distributed keypoints to better cover the input points, by splitting points into different groups based on the radial distribution of LiDAR points. Although there may still exist a very small number of clustered keypoints in the margins of different groups, the experiments show that they have negligible effect on the performance. We consider the reason may be that the clustered keypoints are mostly in the regions around the scene centers, where the objects are generally easier to detect since the raw points around scene centers are much denser than distant regions.

Effects of VectorPool Aggregation. In Sec. 5.2, to tackle the resource-consuming problem of set abstraction in our PV-RCNN framework, we propose the VectorPool aggregation module to effectively and efficiently summarize the structure-preserved local features from point clouds. As shown in Tabel 5, by adopting our VectorPool aggregation in both the voxel set abstraction module and the RoI-grid pooling module, our PV-RCNN++ framework consumes much less computations (*i.e.*, a reduction of 4.679 GFLOPS) and GPU memory (from 10.62GB to 7.58GB) than the original PV-RCNN framework, while the performance is also consistently increased from 65.92% to 66.87% in terms of average mAPH (LEVEL 2) of three categories. Note that the reduction of memory consumption / calculations can be more significant with larger batch size.

The significant reduction of memory consumption and calculations demonstrates the effectiveness of our VectorPool aggregation for feature learning from large-scale point clouds, which makes our PV-RCNN++ framework a more practical 3D detector to be used on resource-limited devices. Moreover, the final PV-RCNN++ framework also benefits from the structure-preserved spatial features from our VectorPool aggregation, which is critical for the following fine-grained proposal refinement.

We further analyze the effects of VectorPool aggregation by separately investigating the effects of channel reduction [87] (see Eq. 12). As shown in Table 6, our VectorPool aggregation is more effective in reducing the memory consumption no matter whether the channel reduction is incorporated (by comparing the 1^{st} / 3^{rd} rows or the 2^{nd} / 4^{th} rows), since the activations in our VectorPool aggregation modules consume much less memory than those in the set abstraction, by adopting a single local vector representation before multi-layer percetron networks. Meanwhile, Table 6

Strategy	Channel Reduction	Vehicle	Ped.	Cyclist	Average	GFLOPS	Memory (MB)
SA	✗	68.03	62.43	67.29	65.92	-0	10617
SA	✓	68.43	62.06	66.96	65.81	-3.467	9795
VP	✗	68.82	64.06	67.96	66.95	-1.988	8549
VP	✓	68.62	63.74	68.26	66.87	-4.679	7583

TABLE 6
Effects of VectorPool aggregation with and without channel reduction [87]. For the local feature extraction strategies, "SA" denotes the set abstraction operation and "VP" denotes our proposed VectorPool aggregation module. All experiments are based on our PV-RCNN++ framework with a center-based head for proposal generation, and only the local feature extraction modules are changed during the ablation experiments.

Aggregation Strategy of Local Voxels	Vehicle	Pedestrian	Cyclist	Average
Average Pooling	68.35	62.33	67.50	66.06
Random Selection	68.36	62.82	67.68	66.29
Interpolation	68.62	63.74	68.26	66.87

TABLE 7
Effects of the feature aggregation strategies in the feature aggregation process of local voxels of VectorPool aggregation. All experiments are based on our PV-RCNN++ framework with a center-based head for proposal generation.

Kernel Weights	Number of Dense Voxels	Vehicle	Pedestrian	Cyclist	Average
Share	$3 \times 3 \times 3$	68.17	63.28	67.36	66.27
Separate	$3 \times 3 \times 3$	68.62	63.74	68.26	66.87
Separate	$2 \times 2 \times 2$	68.21	62.88	67.44	66.18
Separate	$3 \times 3 \times 3$	68.62	63.74	68.26	66.87
Separate	$4 \times 4 \times 4$	68.74	63.99	67.98	66.90

TABLE 8
Effects of separate local kernel weights and the number of dense voxels in our proposed VectorPool aggregation module. All experiments are based on our PV-RCNN++ framework with a center-based head for proposal generation.

Number of Keypoints	Vehicle	Pedestrian	Cyclist	Average
8192	68.85	64.11	67.88	66.95
4096	68.62	63.74	68.26	66.87
2048	67.99	62.14	67.41	65.85
1024	66.67	59.21	65.07	63.65

TABLE 9
Effects of the number of keypoints for encoding the global scene. All experiments are based on our PV-RCNN++ framework with a center-based head for proposal generation.

also demonstrates that our proposed VectorPool aggregation can achieve better performance than set abstraction [19] in both two cases (with or without channel reduction).

Effects of Different Feature Aggregation Strategies for Local Voxels. As mentioned in Sec. 5.2, in addition to our adopted interpolation-based method, there are two alternative strategies (average pooling and random selection) for aggregating features of local voxels. We investigate the effects of these three strategies in Table 7, where we notice that the our interpolation based feature aggregation achieves much better performance than the other two strategies, especially for the small objects like pedestrian and cyclist. We consider that our strategy can generate more effective features by interpolating from three nearest neighbors (even beyond this local voxel), while both of the other two methods might generate lots of zero features on the empty local voxels, which may degrade the final performance.

Effects of Separate Local Kernel Weights in VectorPool Aggregation. We have demonstrated the fact in Eq. (15) that our proposed VectorPool aggregation generates position-sensitive features by encoding relative position features with separate local kernel weights. The 1^{st} and 2^{nd} rows of Table 8 show that the performance drops a bit if we remove the separate local kernel weights and adopt shared kernel weights for relative position encoding. It validates that the separate local kernel weights are better than previous shared-parameter MLP based local feature encoding, and it is important in our VectorPool aggregation operation.

Effects of Dense Voxel Numbers in VectorPool Aggregation. We investigate the number of dense voxels $n_x \times n_y \times n_z$ in VectorPool aggregation for voxel set abstraction module and RoI-grid pooling module, where we can see that VectorPool aggregation with $3 \times 3 \times 3$ and $4 \times 4 \times 4$ achieve similar performance while the performance of $2 \times 2 \times 2$ setting drops a lot. We consider that our interpolation-based VectorPool aggregation can generate effective voxel-wise features even with large dense voxels, hence the setting with $4 \times 4 \times 4$ achieves slightly better performance than the setting with $3 \times 3 \times 3$. However, since the setting with $4 \times 4 \times 4$ greatly improves the calculations and memory consumption, we finally choose the setting of $3 \times 3 \times 3$ dense voxel representation in both voxel set abstraction module (except the raw point layer) and RoI-grid pooling module of our PV-RCNN++ framework.

Effects of the Number of Keypoints. In Table 9, we investigate the effects of the number of keypoints for encoding the scene features. Table 9 shows that larger number of keypoints achieves better performance, and similar performance is observed when using more than 4,096 keypoints. Hence, to balance the performance and computation cost, we empirically choose to encode the whole scene to 4,096 keypoints for the Waymo dataset (2,048 keypoints for the KITTI dataset since it only needs to detect the frontal-view areas). The above experiments show that our method can effectively encode the whole scene to a small number of keypoints while keeping similar performance with a large number of keypoints, which demonstrates the effectiveness of the keypoint feature encoding strategy of our proposed PV-RCNN detection framework.

Effects of the Grid Size in RoI-grid Pooling. Table 10 shows the performance of adopting different RoI-grid sizes for RoI-grid pooling module. We can see that the performance increases along with the RoI-grid sizes from $3 \times 3 \times 3$ to $6 \times 6 \times 6$, and the settings with larger RoI-grid sizes than $6 \times 6 \times 6$ achieve similar performance. Hence we finally adopt RoI-grid size $6 \times 6 \times 6$ for the RoI-grid pooling module.

RoI-grid Size	Vehicle	Pedestrian	Cyclist	Average
$8 \times 8 \times 8$	68.88	63.74	67.84	66.82
$7 \times 7 \times 7$	68.76	63.81	68.00	66.85
$6 \times 6 \times 6$	68.62	63.74	68.26	66.87
$5 \times 5 \times 5$	68.28	63.54	67.69	66.50
$4 \times 4 \times 4$	68.21	63.58	67.56	66.45
$3 \times 3 \times 3$	67.33	62.93	67.22	65.83

TABLE 10
Effects of the grid size in RoI-grid pooling module. All experiments are based on our PV-RCNN++ framework with a center-based head for proposal generation.

Method	Reference	Veh. (LEVEL 1)		Veh. (LEVEL 2)		Ped. (LEVEL 1)		Ped. (LEVEL 2)		Cyc. (LEVEL 1)		Cyc. (LEVEL 2)	
		mAP	mAPH	mAP	mAPH	mAP	mAPH	mAP	mAPH	mAP	mAPH	mAP	mAPH
†SECOND [16]	Sensors 2018	72.27	71.69	63.85	63.33	68.70	58.18	60.72	51.31	60.62	59.28	58.34	57.05
StarNet [89]	NeurIPSw 2019	53.70	-	-	-	66.80	-	-	-	-	-	-	-
*PointPillar [15]	CVPR 2019	56.62	-	-	-	59.25	-	-	-	-	-	-	-
MVF [75]	CoRL 2019	62.93	-	-	-	65.33	-	-	-	-	-	-	-
Pillar-based [78]	ECCV 2020	69.80	-	-	-	72.51	-	-	-	-	-	-	-
†Part-A2-Net [24]	TPAMI 2020	77.05	76.51	68.47	67.97	75.24	66.87	66.18	58.62	68.60	67.36	66.13	64.93
‡CenterPoint-Voxel [84]	CVPR 2021	76.7	76.2	68.8	68.3	79.0	72.9	71.0	65.3	-	-	-	-
PV-RCNN (anchor)	-	77.51	76.89	68.98	68.41	75.01	65.65	66.04	57.61	67.81	66.35	65.39	63.98
PV-RCNN++ (anchor)	-	**79.19**	**78.64**	**70.45**	**69.95**	77.97	69.26	68.85	60.94	72.10	70.86	69.42	68.22
PV-RCNN (center)	-	78.00	77.50	69.43	68.98	79.21	73.03	70.42	64.72	71.46	70.27	68.95	67.79
PV-RCNN++ (center)	-	79.10	78.63	70.34	69.91	**80.62**	**74.62**	**71.86**	**66.30**	**73.49**	**72.38**	**70.70**	**69.62**
‡PV-RCNN++ (center)	-	**79.25**	**78.78**	**70.61**	**70.18**	**81.83**	**76.28**	**73.17**	**68.00**	**73.72**	**72.66**	**71.21**	**70.19**

TABLE 11
Performance comparison on the Waymo Open Dataset with 202 validation sequences. ∗: re-implemented by [75]. †: re-implemented by ourselves with their open source code. ‡: use the 3D voxel CNN with residual connections as the backbone network, while other settings of our PV-RCNN/PV-RCNN++ frameworks utilize a commonly used 3D voxel CNN without residual connections [16], [24]. We highlight the top two results for each evaluation metric.

Difficulty	Method	Vehicle		Pedestrian		Cyclist	
		mAP	mAPH	mAP	mAPH	mAP	mAPH
LEVEL 1	PV-RCNN	80.60	80.15	78.16	72.01	71.80	70.42
	PV-RCNN++	**81.62**	**81.20**	**80.41**	**74.99**	**71.93**	**70.76**
LEVEL 2	PV-RCNN	72.81	72.39	71.81	66.05	69.13	67.80
	PV-RCNN++	**73.86**	**73.47**	**74.12**	**69.00**	**69.28**	**68.15**

TABLE 12
Performance comparison on the test set of Waymo Open Dataset by submitting to the official test evaluation server. Both the PV-RCNN and PV-RCNN++ frameworks are equipped with the center-based RPN head for 3D proposal generation.

Method	Vehicle 3D mAP (IoU=0.7)				Vehicle BEV mAP (IoU=0.7)			
	Overall	0-30m	30-50m	50m-Inf	Overall	0-30m	30-50m	50m-Inf
LaserNet [90]	55.10	84.90	53.11	23.92	71.57	92.94	74.92	48.87
†SECOND [16]	72.26	90.66	70.03	47.55	89.18	96.84	88.39	78.37
*PointPillar [15]	56.62	81.01	51.75	27.94	75.57	92.10	74.06	55.47
MVF [75]	62.93	86.30	60.02	36.02	80.40	93.59	79.21	63.09
Pillar-based [78]	69.80	88.53	66.50	42.93	87.11	95.78	84.74	72.12
†Part-A2-Net [24]	77.05	92.35	75.91	54.06	90.81	97.52	90.35	80.12
PV-RCNN (center)	78.00	92.96	76.47	55.96	90.99	97.76	90.20	80.69
PV-RCNN++ (center)	**79.10**	**93.34**	**78.08**	**57.19**	**91.57**	**98.13**	**90.95**	**81.54**

TABLE 13
Performance comparison on the Waymo Open Dataset with 202 validation sequences for the vehicle detection. ∗: re-implemented by [75]. †: re-implemented by ourselves with their open source code.

Method	Pedestrian 3D mAP (IoU=0.7)				Pedestrian BEV mAP (IoU=0.7)			
	Overall	0-30m	30-50m	50m-Inf	Overall	0-30m	30-50m	50m-Inf
LaserNet [90]	63.40	73.47	61.55	42.69	70.01	78.24	69.47	52.68
†SECOND [16]	68.70	74.39	67.24	56.71	76.26	79.92	75.50	67.92
*PointPillar [15]	59.25	67.99	57.01	41.29	68.57	75.02	67.11	53.86
MVF [75]	65.33	72.51	63.35	50.62	74.38	80.01	72.98	62.51
Pillar-based [78]	72.51	79.34	72.14	56.77	78.53	83.56	78.70	65.86
†Part-A2-Net [24]	75.24	81.87	73.65	62.34	80.25	84.49	79.22	70.34
PV-PCNN (center)	79.21	83.33	78.53	69.36	84.23	87.20	83.87	76.74
PV-RCNN++ (center)	**80.62**	**84.88**	**79.65**	**70.64**	**85.43**	**88.48**	**84.88**	**77.80**

TABLE 14
Performance comparison on the Waymo Open Dataset with 202 validation sequences for the pedestrian detection. ∗: re-implemented by [75]. †: re-implemented by ourselves with their open source code.

Method	Cyclist 3D mAP (IoU=0.7)				Cyclist BEV mAP (IoU=0.7)			
	Overall	0-30m	30-50m	50m-Inf	Overall	0-30m	30-50m	50m-Inf
†SECOND [16]	60.61	73.33	55.51	41.98	63.55	74.58	59.31	46.75
†Part-A2-Net [24]	68.60	80.87	62.57	45.04	71.00	81.96	66.38	48.15
PV-RCNN (center)	71.46	81.10	65.65	52.58	74.00	82.07	69.43	**57.61**
PV-RCNN++ (center)	**73.49**	**83.65**	**68.90**	**51.41**	**75.94**	**84.06**	**72.89**	57.10

TABLE 15
Performance comparison on the Waymo Open Dataset with 202 validation sequences for the cyclist detection. †: re-implemented by ourselves with their open source code.

Moreover, from Table 10 and Table 5, we also notice that PV-RCNN++ with a much smaller RoI-grid size $4 \times 4 \times 4$ (66.45% in terms of mAPH@L2) can also outperform PV-RCNN with larger RoI-grid size $6 \times 6 \times 6$ (65.24% in terms of mAPH@L2), which further validates the effectiveness of our proposed sectorized proposal-centric sampling strategy and the VectorPool aggregation module.

6.3 Main Results of PV-RCNN Framework and Comparison with State-of-the-Art Methods

In this section, we demonstrate the main results of our proposed PV-RCNN/PV-RCNN++ frameworks, and make the comparison with state-of-the-art methods on both the large-scale Waymo Open Dataset [85] and the highly-competitive KITTI dataset [86].

6.3.1 3D Detection on the Waymo Open Dataset

To evaluate our methods on the Waymo Open Dataset [85], we adopt two settings of both PV-RCNN and PV-RCNN++ frameworks by equipping with different RPN heads for proposal generation, which are the anchor-based RPN head as in [24] and the center-based RPN head as in [84].

Comparison with State-of-the-Art Methods. As shown in Table 11, by equipping with the commonly used 3D voxel CNN as in [16], [24], our PV-RCNN++ with anchor-based RPN head achieves state-of-the-art performance on vehicle category, which surpasses the previous state-of-the-art work CenterPoint [84] with +1.65% mAPH of LEVEL 2 difficulty. Our PV-RCNN++ with center-based RPN head outperforms previous state-of-the-art works [24], [84] on all three categories with remarkable performance gains (+1.61% for vehicle, +1.0% for pedestrian and +4.69% for cyclist in terms of mAPH of LEVEL 2 difficulty). Moreover, by improving the backbone network with residual connections as used in CenterPoint [84], the performance of PV-RCNN++ with center-based RPN head can be further boosted, which outperforms CenterPoint significantly with a +1.88% mAPH gain for vehicle detection and +2.70% mAPH gain for pedestrian detection in terms of LEVEL 2 difficulty.

We also present the performance of center-based PV-RCNN/PV-RCNN++ at different distance ranges (see Table 13, Table 14 and Table 15) for reference, where we follow [75], [89] to evaluate the models on the LEVEL 1 difficulty

Method	Modality	3D Detection (Car)			BEV Detection (Car)			3D Detection (Ped.)			BEV Detection (Ped.)			3D Detection (Cyc.)			BEV Detection (Cyc.)		
		Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
MV3D [9]	R+L	74.97	63.63	54.00	86.62	78.93	69.80	-	-	-	-	-	-	-	-	-	-	-	-
ContFuse [13]	R+L	83.68	68.78	61.67	94.07	85.35	75.88	-	-	-	-	-	-	-	-	-	-	-	-
AVOD-FPN [11]	R+L	83.07	71.76	65.73	90.99	84.82	79.62	50.46	42.27	39.04	58.49	50.32	46.98	63.76	50.55	44.93	69.39	57.12	51.09
F-PointNet [20]	R+L	82.19	69.79	60.59	91.17	84.67	74.77	50.53	42.15	38.08	57.13	49.57	45.48	72.27	56.12	49.01	77.26	61.37	53.78
UberATG-MMF [17]	R+L	88.40	77.43	70.22	93.67	88.21	81.99	-	-	-	-	-	-	-	-	-	-	-	-
3D-CVF at SPA [72]	R+L	89.20	80.05	73.11	93.52	89.56	82.45	-	-	-	-	-	-	-	-	-	-	-	-
CLOCs [91]	R+L	88.94	80.67	**77.15**	93.05	89.80	86.57	-	-	-	-	-	-	-	-	-	-	-	-
SECOND [16]	L	83.34	72.55	65.82	89.39	83.77	78.59	-	-	-	-	-	-	71.33	52.08	45.83	76.50	56.05	49.45
PointPillars [15]	L	82.58	74.31	68.99	90.07	86.56	82.81	51.45	41.92	38.89	57.60	48.64	45.78	77.10	58.65	51.92	79.90	62.73	55.58
PointRCNN [21]	L	86.96	75.64	70.70	92.13	87.39	82.72	47.98	39.37	36.01	54.77	46.13	42.84	74.96	58.82	52.53	82.56	67.24	60.28
3D IoU Loss [92]	L	86.16	76.50	71.39	91.36	86.22	81.20	-	-	-	-	-	-	-	-	-	-	-	-
STD [23]	L	87.95	79.71	75.09	94.74	89.19	**86.42**	53.29	42.47	38.35	60.02	48.72	44.55	78.69	61.59	55.30	81.36	67.23	59.35
Part-A2-Net [24]	L	87.81	78.49	73.51	91.70	87.79	84.61	53.10	43.35	40.06	59.04	49.81	45.92	79.17	63.52	56.93	83.43	68.73	61.85
3DSSD [81]	L	88.36	79.57	74.55	92.66	89.02	85.86	54.64	44.27	40.23	60.54	49.94	45.73	**82.48**	64.10	56.90	**85.04**	67.62	61.14
Point-GNN [93]	L	88.33	79.47	72.29	93.11	89.17	83.90	51.92	43.77	40.14	55.36	47.07	44.61	78.60	63.48	57.08	81.17	67.28	59.67
PV-RCNN (Ours)	L	**90.25**	81.43	76.82	**94.98**	**90.65**	86.14	52.17	43.29	40.29	**59.86**	50.57	46.74	78.60	63.71	57.65	82.49	68.89	62.41
PV-RCNN++ (Ours)	L	90.14	**81.88**	**77.15**	92.66	88.74	85.97	**54.29**	**47.19**	**43.49**	59.73	**52.43**	**48.73**	82.22	**67.33**	**60.04**	84.60	**71.86**	**63.84**

TABLE 16
Performance comparison on the KITTI *test* set. The results are evaluated by the mean Average Precision with 40 recall positions by submitting to the official KITTI evaluation server. "L" indicates the LiDAR-only methods while "R+L" indicates that multi-modality methods with both RGB images and LiDAR sensors.

for comparing with previous methods.

Comparison between PV-RCNN and PV-RCNN++. Table 11 demonstrates that no matter which type of RPN head is adopted, our PV-RCNN++ framework consistently outperforms previous PV-RCNN framework on all three categories of all difficulty levels. Specifically, for the anchor-based setting, PV-RCNN++ surpasses PV-RCNN with a performance gain of +1.54% for vehicle, +3.33% for pedestrian and 4.24% for cyclist in terms of LEVEL 2 difficulty. By taking the center-based head, PV-RCNN++ also outperforms PV-RCNN with a +0.93% mAPH gain for vehicle, a +1.58% mAPH gain for pedestrian and a +1.83% mAPH gain for cyclist in terms of LEVEL 2 difficulty. Meanwhile, we also evaluate our PV-RCNN/PV-RCNN++ frameworks with center-based head on the test set by submitting to the official test server of Waymo Open Dataset [85]. As shown in Table 12, our PV-RCNN++ outperforms previous PV-RCNN with remarkable margins on all three categories.

The stable and consistent improvements on both the validation and test set of Waymo Open Dataset prove the effectiveness of our proposed sectorized proposal-centric sampling algorithm and the VectorPool aggregation module. Moreover, our PV-RCNN++ consumes much less calculations and GPU memory than PV-RCNN framework, while also increasing the processing speed from 3.3 FPS to 10 FPS for the 3D detection of $150m \times 150m$ such a large area, which further validates the efficiency and the effectiveness of our proposed PV-RCNN++.

6.3.2 3D detection on the KITTI dataset

To evaluate our PV-RCNN frameworks on the highly-competitive 3D detection learderboard of KITTI dataset, we train our models with 80% of *train* + *val* data and the remaining 20% data is used for validation. Both PV-RCNN and PV-RCNN++ frameworks are equipped with an anchor-based RPN head to generate 3D proposals. All results are evaluated by submitting to the official evaluation server.

Comparison with State-of-the-Art Methods. As shown in Table 16, our PV-RCNN and PV-RCNN++ outperform all published methods with remarkable margins on the most important moderate difficulty level. Specifically, compared with previous LiDAR-only state-of-the-art methods on the 3D detection benchmark of car category, our PV-RCNN++ increases the mAP by +1.78%, +2.17%, +2.06% on easy, moderate and hard difficulty levels, respectively. For the 3D detection and bird-view detection of both pedestrian and cyclist, our methods outperforms all previous methods with large margins on the moderate and hard difficulty level.

Compared with our preliminary work PV-RCNN, our PV-RCNN++ achieves better performance on the moderate and hard levels of 3D detection over all three categories, while also greatly reducing the GPU-memory consumption and increasing the running speed from 10 FPS to 16 FPS in the KITTI dataset. The significant improvements on both the performance and the efficiency manifest the effectiveness of our PV-RCNN++ framework.

7 Conclusion

In this paper, we present two novel frameworks, named PV-RCNN and PV-RCNN++, for accurate 3D object detection from point clouds. Our PV-RCNN framework adopts a novel voxel set abstraction module to deeply integrates both the multi-scale 3D voxel CNN features and the PointNet-based features to a small set of keypoints, and the learned discriminative keypoint features are then aggregated to the RoI-grid points through our proposed RoI-grid pooling module to capture much richer contextual information for proposal refinement. Our PV-RCNN++ further improves the PV-RCNN framework by efficiently generating more representative keypoints with our novel sectorized proposal-centric keypoint sampling strategy, and also by equipping with our proposed VectorPool aggregation module to learn structure-preserved local features in both the voxel set abstraction module and RoI-grid pooling module. Thus, our PV-RCNN++ finally achieves better performance with much faster running speed than the PV-RCNN.

Both of our proposed two PV-RCNN frameworks significantly outperform previous 3D detection methods and achieve new state-of-the-art performance on both the Waymo Open Dataset and the KITTI 3D detection benchmark, and extensive experiments are designed and conducted to deeply investigate the individual components of our proposed frameworks.

REFERENCES

[1] R. Girshick, "Fast r-cnn," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1440–1448.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *ECCV*. Springer, 2016, pp. 21–37.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[5] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 2117–2125.

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2961–2969.

[7] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[8] S. Song and J. Xiao, "Deep sliding shapes for amodal 3d object detection in rgb-d images," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016, pp. 808–816.

[9] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, July 2017.

[10] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[11] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander, "Joint 3d proposal generation and object detection from view aggregation," *IROS*, 2018.

[12] B. Yang, W. Luo, and R. Urtasun, "Pixor: Real-time 3d object detection from point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 7652–7660.

[13] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3d object detection," in *ECCV*, 2018.

[14] B. Yang, M. Liang, and R. Urtasun, "Hdnet: Exploiting hd maps for 3d object detection," in *CoRL*, 2018.

[15] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[16] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, 2018.

[17] M. Liang*, B. Yang*, Y. Chen, R. Hu, and R. Urtasun, "Multi-task multi-sensor fusion for 3d object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[18] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 652–660.

[19] 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*, 2017, pp. 5099–5108.

[20] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, June 2018.

[21] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 770–779.

[22] Z. Wang and K. Jia, "Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection," in *IROS*. IEEE, 2019.

[23] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "STD: sparse-to-dense 3d object detector for point cloud," *Proc. IEEE Int. Conf. Comput. Vis.*, 2019.

[24] S. Shi, Z. Wang, J. Shi, X. Wang, and H. 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.

[25] O. D. Team, "Openpcdet: An open-source toolbox for 3d object detection from point clouds," https://github.com/open-mmlab/OpenPCDet, 2020.

[26] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," in *Advances in neural information processing systems*, 2016.

[27] Z. Cai and N. Vasconcelos, "Cascade r-cnn: Delving into high quality object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[28] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang *et al.*, "Hybrid task cascade for instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[29] J. Pang, K. Chen, J. Shi, H. Feng, W. Ouyang, and D. Lin, "Libra r-cnn: Towards balanced learning for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[30] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[31] B. Li, Y. Liu, and X. Wang, "Gradient harmonized single-stage detector," in *AAAI*, 2019.

[32] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[33] X. Zhou, J. Zhuo, and P. Krahenbuhl, "Bottom-up object detection by grouping extreme and center points," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[34] Z. Yang, S. Liu, H. Hu, L. Wang, and S. Lin, "Reppoints: Point set representation for object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019.

[35] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019.

[36] L. Huang, Y. Yang, Y. Deng, and Y. Yu, "Densebox: Unifying landmark localization with end to end object detection," *arXiv preprint arXiv:1509.04874*, 2015.

[37] Z. Tian, C. Shen, H. Chen, and T. He, "Fcos: Fully convolutional one-stage object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019.

[38] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv preprint arXiv:1904.07850*, 2019.

[39] T. Kong, F. Sun, H. Liu, Y. Jiang, L. Li, and J. Shi, "Foveabox: Beyound anchor-based object detection," *IEEE Transactions on Image Processing*, 2020.

[40] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *ECCV*, 2020.

[41] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3d object detection for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.

[42] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3d bounding box estimation using deep learning and geometry," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[43] B. Li, W. Ouyang, L. Sheng, X. Zeng, and X. Wang, "Gs3d: An efficient 3d object detection framework for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[44] G. Brazil and X. Liu, "M3d-rpn: Monocular 3d region proposal network for object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019.

[45] M. Zeeshan Zia, M. Stark, and K. Schindler, "Are cars just 3d boxes?-jointly estimating the 3d shape of multiple objects," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2014.

[46] R. Mottaghi, Y. Xiang, and S. Savarese, "A coarse-to-fine model for 3d pose estimation and sub-category recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2015.

[47] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teuliere, and T. Chateau, "Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017.

[48] J. K. Murthy, G. S. Krishna, F. Chhaya, and K. M. Krishna, "Reconstructing vehicles from a single image: Shape priors for road scene understanding," in *International Conference on Robotics and Automation*, 2017.

[49] F. Manhardt, W. Kehl, and A. Gaidon, "Roi-10d: Monocular lifting of 2d detection to 6d pose and metric shape," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[50] P. Li, H. Zhao, P. Liu, and F. Cao, "Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving," in *ECCV*, 2020.

[51] P. Li, X. Chen, and S. Shen, "Stereo r-cnn based 3d object detection for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[52] R. Qian, D. Garg, Y. Wang, Y. You, S. Belongie, B. Hariharan, M. Campbell, K. Q. Weinberger, and W.-L. Chao, "End-to-end

pseudo-lidar for image-based 3d object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[53] Y. Chen, S. Liu, X. Shen, and J. Jia, "Dsgn: Deep stereo geometry network for 3d object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[54] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2019.

[55] Y. You, Y. Wang, W.-L. Chao, D. Garg, G. Pleiss, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving," *arXiv preprint arXiv:1906.06310*, 2019.

[56] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *ACM Transactions on Graphics (TOG)*, vol. 38, no. 5, p. 146, 2019.

[57] Q. Huang, W. Wang, and U. Neumann, "Recurrent slice networks for 3d segmentation of point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[58] H. Zhao, L. Jiang, C.-W. Fu, and J. Jia, "Pointweb: Enhancing local neighborhood features for point cloud processing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 5565–5573.

[59] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "Pointcnn: Convolution on x-transformed points," in *Advances in Neural Information Processing Systems*, 2018, pp. 820–830.

[60] H. Su, V. Jampani, D. Sun, S. Maji, E. Kalogerakis, M.-H. Yang, and J. Kautz, "Splatnet: Sparse lattice networks for point cloud processing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018, pp. 2530–2539.

[61] W. Wu, Z. Qi, and L. Fuxin, "Pointconv: Deep convolutional networks on 3d point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 9621–9630.

[62] M. Jaritz, J. Gu, and H. Su, "Multi-view pointnet for 3d scene understanding," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops*, 2019, pp. 0–0.

[63] L. Jiang, H. Zhao, S. Liu, X. Shen, C.-W. Fu, and J. Jia, "Hierarchical point-edge interaction network for point cloud semantic segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 10 433–10 441.

[64] 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 *Proc. IEEE Int. Conf. Comput. Vis.*, October 2019.

[65] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 3075–3084.

[66] Z. Liu, H. Hu, Y. Cao, Z. Zhang, and X. Tong, "A closer look at local aggregation operators in point cloud analysis," *arXiv preprint arXiv:2007.01294*, 2020.

[67] Q. Xu, X. Sun, C.-Y. Wu, P. Wang, and U. Neumann, "Grid-gcn for fast and scalable point cloud learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[68] B. Graham and L. van der Maaten, "Submanifold sparse convolutional networks," *arXiv preprint arXiv:1706.01307*, 2017.

[69] B. Graham, M. Engelcke, and L. van der Maaten, "3d semantic segmentation with submanifold sparse convolutional networks," *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2018.

[70] Z. Liu, H. Tang, Y. Lin, and S. Han, "Point-voxel cnn for efficient 3d deep learning," in *Advances in Neural Information Processing Systems*, 2019.

[71] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, "Pointpainting: Sequential fusion for 3d object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[72] J. H. Yoo, Y. Kim, J. S. Kim, and J. W. Choi, "3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object detection," in *ECCV*.

[73] T. Huang, Z. Liu, X. Chen, and X. Bai, "Epnet: Enhancing point features with image semantics for 3d object detection," in *ECCV*.

[74] M. Ye, S. Xu, and T. Cao, "Hvnet: Hybrid voxel network for lidar based 3d object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[75] Y. Zhou, P. Sun, Y. Zhang, D. Anguelov, J. Gao, T. Ouyang, J. Guo, J. Ngiam, and V. Vasudevan, "End-to-end multi-view fusion for 3d object detection in lidar point clouds," in *Conference on Robot Learning*, 2020, pp. 923–932.

[76] H. Kuang, B. Wang, J. An, M. Zhang, and Z. Zhang, "Voxel-fpn: Multi-scale voxel feature aggregation for 3d object detection from lidar point clouds," *Sensors*, vol. 20, no. 3, p. 704, 2020.

[77] B. Zhu, Z. Jiang, X. Zhou, Z. Li, and G. Yu, "Class-balanced grouping and sampling for point cloud 3d object detection," *arXiv preprint arXiv:1908.09492*, 2019.

[78] Y. Wang, A. Fathi, A. Kundu, D. Ross, C. Pantofaru, T. Funkhouser, and J. Solomon, "Pillar-based object detection for autonomous driving," *arXiv preprint arXiv:2007.10323*, 2020.

[79] Q. Chen, L. Sun, Z. Wang, K. Jia, and A. Yuille, "Object as hotspots: An anchor-free 3d object detection approach via firing of hotspots," 2019.

[80] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3d object detection in point clouds," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019.

[81] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3dssd: Point-based 3d single stage object detector," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[82] Y. Chen, S. Liu, X. Shen, and J. Jia, "Fast point r-cnn," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019.

[83] C. He, H. Zeng, J. Huang, X.-S. Hua, and L. Zhang, "Structure aware single-stage 3d object detection from point cloud," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

[84] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3d object detection and tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2021, pp. 11 784–11 793.

[85] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, June 2020.

[86] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2012.

[87] S. Sun, J. Pang, J. Shi, S. Yi, and W. Ouyang, "Fishnet: A versatile backbone for image, region, and pixel level prediction," in *Advances in neural information processing systems*, 2018, pp. 754–764.

[88] KITTI leader board of 3D object detection benchmark, http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d, Accessed on 2019-11-15.

[89] J. Ngiam, B. Caine, W. Han, B. Yang, Y. Chai, P. Sun, Y. Zhou, X. Yi, O. Alsharif, P. Nguyen *et al.*, "Starnet: Targeted computation for object detection in point clouds," *arXiv preprint arXiv:1908.11069*, 2019.

[90] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington, "Lasernet: An efficient probabilistic 3d object detector for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2019, pp. 12 677–12 686.

[91] S. Pang, D. Morris, and H. Radha, "Clocs: Camera-lidar object candidates fusion for 3d object detection," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020.

[92] D. Zhou, J. Fang, X. Song, C. Guan, J. Yin, Y. Dai, and R. Yang, "Iou loss for 2d/3d object detection," in *International Conference on 3D Vision (3DV)*. IEEE, 2019.

[93] W. Shi and R. Rajkumar, "Point-gnn: Graph neural network for 3d object detection in a point cloud," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2020, pp. 1711–1719.