Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection

Jiajun Deng[1], **Shaoshuai Shi**[2], **Peiwei Li**[1], **Wengang Zhou**[1,3], **Yanyong Zhang**[4], **Houqiang Li**[1,3]
[1] CAS Key Laboratory of GIPAS, EEIS Department, University of Science and Technology of China
[2] Multimedia Laboratory, The Chinese University of Hong Kong
[3] Institute of Artificial Intelligence, Hefei Comprehensive National Science Center
[4] Department of Computer Science, University of Science and Technology of China
{dengjj, lpw0622}@mail.ustc.edu.cn, {ssshi}@ee.cuhk.edu.hk {zhwg, yanyongz, lihq}@ustc.edu.cn

Abstract

Recent advances on 3D object detection heavily rely on how the 3D data are represented, i.e., voxel-based or point-based representation. Many existing high performance 3D detectors are point-based because this structure can better retain precise point positions. Nevertheless, point-level features lead to high computation overheads due to unordered storage. In contrast, the voxel-based structure is better suited for feature extraction but often yields lower accuracy because the input data are divided into grids. In this paper, we take a slightly different viewpoint — we find that precise positioning of raw points is not essential for high performance 3D object detection and that the coarse voxel granularity can also offer sufficient detection accuracy. Bearing this view in mind, we devise a simple but effective voxel-based framework, named Voxel R-CNN. By taking full advantage of voxel features in a two stage approach, our method achieves comparable detection accuracy with state-of-the-art point-based models, but at a fraction of the computation cost. Voxel R-CNN consists of a 3D backbone network, a 2D bird-eye-view (BEV) Region Proposal Network and a detect head. A voxel RoI pooling is devised to extract RoI features directly from voxel features for further refinement. Extensive experiments are conducted on the widely used KITTI Dataset and the more recent Waymo Open Dataset. Our results show that compared to existing voxel-based methods, Voxel R-CNN delivers a higher detection accuracy while maintaining a real-time frame processing rate, i.e., at a speed of 25 FPS on an NVIDIA RTX 2080 Ti GPU. The code is available at https://github.com/djiajunustc/Voxel-R-CNN.

1 Introduction

3D object detection using point clouds has received substantial attention in autonomous vehicles, robotics and augmented/virtual reality. Although the recent development of deep learning has led to the surge of object detection with 2D images (Ren et al. 2015; Liu et al. 2016; Redmon and Farhadi 2017; Szegedy et al. 2017), it is still non-trivial to apply these 2D methods to 3D point clouds, especially when dealing with the sparsity and unstructured property of point clouds. Besides, the applications usually demand high efficiency from detection systems, making it even harder to design a 3D detector due to the much larger 3D space.

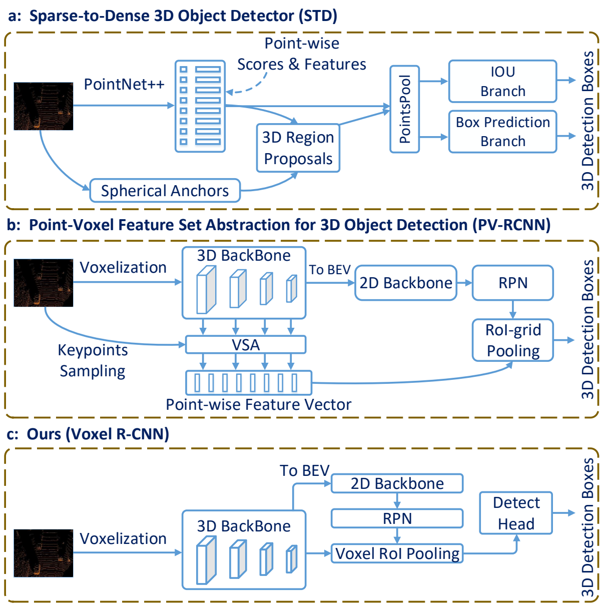


Figure 1: Comparison with state-of-the-art 3D object detection methods in two stage paradigm. Instead of aggregating RoI features from a set of point representations, our method directly extract RoI features from 3D voxel feature volumes.

Existing 3D detection methods can be broadly grouped into two categories, *i.e.*, voxel-based and point-based. The *voxel-based* methods (Zhou and Tuzel 2018; Yan, Mao, and Li 2018; Lang et al. 2019) divide point clouds into regular grids, which are more applicable for convolutional neural networks (CNNs) and more efficient for feature extraction due to superior memory locality. Nevertheless, the downside is that voxelization often causes loss of precise position information. Current state-of-the-art 3D detectors are mainly *point-based*, which take raw point clouds as input, and abstract a set of point representations with iterative sampling and grouping (Qi et al. 2017a,b). The advanced point-based methods (Yang et al. 2019; Shi, Wang, and Li 2019; Yang et al. 2020; Shi et al. 2020a) are top ranked on various benchmarks (Geiger, Lenz, and Urtasun 2012; Caesar et al. 2020; Sun et al. 2020). This has thus led to the popular viewpoint that the precise position information in raw point clouds is crucial for accurate object localization. Despite the superior detection accuracy, point-based methods are in general less efficient because it is more costly to search nearest neighbor with the point representation for point set abstraction.

As the detection algorithms become mature, we are ready to deploy these algorithms on realistic systems. Here, a new challenge arises: can we devise a method that is as accurate as advanced point-based methods and as fast as voxel-based methods? In this work, towards this objective, we take the voxel-based framework and try to boost its accuracy. We first argue that precise positioning of raw point clouds is nice but unnecessary. We observe that voxel-based methods generally perform object detection on the bird-eye-view (BEV) representation, even if the input data are in 3D voxels (Yan, Mao, and Li 2018). In contrast, point-based methods commonly rely on abstracted point representations to restore 3D structure context and make further refinement based on the point-wise features, as in Figure 1 (a) (b). By taking a close look at the underlying mechanisms, we find that the key disadvantage of existing voxel-based methods stems from the fact that they convert 3D feature volumes into BEV representations without ever restoring the 3D structure context.

Bearing this in mind, we propose to aggregate 3D structure context from 3D feature volumes. Specifically, we introduce a novel voxel-based detector, *i.e.*, Voxel R-CNN, to take full advantage of voxel features in a two stage pipeline (see Figure 1 (c)). Voxel R-CNN consists of three core modules: (1) a 3D backbone network, (2) a 2D backbone network followed by the Region Proposal Network (RPN), and (3) a detect head with a new voxel RoI pooling operation. The 3D backbone network gradually abstracts voxels into 3D feature volumes. Dense region proposals are generated by the 2D backbone and the RPN. Then, the RoI features are directly extracted from 3D feature volumes with voxel RoI pooling. In designing voxel RoI pooling, we take the neighbor-aware property (which facilitates better memory locality) to extract neighboring voxel features and devise a local feature aggregation module for further acceleration. Finally, the 3D RoI features are taken for further box refinement.

The main contribution of this work stems from the design of Voxel R-CNN, which strikes a careful balance between accuracy and efficiency. The encouraging experiment results of Voxel R-CNN also confirm our viewpoint: the precise positioning of raw points is not essential for high performance 3D object detection and coarser voxel granularity can also offer sufficient spatial context cues for this task. Please note that our Voxel R-CNN framework serves as a simple but effective baseline facilitating further investigation and downstream tasks.

2 Reflection on 3D Object Detection

In this section, we first revisit two representative baseline methods, *i.e.*, SECOND (Yan, Mao, and Li 2018) and PV-RCNN (Shi et al. 2020a), and then investigate the key factors of developing a high performance 3D object detector.

Methods	AP_{3D} (%)		
	Easy	Mod.	Hard
PV-RCNN (Shi et al. 2020a)	89.35	83.69	78.70
SECOND (Yan, Mao, and Li 2018)	88.61	78.62	77.22
SECOND +BEV detect head	89.50	79.26	78.45

Table 1: Performance comparison for adding BEV detect head on the top of SECOND. These results are evaluated on the KITTI *val* set with average precision calculated by 11 recall positions for car class.

Components	3D backbone	2D backbone	VSA	Detect head
Avg. time (ms)	21.1	9.4	49.4	19.1

Table 2: Running time comparison for each component in PV-RCNN. This result is calculated by the average over the 3,769 samples in the KITTI *val* set.

2.1 Revisiting

SECOND. SECOND (Yan, Mao, and Li 2018) is a voxel-based one stage object detector. It feeds the voxelized data to a 3D backbone network for feature extraction. The 3D feature volumes are then converted to BEV representations. Finally, a 2D backbone followed by a Region Proposal Network (RPN) is applied to perform detection.

PV-RCNN. PV-RCNN (Shi et al. 2020a) extends SECOND by adding a keypoints branch to preserve 3D structural information. Voxel Set Abstraction (VSA) is introduced to integrate multiscale 3D voxels features into keypoints. The features of each 3D region proposals are further extracted from the keypoints through RoI-grid pooling for box refinement.

2.2 Analysis

There exists a large gap between SECOND and PV-RCNN in terms of the detection performance (*i.e.*, accuracy and efficiency). These two methods differ in the following ways. Firstly, SECOND is a one stage method while PV-RCNN takes detect head for box refinement. Secondly, keypoints in PV-RCNN preserve the 3D structure information, while SECOND performs detection directly on BEV representations. To verify the influence of box refinement and 3D structure information on detection performance, we add a detect head on the top of the 2D backbone network in SECOND. Since the BEV boxes are not aligned with the axis, we exploit Rotated RoI Align for RoI feature extraction.

As illustrated in Table 1, directly adding a BEV detect head on top of BEV features leads to 0.6% AP improvement for KITTI car moderate data, but has still trailed the accuracy of PV-RCNN thus far. This verifies the effectiveness of box refinement, and also demonstrates that *the capacity of BEV representation is rather limited*. Typically, PV-RCNN integrates voxel features into sampled keypoints with Voxel Set Abstraction. The keypoints works as an intermediate feature representation to effectively preserve 3D structure information. However, as illustrated in Table 2, *the point-voxel interaction takes almost half of the overall running time*, which makes PV-RCNN much slower than SECOND.

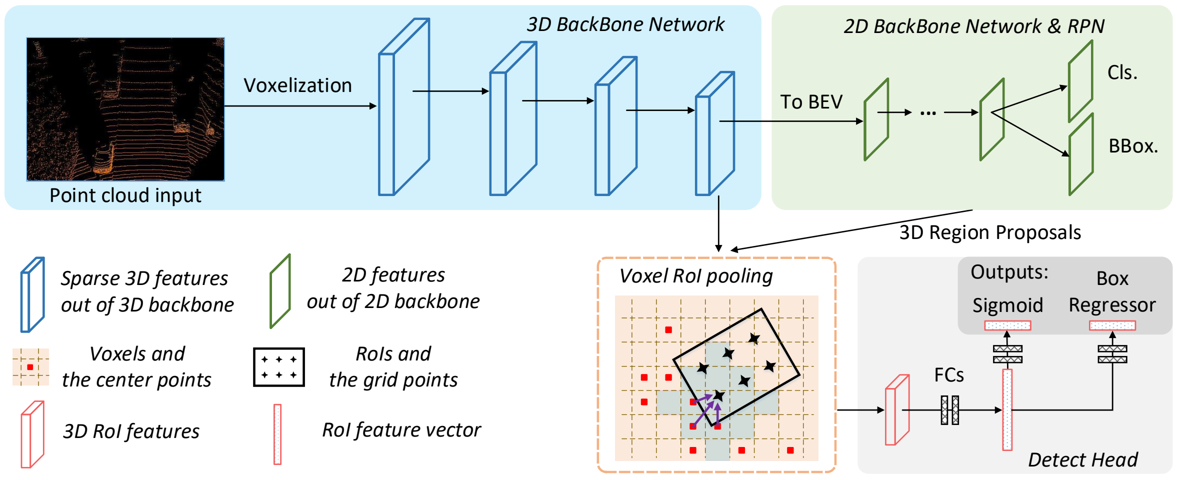


Figure 2: An overview of Voxel R-CNN for 3D object detection. The point clouds are first divided into regular voxels and fed into the 3D backbone network for feature extraction. Then, the 3D feture volumes are converted into BEV representation, on which we apply the 2D backbone and RPN for region proposal generation. Subsequently, voxel RoI pooling directly extracts RoI features from the 3D feature volumes. Finally the RoI features are exploited in the detect head for further box refinement.

Summary. In summary, by analyzing the limitations of bird-eye-view (BEV) feature representations in SECOND and the computation cost of each component in PV-RCNN, we observe the following: (a) the ***3D structure is of significant importance*** for 3D object detectors, since the BEV representation alone is insufficient to precisely predict bounding boxes in a 3D space; and (b) ***the point-voxel feature interaction is time-consuming*** and affects the detector's efficiency. These observations motivate us to directly leverage the 3D voxel tensors and develop a voxel-only 3D object detector.

3 Voxel R-CNN Design

In this section, we present the design of Voxel R-CNN, a voxel-based two stage framework for 3D object detection. As shown in Figure 2, Voxel R-CNN includes: (a) a 3D backbone network, (b) a 2D backbone network followed by the Region Proposal Network (RPN), and (c) a voxel RoI pooling and a detect subnet for box refinement. In Voxel R-CNN, we first divide the raw point cloud into regular voxels and utilize the 3D backbone network for feature extraction. We then convert the sparse 3D voxels into BEV representations, on which we apply the 2D backbone network and the RPN to generate 3D region proposals. Subsequently, we use Voxel RoI pooling to extract RoI features, which are fed into the detect subnet for box refinement. Below we discuss these modules in detail. Since our innovation mainly lies in voxel RoI pooling, we discuss it first.

3.1 Voxel RoI Pooling

To directly aggregate spatial context from 3D voxel feature volumes, we propose voxel RoI pooling.

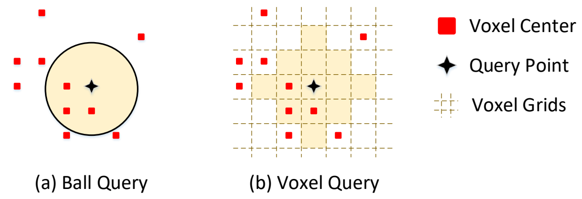


Figure 3: Illustration of ball query and our voxel query (performed in 3D space but shown in 2D space here)

Voxel Volumes as Points. We represent the sparse 3D volumes as a set of non-empty voxel center points $\{\boldsymbol{v}_i = (x_i, y_i, z_i)\}_{i=1}^N$ and their corresponding feature vectors $\{\phi_i\}_{i=1}^N$. Specifically, the 3D coordinates of voxel centers are calculated with the indices, voxel sizes and the point cloud boundaries.

Voxel Query. We propose a new operation, named voxel query, to find neighbor voxels from the 3D feature volumes. Compared to the unordered point clouds, the voxels are regularly arranged in the quantified space, lending itself to easy neighbor access. For example, the 26-neighbor voxels of a query voxel can be easily computed by adding a triplet of offsets $(\Delta_i, \Delta_j, \Delta_k), \Delta_i, \Delta_j, \Delta_k \in \{-1, 0, 1\}$ on the voxel indices (i, j, k). By taking advantage of this property, we devise voxel query to efficiently group voxels. The voxel query is illustrated in Figure 3. The query point is first quantified into a voxel, and then the neighbor voxels can be efficiently obtained by indices translation. We exploit Manhattan distance in voxel query and sample up to K voxels within a distance threshold. Specifically, the Manhattan distance $D(\alpha, \beta)$ between the voxel $\alpha = (i_\alpha, j_\alpha, k_\alpha)$ and

$\beta = (i_\beta, j_\beta, k_\beta)$ is computed as:

$$D_m(\alpha, \beta) = |i_\alpha - i_\beta| + |j_\alpha - j_\beta| + |k_\alpha - k_\beta|. \quad (1)$$

Suppose there are N non-empty voxels in the 3D feature volumes and we utilize ball query to find neighboring voxels to a given query point, the time complexity is $O(N)$. Nevertheless, the time complexity of conducting voxel query is only $O(K)$, where K is number of neighbors. The neighbor-aware property makes it more efficient to group neighbor voxel features with our voxel query than to group neighbor point features with ball query (Qi et al. 2017b).

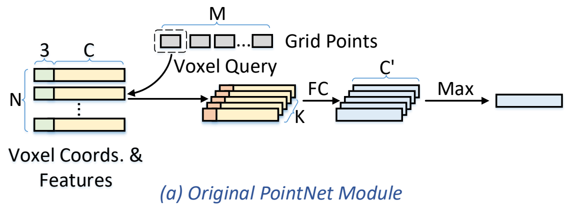
Voxel RoI Pooling Layer. We design the voxel RoI pooling layer as follows. It starts by dividing a region proposal into $G \times G \times G$ regular sub-voxels. The center point is taken as the grid point of the corresponding sub-voxel. Since 3D feature volumes are extremely sparse (non-empty voxels account for $< 3\%$ spaces), we cannot directly utilize max pooling over features of each sub-voxel as in (Girshick 2015). Instead, we integrate features from neighboring voxels into the grid points for feature extraction. Specifically, given a grid point $\boldsymbol{g}_i$, we first exploit voxel query to group a set of neighboring voxels $\Gamma_i = \{\boldsymbol{v}_i^1, \boldsymbol{v}_i^2, \cdots, \boldsymbol{v}_i^K\}$. Then, we aggregate the neighboring voxel features with a PointNet module (Qi et al. 2017a) as:

$$\boldsymbol{\eta}_i = \max_{k=1,2,\cdots,K} \{\Psi([\boldsymbol{v}_i^k - \boldsymbol{g}_i; \boldsymbol{\phi}_i^k])\}, \quad (2)$$

where $\boldsymbol{v}_i - \boldsymbol{g}_i$ represents the relative coordinates, $\boldsymbol{\phi}_i^k$ is the voxel feature of $\boldsymbol{v}_i^k$, and $\Psi(\cdot)$ indicates an MLP. The max pooling operation $\max(\cdot)$ is performed along the channels to obtain the aggregated feature vector $\boldsymbol{\eta}_i$. Particularly, we exploit Voxel RoI pooling to extract voxel features from the 3D feature volumes out of the last two stages in the 3D backbone network. And for each stage, two Manhattan distance thresholds are set to group voxels with multiple scales. Then, we concatenate the aggregated features pooled from different stages and scales to obtain the RoI features.

Accelerated Local Aggregation. Even with our proposed voxel query, the local aggregation operation (*i.e.*, PointNet module) in Voxel RoI pooling still involves large computation complexity. As shown in Figure 4 (a), there are totally M grid points ($M = r \times G^3$, where r is the number of RoI, and G is the grid size), and K voxels are grouped for each grid point. The dimension of grouped feature vectors is $C + 3$, including the C-dim voxel features and 3-dim relative coordinates. The grouped voxels occupy a lot of memories and lead to large computation FLOPs ($O(M \times K \times (C+3) \times C')$) when applying the FC layer.

As motivated by (Liu et al. 2020a; Hu et al. 2020), we additionally introduce an accelerated PointNet Module to further reduce the computation complexity of Voxel Query. Typically, as shown in Figure 4 (b), the voxel features and relative coordinates are decomposed into two streams. Given the FC layer with weight $W \in \mathbb{R}^{C',C+3}$, we divide it into $W_{\mathrm{F}} \in \mathbb{R}^{C',C}$ and $W_{\mathrm{C}} \in \mathbb{R}^{C',3}$. Since the voxel features are independent of the grid points, we apply a FC layer with W_{F} on the voxel features before performing voxel query. Then,



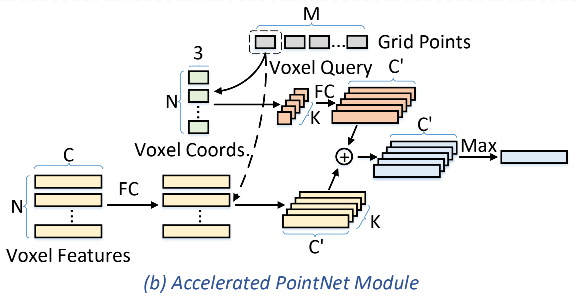


Figure 4: Illustration of different schemes to aggregate voxel features: (a) original PointNet module; (b) accelerated PointNet module. We only pick one grid point as an example, and the same operations are applied on all the other grid points.

after voxel query, we only multiply the grouped relative coordinates by W_{C} to obtain the relative position features and add them to the grouped voxel features. The FLOPs of our accelerated PointNet module are $O(N \times C \times C' + M \times K \times 3 \times C')$. Since the number of grouped voxels $(M \times K)$ is an order of magnitude higher than N, the accelerated PointNet module is more efficient than the original one.

3.2 Backbone and Region Proposal Networks

We follow the similar design of (Yan, Mao, and Li 2018; He et al. 2020; Shi et al. 2020a) to build our backbone networks. The 3D backbone network gradually converts the voxelized inputs into feature volumes. Then, the output tensors are stacked along the Z axis to produce BEV feature maps. The 2D backbone network consists of two components: a top-down feature extraction sub-network with two blocks of standard 3×3 convolution layers and a mutli-scale feature fusion sub-network that upsamples and concatenates the top-down features. Finally, the output of the 2D backbone network is convolved with two sibling 1×1 convolutional layers to generate 3D region proposals. We will detail the architecture of our backbone networks in Section 4.2.

3.3 Detect Head

The detect head takes RoI features as input for box refinement. Specifically, a shared 2-layer MLP first transform RoI features into feature vectors. Then, the flattened features are injected into two sibling branches: one for bounding box regression and the other for confidence prediction. As motivated by (Jiang et al. 2018; Li et al. 2019; Shi et al. 2020a,b), the box regression branch predicts the residue from 3D region proposals to the ground truth boxes, and the confidence branch predicts the IoU-related confidence score.

3.4 Training Objectives

Losses of RPN. We follow (Yan, Mao, and Li 2018; Lang et al. 2019) to devise the losses of the RPN as a combination of classification loss and box regression loss, as:

$$\mathcal{L}_{\text{RPN}} = \frac{1}{N_{\text{fg}}}[\sum_i \mathcal{L}_{\text{cls}}(p_i^a, c_i^*) + \mathbb{1}(c_i^* \geq 1)\sum_i \mathcal{L}_{\text{reg}}(\delta_i^a, t_i^*)], \tag{3}$$

where N_{fg} represents the number of foreground anchors, p_i^a and δ_i^a are the outputs of classification and box regression branches, c_i^* and t_i^* are the classification label and regression targets respectively. $\mathbb{1}(c_i^* \geq 1)$ indicates regression loss in only calculated with foreground anchors. Here we utilize Focal Loss (Lin et al. 2017) for classification and Huber Loss for box regression.

Losses of Detect Head. The target assigned to the confidence branch is an IoU related value, as:

$$l_i^*(\text{IoU}_i) = \begin{cases} 0 & \text{IoU}_i < \theta_L, \\ \frac{\text{IoU}_i - \theta_L}{\theta_H - \theta_L} & \theta_L \leq \text{IoU}_i < \theta_H, \\ 1 & \text{IoU}_i > \theta_H, \end{cases} \tag{4}$$

where IoU_i is the IoU between the i-th proposal and the corresponding ground truth box, θ_H and θ_L are foreground and background IoU thresholds. Binary Cross Entropy Loss is exploited here for confidence prediction. The box regression branch also uses Huber Loss as in the RPN. The losses of our detect head are computed as:

$$\begin{aligned}\mathcal{L}_{\text{head}} =& \frac{1}{N_s}[\sum_i \mathcal{L}_{\text{cls}}(p_i, l_i^*(\text{IoU}_i)) \\ &+ \mathbb{1}(\text{IoU}_i \geq \theta_{reg})\sum_i \mathcal{L}_{\text{reg}}(\delta_i, t_i^*)],\end{aligned} \tag{5}$$

where N_s is the number of sampled region proposals at the training stage, and $\mathbb{1}(\text{IoU}_i \geq \theta_{reg})$ indicates that only region proposals with IoU $> \theta_{\text{reg}}$ contribute to the regression loss.

4 Experiments

4.1 Datasets

KITTI. The *KITTI Dataset* (Geiger, Lenz, and Urtasun 2012) contains 7481 training samples and 7518 testing samples in autonomous driving scenes. As a common practice, the training data are divided into a *train* set with 3712 samples and a *val* set with 3769 samples. The performance on both the *val* set and online test leaderboard are reported for comparison. When performing experimental studies on the *val* set, we use the *train* set for training. For test server submission, we randomly select 80% samples from the training point clouds for training, and use the remaining 20% samples for validation as in (Shi et al. 2020a).

Waymo Open Dataset. The *Waymo Open Dataset* (Sun et al. 2020) is the largest public dataset for autonomous driving to date. There are totally $1,000$ sequences in this dataset, including 798 sequences ($\sim$158k point clouds samples) in training set and 202 sequences ($\sim$40k point clouds samples) in validation set. Different from KITTI that only provides annotations in camera FOV, the *Waymo Open Dataset* provides annotations for objects in the full 360°.

Method	FPS (Hz)	AP$_{\text{3D}}$ (%) Easy	Mod.	Hard
RGB+LiDAR				
MV3D (Chen et al. 2017)	-	74.97	63.63	54.00
AVOD-FPN (Ku et al. 2018)	10.0	83.07	71.76	65.73
F-PointNet (Qi et al. 2018)	5.9	82.19	69.79	60.59
PointSIFT+SENet (Zhao et al. 2019)	-	85.99	72.72	64.58
UberATG-MMF (Liang et al. 2019)	-	88.40	77.43	70.22
LiDAR-only				
Point-based:				
PointRCNN (Shi, Wang, and Li 2019)	10.0	86.96	75.64	70.70
STD (Yang et al. 2019)	12.5	87.95	79.71	75.09
Patches (Lehner et al. 2019)	-	88.67	77.20	71.82
3DSSD (Yang et al. 2020)	26.3	88.36	79.57	74.55
PV-RCNN (Shi et al. 2020a)	8.9	90.25	81.43	76.82
Voxel-based:				
VoxelNet (Zhou and Tuzel 2018)	-	77.47	65.11	57.73
SECOND (Yan, Mao, and Li 2018)	30.4	83.34	72.55	65.82
PointPillars (Lang et al. 2019)	42.4	82.58	74.31	68.99
Part-A^2 (Shi et al. 2020b)	-	87.81	78.49	73.51
TANet (Liu et al. 2020b)	28.7	85.94	75.76	68.32
HVNet (Ye, Xu, and Cao 2020)	31.0	87.21	77.58	71.79
SA-SSD (He et al. 2020)	25.0	88.75	79.79	74.16
Voxel R-CNN (ours)	25.2	**90.90**	**81.62**	**77.06**

Table 3: Performance comparison on the KITTI *test* set with AP calculated by recall 40 positions for car class

4.2 Implementation Details

Voxelization. The raw point clouds are divided into regular voxels before taken as the input of our Voxel R-CNN. Since the KITTI Dataset only provides the annotations of object in FOV, we clip the range of point clouds into $[0, 70.4]m$ for the X axis, $[-40, 40]m$ for the Y axis and $[-3, 1]m$ for Z axis. The input voxel size is set as $(0.05m, 0.05m, 0.1m)$. For the Waymo Open Dataset, the range of point clouds is clipped into $[-75.2, 75.2]m$ for X and Y axes and $[-2, 4]m$ for the Z axis. The input voxel size is set as $(0.1m, 0.1m, 0.15m)$.

Network Architecture. The architecture of 3D backbone and 2D backbone follows the design in (Yan, Mao, and Li 2018; Shi et al. 2020a). There are four stages in the 3D backbone with filter numbers of $16, 32, 48, 64$ respectively. There are two blocks in the 2D backbone network, the first block keeps the same resolution along X and Y axes as the output of 3D backbone network, while the second block is half the resolution of the first one. The numbers of convolutional layers in these two blocks are both set as 5. And the feature dimensions of these blocks are $(64, 128)$ for KITTI Dataset and $(128, 256)$ for Waymo Open Dataset. Each region proposal is divided into a $6 \times 6 \times 6$ grid for Voxel RoI pooling. The Manhattan distance thresholds are set as 2 and 4 for multi-scale voxel query. The output channels of the local feature aggregation module are 32 for KITTI Dataset and 64 for Waymo Open Dataset.

Training. The whole architecture of our Voxel R-CNN is end-to-end optimized with the Adam optimizer. For KITTI Dataset, the network is trained for 80 epochs with the batch size 16. For Waymo Open Dataset, the network is trained for 30 epochs with the batch size 32. The learning rate is

IoU Thresh.	AP_{3D} (%) Easy	Moderate	Hard	AP_{BEV} (%) Easy	Moderate	Hard
0.7	92.38	85.29	82.86	95.52	91.25	88.99

Table 4: Performance of Voxel R-CNN on the KITTI *val* set with AP calculated by 40 recall positions for car class

Method	AP_{3D} (%) Easy	Mod.	Hard
Point-based:			
PointRCNN (Shi, Wang, and Li 2019)	88.88	78.63	77.38
STD (Yang et al. 2019)	89.70	79.80	**79.30**
3DSSD (Yang et al. 2020)	89.71	79.45	78.67
PV-RCNN (Shi et al. 2020a)	89.35	83.69	78.70
Voxel-based:			
VoxelNet (Zhou and Tuzel 2018)	81.97	65.46	62.85
SECOND (Yan, Mao, and Li 2018)	88.61	78.62	77.22
PointPillars (Lang et al. 2019)	86.62	76.06	68.91
Part-A^2 (Shi et al. 2020b)	89.47	79.47	78.54
TANet (Liu et al. 2020b)	87.52	76.64	73.86
SA-SSD (He et al. 2020)	**90.15**	79.91	78.78
Voxel R-CNN (ours)	89.41	**84.52**	78.93

Table 5: Performance comparison on the KITTI *val* set with AP calculated by 11 recall positions for car class

initialized as 0.01 for both datasets and updated by cosine annealing strategy. In the detect head, the foreground IoU threshold θ_H is set as 0.75, background IoU threshold θ_L is set as 0.25, and the box regression IoU threshold θ_{reg} is set as 0.55. We randomly sample 128 RoIs as the training samples of detect head. Within the sampled RoIs, half of them are positive samples that have IoU $> \theta_{reg}$ with the corresponding ground truth boxes. We conduct data augmentation at the training stage following strategies in (Lang et al. 2019; Shi et al. 2020a; Yang et al. 2020; Ye, Xu, and Cao 2020). Please refer to OpenPCDet [1] for more detailed configurations since we conduct all experiments with this toolbox.

Inference. At the inference stage, we first perform non-maximum suppression (NMS) in the RPN with IoU threshold 0.7 and keep the top-100 region proposals as the input of detect head. Then, after refinement, NMS is applied again with IoU threshold 0.1 to remove the redundant predictions.

4.3 Results on KITTI Dataset

We evaluate our Voxel R-CNN on KITTI Dataset following the common protocol to report the average precision (AP) of class Car with the 0.7 (IoU) threshold. We report our performance on both *val* set and *test* set for comparison and analysis. The performance on *val* set is calculated with the AP setting of recall 11 positions. And the results evaluated by the test server utilize AP setting of recall 40 positions[2] .

Comparison with State-of-the-art Methods. We compare our Voxel R-CNN with several state-of-the-art methods on the KITTI *test* set by submitting our results to the online test server. As shown in the Table 3, our Voxel R-CNN makes the best balance between the accuracy and efficiency among all the methods. By taking full advantage of voxel-based representation, Voxel R-CNN achieves 81.62% average precision (AP) on the moderate level of class Car, with the real time processing frame rate (25.2 FPS). Particularly, Voxel R-CNN achieves comparable accuracy with the strongest competitor, *i.e.*, PV-RCNN (Shi et al. 2020a), with only about $1/3$ running time. This verifies the spatial context in the voxel representation is almost sufficient for 3D object detection, and the voxel representation is more efficient for feature extraction. Besides, Voxel R-CNN outperforms all of the existing voxel-based models by a large margin, *i.e.*, 2.15%, 1.83%, 2.90% absolute improvements for easy, moderate and hard level over the SA-SSD(He et al. 2020).

The AP for 3D object detection and BEV object detection of our Voxel R-CNN on KITTI Dataset is presented in Table 4. The results in this table are calculated by recall 40 positions with the (IoU) threshold 0.7. In addition, we report the performance on the KITTI *val* set with AP calculated by recall 11 positions for further comparison. As shown in Table 5, our Voxel R-CNN achieves the best performance on moderate and hard level on the *val* set.

Overall, the results on both *val* set and *test* set consistently demonstrate that our proposed Voxel R-CNN achieves the state-of-the-art average precision on 3D object detection and keeps the high efficiency of voxel-based models.

Method	Overall	0-30m	30-50m	50m-Inf
LEVEL_1 3D mAP (IoU=0.7):				
PointPillar (Lang et al. 2019)	56.62	81.01	51.75	27.94
MVF (Zhou et al. 2020)	62.93	86.30	60.02	36.02
Pillar-OD (Wang et al. 2020)	69.80	88.53	66.50	42.93
AFDet+PointPillars (Ge et al. 2020)	63.69	87.38	62.19	29.27
PV-RCNN (Shi et al. 2020a)	70.30	91.92	69.21	42.17
Voxel R-CNN (ours)	**75.59**	**92.49**	**74.09**	**53.15**
LEVEL_1 BEV mAP (IoU=0.7):				
PointPillar (Lang et al. 2019)	75.57	92.1	74.06	55.47
MVF (Zhou et al. 2020)	80.40	93.59	79.21	63.09
Pillar-OD (Wang et al. 2020)	87.11	95.78	84.87	72.12
PV-RCNN (Shi et al. 2020a)	82.96	97.35	82.99	64.97
Voxel R-CNN (ours)	**88.19**	**97.62**	**87.34**	**77.70**
LEVEL_2 3D mAP (IoU=0.7):				
PV-RCNN (Shi et al. 2020a)	65.36	91.58	65.13	36.46
Voxel R-CNN (ours)	**66.59**	**91.74**	**67.89**	**40.80**
LEVEL_2 BEV mAP (IoU=0.7):				
PV-RCNN (Shi et al. 2020a)	77.45	94.64	80.39	55.39
Voxel R-CNN (ours)	**81.07**	**96.99**	**81.37**	**63.26**

Table 6: Performance comparison on the Waymo Open Dataset with 202 validation sequences (~40k samples) for the vehicle detection

4.4 Results on Waymo Open Dataset

We also conduct experiments on the larger Waymo Open Dataset to further validate the effectiveness of our proposed Voxel R-CNN. The objects on the Waymo Open Dataset are split into two levels based on the number of points of a single object, where the LEVEL_1 objects have more than 5 points while the LEVEL_2 objects have 1~5 points.

[1]https://github.com/open-mmlab/OpenPCDet

[2]The setting of AP calculation is modified from recall 11 positions to recall 40 positions on 08.10.2019. We exploits the recall 11 setting on val set for fair comparison with previous methods.

Methods	D.H.	V.Q.	A.P.	Moderate AP_{3D} (%)	FPS (Hz)
(a)				77.99	40.8
(b)	✓			84.21	17.4
(c)	✓	✓		84.33	19.1
(d)	✓		✓	84.27	21.4
(e)	✓	✓	✓	84.52	25.2

Table 7: Performance of proposed method with different configurations on KITTI *val* set. "D.H.", "V.Q." and "A.P." stand for detect head, voxel query and accelerated PointNet module respectively. The results are evaluated with the average precision calculated by 11 recall positions for car class.

Comparison with State-of-the-art Methods. We evaluate our Voxel R-CNN on both LEVEL_1 and LEVEL_2 objects and compare with several top-performing methods on the Waymo Open Dataset. Table 6 shows that our method surpasses all previous methods with remarkable margins on all ranges of both LEVEL_1 and LEVEL_2. Specifically, with the commonly used LEVEL_1 3D mAP evaluation metric, our Voxel R-CNN achieves new state-of-the-art performance with 75.59% mAP, and outperforms previous state-of-the-art method PV-RCNN by 5.29% mAP. Especially, our method outperforms PV-RCNN with +10.98% mAP in the range of 50m-Inf, the significant gains on the far away area demonstrate the effectiveness of our proposed Voxel R-CNN for detecting the objects with very sparse points.

4.5 Ablation Study

Table 7 details how each proposed module influences the accuracy and efficiency of our Voxel R-CNN. The results are evaluated with AP of moderate level for car class.

Method (a) is the one stage baseline that performs detection on BEV features. It runs at 40.8 FPS, but with unsatisfactory AP, which indicates *BEV representation alone is insufficient to precisely detect objects in a 3D space*.

Method (b) extends *(a)* with a detect head for box refinement, which leads to a boost of 4.22% moderate AP. This verifies *the spatial context from 3D voxels provides sufficient cues for precise object detection*. However, *(b)* applies time-consuming ball query and original PointNet modules to extract RoI features, leading to a decrease of 23.4 FPS.

Method (c) replaces ball query with our voxel query, which makes 1.7 FPS improvement by *taking advantage of the neighbor-aware property of voxel-based representations*.

Method (d) uses our accelerated PointNet module to aggregate voxel features, boosting the FPS from 17.4 to 21.4.

Method (e) is the proposed Voxel R-CNN. By extending the one stage baseline with a detect head, taking voxel query for voxel grouping, and utilizing the accelerated PointNet module for local feature aggregation, Voxel R-CNN *achieves the state-of-the-art accuracy* for 3D object detection and *maintain the high efficiency* of voxel-based models.

5 Related Work

We broadly categorize the methods for 3D object detection on point clouds into point-based (Shi, Wang, and Li 2019; Yang et al. 2019; Yilun Chen and Jia 2019; Yang et al. 2020; Shi et al. 2020a) methods and voxel-based (Zhou and Tuzel 2018; Yan, Mao, and Li 2018; Yang, Luo, and Urtasun 2018; Simon et al. 2018; Lang et al. 2019; Liu et al. 2020b) methods. Here we give a brief review on these two directions:

Point-based Methods. Point-based methods take the raw point clouds as inputs, and abstract a set of point representations with iterative sampling and grouping (Qi et al. 2017a,b). PointRCNN (Shi, Wang, and Li 2019) introduces a 3D region proposal network based on PointNet++ (Qi et al. 2017b) and utilizes point cloud RoI pooling to extract 3D region features of each proposal. STD (Yang et al. 2019) exploits the PointsPool operation to extract points features of each proposal and transfer the sparse points into dense voxel representation. Then, 3D CNNs are applied on the voxelized region features for further refinement. 3DSSD (Yang et al. 2020) introduces F-FPS as a supplement of D-FPS and builds a one stage anchor-free 3D object detector based on feasible representative points. Different from grouping neighboring points for set abstraction, PV-RCNN (Shi et al. 2020a) devises voxel set abstraction to integrates multi-scale voxel features into sampled keypoints. By leveraging both the accurate position information from raw points and spatial context from voxel representations, PV-RCNN achieves remarkable improvements on 3D object detection.

Voxel-based Methods. Voxel-based methods typically discrete point clouds into equally spaced grids, and then capitalize on 2D/3D CNN to perform object detection. The early work VoxelNet (Zhou and Tuzel 2018) first divides points into 3D voxels and uses a tiny PointNet to transform points of each voxel into a compact feature representation. Then, 3D CNNs are leveraged to aggregate spatial context and generate 3D detections. However, due to the sparsity of non-empty voxels in the large space, it is inefficient to exploit conventional convolution networks for feature extraction.

To reduce the computation cost, SECOND (Yan, Mao, and Li 2018) introduces sparse 3D convolution for efficient voxel processing. SA-SSD (He et al. 2020) proposes an auxiliary network and losses on the basis of SECOND (Yan, Mao, and Li 2018) to preserve structure information. Different from the methods operating on voxels in 3D space, PointPillars (Lang et al. 2019) groups points as "pillars" and capitalizes on a simplified PointNet (Qi et al. 2017a,b) for pillar feature extraction before forming pseudo BEV images.

6 Conclusion

In this paper, we present Voxel R-CNN, a novel 3D object detector with voxel-based representations. Taking the voxels as input, Voxel R-CNN first generates dense region proposals from bird-eye-view feature representations, and subsequently, utilizes voxel RoI pooling to extract region features from 3D voxel features for further refinement. By taking full advantage of voxel representations, our Voxel R-CNN strikes a careful balance between accuracy and efficiency. The encouraging results on both KITTI Dataset and Waymo Open Dataset demonstrate our Voxel-RCNN can serve as a simple but effective baseline to facilitate the investigation of 3D object detection and other downstream tasks.

Acknowledgements

This work was supported by the National Major Program for Technological Innovation 2030—New Generation Artificial Intelligence (2018AAA0100500). The work of Wengang Zhou was supported in part by NSFC under Contract 61822208, 61632019 and 62021001, and in part by the Youth Innovation Promotion Association CAS under Grant 2018497. The work of Houqiang Li was supported by NSFC under Contract 61836011.

References

Caesar, H.; Bankiti, V.; Lang, A. H.; Vora, S.; Liong, V. E.; Xu, Q.; Krishnan, A.; Pan, Y.; Baldan, G.; and Beijbom, O. 2020. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 11621–11631.

Chen, X.; Ma, H.; Wan, J.; Li, B.; and Xia, T. 2017. Multi-view 3d object detection network for autonomous driving. In *CVPR*, 1907–1915.

Ge, R.; Ding, Z.; Hu, Y.; Wang, Y.; Chen, S.; Huang, L.; and Li, Y. 2020. Afdet: Anchor free one stage 3d object detection. *arXiv preprint arXiv:2006.12671* .

Geiger, A.; Lenz, P.; and Urtasun, R. 2012. Are we ready for autonomous driving? In *CVPR*, 3354–3361.

Girshick, R. 2015. Fast r-cnn. In *ICCV*, 1440–1448.

He, C.; Zeng, H.; Huang, J.; Hua, X.-S.; and Zhang, L. 2020. Structure aware single-stage 3d object detection from point cloud. In *CVPR*, 11873–11882.

Hu, Q.; Yang, B.; Xie, L.; Rosa, S.; Guo, Y.; Wang, Z.; Trigoni, N.; and Markham, A. 2020. RandLA-Net: Efficient semantic segmentation of large-scale point clouds. In *CVPR*, 11108–11117.

Jiang, B.; Luo, R.; Mao, J.; Xiao, T.; and Jiang, Y. 2018. Acquisition of localization confidence for accurate object detection. In *ECCV*, 784–799.

Ku, J.; Mozifian, M.; Lee, J.; Harakeh, A.; and Waslander, S. L. 2018. Joint 3d proposal generation and object detection from view aggregation. In *IROS*, 1–8.

Lang, A. H.; Vora, S.; Caesar, H.; Zhou, L.; Yang, J.; and Beijbom, O. 2019. PointPillars: Fast encoders for object detection from point clouds. In *CVPR*, 12697–12705.

Lehner, J.; Mitterecker, A.; Adler, T.; Hofmarcher, M.; Nessler, B.; and Hochreiter, S. 2019. Patch refinement–localized 3D object detection. *arXiv preprint arXiv:1910.04093* .

Li, B.; Ouyang, W.; Sheng, L.; Zeng, X.; and Wang, X. 2019. Gs3d: An efficient 3d object detection framework for autonomous driving. In *CVPR*, 1019–1028.

Liang, M.; Yang, B.; Chen, Y.; Hu, R.; and Urtasun, R. 2019. Multi-task multi-sensor fusion for 3d object detection. In *CVPR*, 7345–7353.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017. Focal loss for dense object detection. In *ICCV*, 2980–2988.

Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; and Berg, A. C. 2016. Ssd: Single shot multibox detector. In *ECCV*, 21–37.

Liu, Z.; Hu, H.; Cao, Y.; Zhang, Z.; and Tong, X. 2020a. A Closer Look at Local Aggregation Operators in Point Cloud Analysis. *arXiv preprint arXiv:2007.01294* .

Liu, Z.; Zhao, X.; Huang, T.; Hu, R.; Zhou, Y.; and Bai, X. 2020b. TANet: Robust 3D object detection from point clouds with triple attention. In *AAAI*, 11677–11684.

Qi, C. R.; Liu, W.; Wu, C.; Su, H.; and Guibas, L. J. 2018. Frustum pointnets for 3d object detection from rgb-d data. In *CVPR*, 918–927.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 652–660.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 5099–5108.

Redmon, J.; and Farhadi, A. 2017. YOLO9000: better, faster, stronger. In *CVPR*, 7263–7271.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 91–99.

Shi, S.; Guo, C.; Jiang, L.; Wang, Z.; Shi, J.; Wang, X.; and Li, H. 2020a. PV-RCNN: Point-voxel feature set abstraction for 3D object detection. In *CVPR*, 10529–10538.

Shi, S.; Wang, X.; and Li, H. 2019. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 770–779.

Shi, S.; Wang, Z.; Shi, J.; Wang, X.; and Li, H. 2020b. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *TPAMI* .

Simon, M.; Milz, S.; Amende, K.; and Gross, H.-M. 2018. Complex-YOLO: An euler-region-proposal for real-time 3D object detection on point clouds. In *ECCV*, 197–209.

Sun, P.; Kretzschmar, H.; Dotiwalla, X.; Chouard, A.; Patnaik, V.; Tsui, P.; Guo, J.; Zhou, Y.; Chai, Y.; Caine, B.; et al. 2020. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2446–2454.

Szegedy, C.; Ioffe, S.; Vanhoucke, V.; and Alemi, A. A. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, 4278–4284.

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

Yan, Y.; Mao, Y.; and Li, B. 2018. Second: Sparsely embedded convolutional detection. *Sensors* 3337.

Yang, B.; Luo, W.; and Urtasun, R. 2018. Pixor: Real-time 3d object detection from point clouds. In *CVPR*, 7652–7660.

Yang, Z.; Sun, Y.; Liu, S.; and Jia, J. 2020. 3DSSD: Point-based 3D single stage object detector. In *CVPR*, 11040–11048.

Yang, Z.; Sun, Y.; Liu, S.; Shen, X.; and Jia, J. 2019. STD: Sparse-to-dense 3D object detector for point cloud. In *ICCV*, 1951–1960.

Ye, M.; Xu, S.; and Cao, T. 2020. HVNet: Hybrid voxel network for LiDAR based 3D object detection. In *CVPR*, 1631–1640.

Yilun Chen, Shu Liu, X. S.; and Jia, J. 2019. Fast point r-cnn. In *ICCV*, 9774–9783.

Zhao, X.; Liu, Z.; Hu, R.; and Huang, K. 2019. 3D Object Detection Using Scale Invariant and Feature Reweighting Networks. In *AAAI*, 9267–9274.

Zhou, Y.; Sun, P.; Zhang, Y.; Anguelov, D.; Gao, J.; Ouyang, T.; Guo, J.; Ngiam, J.; and Vasudevan, V. 2020. End-to-end multi-view fusion for 3d object detection in lidar point clouds. In *CoRL*, 923–932.

Zhou, Y.; and Tuzel, O. 2018. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 4490–4499.