

PV-RCNN&PV-RCNN++

Update date: Sep 25, 2023.

Written by Lucas KIM

⚠ Before Reading

- My undergraduate major is not Computer Science.
- Only 'Basic' understanding of linear algebra, calculus, and probability and statistics.
- No experience presenting papers in the field of computer vision.
- Might be wrong, feedback is welcome.
- Not yet capable of fully understanding all the papers cited as references.

For today, I would appreciate it if you could consider my goal to be grasping the big picture.

PV-RCNN

Abstract

- 3D object detection from [Point clouds](#)
- Integrates both 3D voxel CNN and PointNet-based set abstraction
- PV-RCNN surpasses state-of-art 3D detection methods by using only point clouds.

📄 Code?





Please check [Official](#) code. KITTI and Waymo [Data](#)

Conclusion

- A novel method for **accurate 3D Object Detection** from point clouds
 - Integrates both the multi-scale 3D [Voxel](#) CNN features and the PointNet-based features
 - Significantly improve the 3D Object Detection performance
-

1. Intro

Most existing 3D detection methods could be classified into two categories (in terms of point cloud representations.)

- [Grid-based Methods](#)
 -  Computationally efficient
 -  The inevitable information loss degrades the fine-grained localization accuracy
- [Point-based Methods](#)
 -  could easily achieve larger receptive field
 -  higher computation cost

⇒ Thus, Integrate these two things (PV = Point-Voxel)

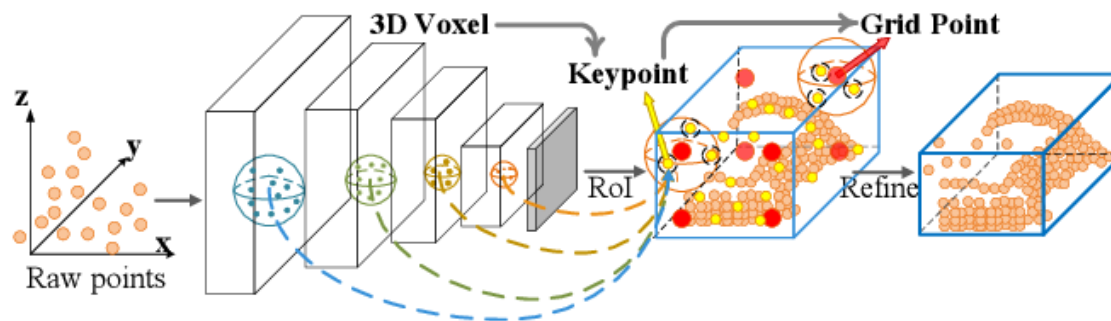


Figure 1. Our proposed PV-DCNN framework deeply integrates

- 1. Voxel-to-keypoint scene encoding step
 - Encodes multi-scale voxel features of the whole scene to a small set of keypoints by the voxel set abstraction layer
- 2. Keypoint-to-grid feature abstraction step

- Point-to-grid [RoI feature abstraction](#), which effectively aggregates the scene keypoint feature to RoI grids

2. Related work

- 3D Object Detection with Grid-based Methods
- 3D Object Detection with Point-based Methods
- Representation Learning on Point Clouds

3. PV-RCNN for Point Cloud Object Detection

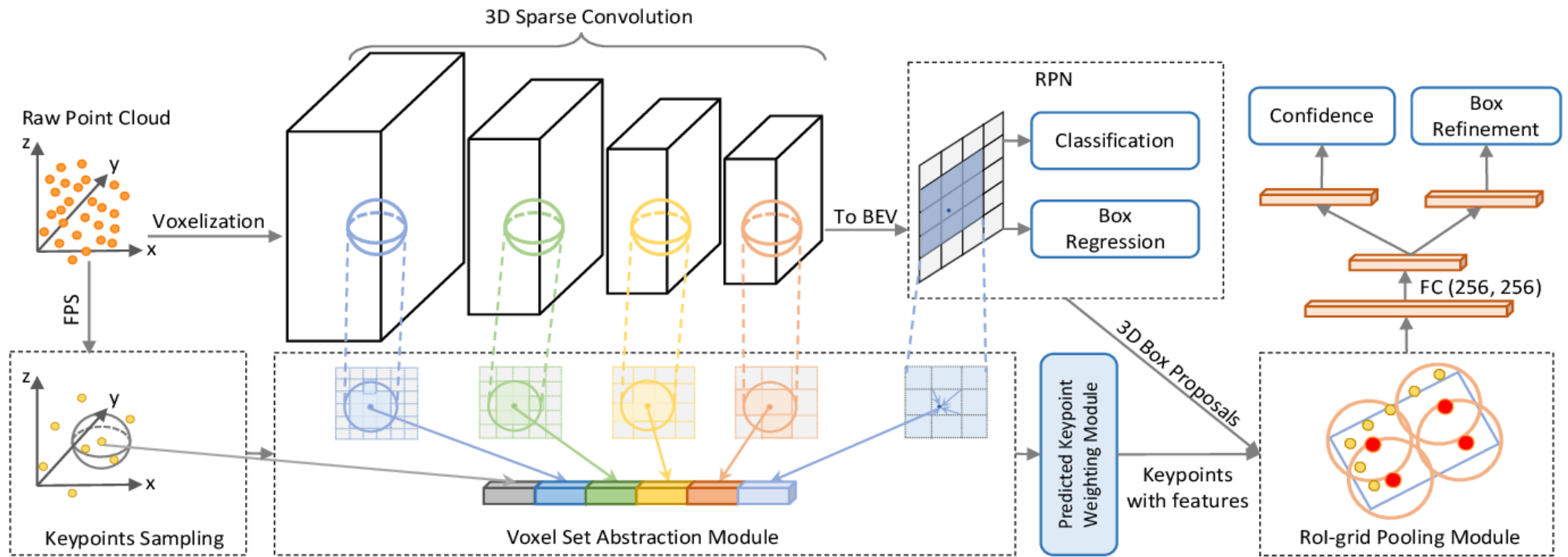


Figure 2. The overall architecture of our proposed PV-RCNN. The raw point clouds are first voxelized to feed into the 3D sparse convolution.

3.1 3D Voxel CNN for Efficient Feature Encoding and Proposal Generation

- 3D voxel CNN
 - The input point P are first divided into small voxel with spatial resolution of $L \times W \times H$
 - The features of the non-empty voxels are directly calculated as the mean of point-wise features of all inside points.

- Commonly used: [3D coordinates](#), [Reflectance intensities](#)
 - The network utilizes a series of $3 \times 3 \times 3$ 3D sparse convolution to gradually convert the point clouds into features volume with 1x, 2x, 4x, 8x
 - 3D proposal generation
 - By converting the encoded 8x downsampled 3D feature volumes into [2D bird-view feature maps](#), high-quality 3D proposals are generated following the [Anchor-Based approaches](#).
-

- Discussion
 - Pooling RoI specific features from the resulting 3D feature volumes or 2D maps has major limitations. *1. Low spatial resolution, 2. Waste much computation and memory*
 - The set abstraction has shown the strong capability of encoding feature points from a neighborhood of an arbitrary size.
- So? 🤔
 - Propose a *two step approach* to first encode voxels at different neural layers of the entire scene into a small number of keypoints and then aggregate keypoint features to RoI grids for box proposal refinement.

3.2 Voxel-to-keypoint Scene Encoding via Voxel Set Abstraction

First, aggregates the voxel at the multiple neural layers representing the entire scene into a small number of keypoints.

- Keypoints Sampling
 - Adopt [FPS](#) algorithm to sample a small number of n keypoints $K = p_1, \dots, p_n$ from the point clouds \mathbf{P}
 - $n = 2,048$ for the KITTI dataset and $n = 4,096$ for the Waymo dataset
 - Voxel Set Abstraction Module
 - To encode the multi-scale semantic features from the 3D CNN feature volumes to the keypoints
 - The surrounding points of keypoints are now regular voxels with multi-scale semantic features encoded by the 3D voxel CNN from the multiple levels, instead of the neighboring raw points with features learned from [PointNet](#)
-

Notation

- N_k is the number of non-empty voxels in the k -th level

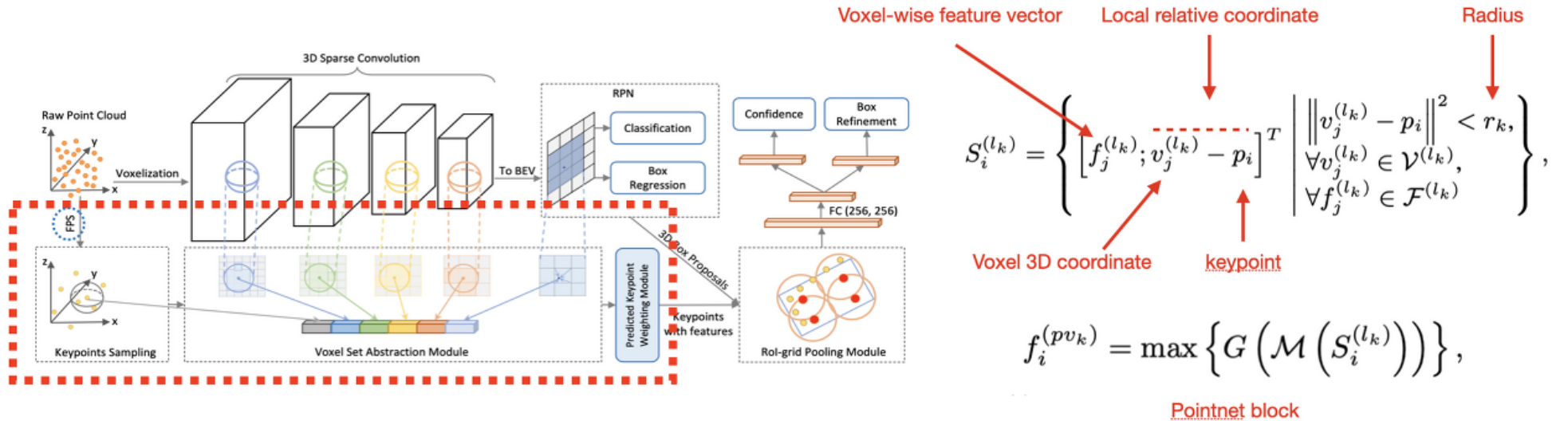
$$\mathcal{F}^{(l_k)} = \{f_1^{(l_k)}, \dots, f_{N_k}^{(l_k)}\}$$

- As the set of voxel-wise feature vectors in the k -th level of 3D Voxel CNN

$$\mathcal{V}^{(l_k)} = \{v_1^{(l_k)}, \dots, v_{N_k}^{(l_k)}\}$$

- As their 3D coordinates calculated by the voxel indices and actual voxel size of the k -th level

$$S_i^{(l_k)} = \left\{ \begin{array}{l} [f_j^{(l_k)}; v_j^{(l_k)} - p_i]^T \\ \left\| v_j^{(l_k)} - p_i \right\|^2 < r_k \\ \forall v_j^{(l_k)} \in \mathcal{V}^{(l_k)} \\ \forall f_j^{(l_k)} \in \mathcal{F}^{(l_k)} \end{array} \right\}$$



Additional explanation

Looking at the equation on the right,

- ' l_k ' represents the index of the feature volume
- ' f ' is a voxel-wise feature vector

- ' $v - p$ ' denotes the relative coordinates of a voxel with respect to a keypoint.

These two vectors are concatenated to form a set called ' S '. The term ' r_k ' in the equation on the right signifies a radius determined for each layer k .

By setting this radius differently for each layer, we can consider that our set abstraction operation possesses a flexible receptive field. In our implementation process, we have assigned two radii per layer.

$$f_i^{(pv_k)} = \max\{G(\mathcal{M}(S_i^{(l_k)}))\}$$

- $\mathcal{M}(\cdot)$ = randomly sampling at most T_k voxels from the neighboring set $S_i^{(l_k)}$ **for saving computation**.
- $G(\cdot)$ = a multi-layer perceptron network to encode the voxel-wise features and relative location.

Additional explanation

The set is transformed into a feature vector for the keypoint through a PointNet block.

Looking at the PointNet block, ' \mathcal{M} ' denotes maximum T random samplings. ' G ' represents a multilayer-perceptron layer, and 'max' is a max pooling layer.

Through this process, we obtain the feature vector from each feature volume for each keypoint.

$$f_i^{(pv)} = [f_i^{(pv1)}, f_i^{(pv2)}, f_i^{(pv3)}, f_i^{(pv4)}]$$

$$f_i^{(p)} = [f_i^{(pv)}, f_i^{(raw)}, f_i^{(bev)}]$$

- for $i = 1, \dots, n$,

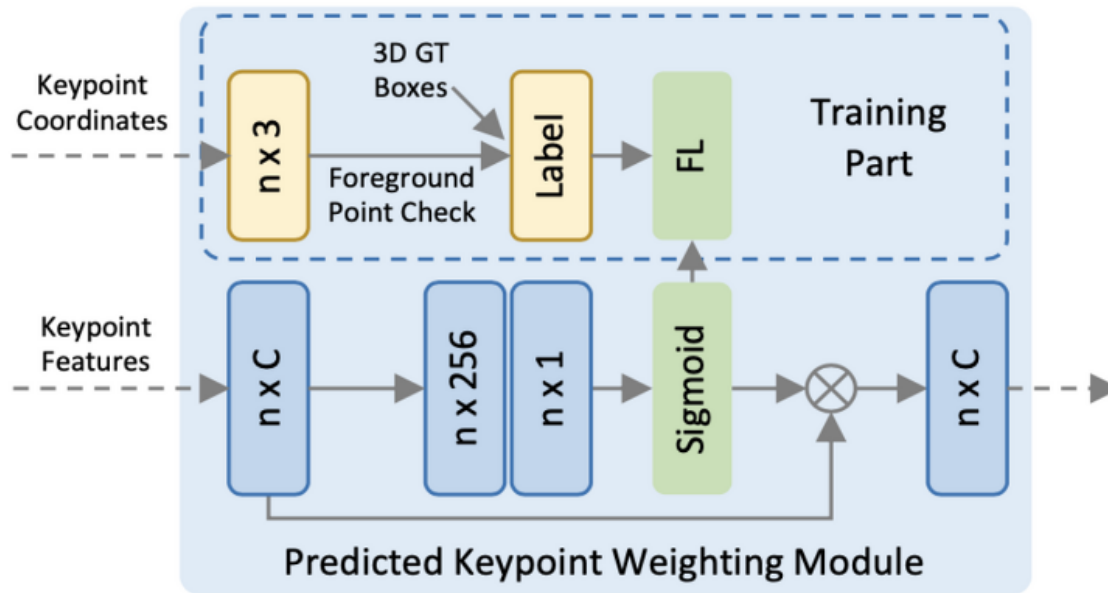
Additional explanation

Gather the key point features calculated for each layer, and combine these with the set abstraction of raw data and the key point features obtained from a bird's eye view to form the final key point feature.

The raw data was added to recover any quantization loss that may occur during voxelization, and the bird's eye view was incorporated to gain a wider receptive field along the z-axis.

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

- \mathcal{A} is a three-layer [MLP Network](#) with a sigmoid function to predict foreground confidence between $[0, 1]$
- PKW module is trained by [Focal loss](#) for handling the unbalanced number of foreground/ background points in the training set.



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

$\mathcal{A}(\cdot)$ Three-layer MLP network
Loss function : focal loss

Additional explanation

This module is an implementation of the idea that foreground keypoints should contribute more than background keypoints during the refinement process.

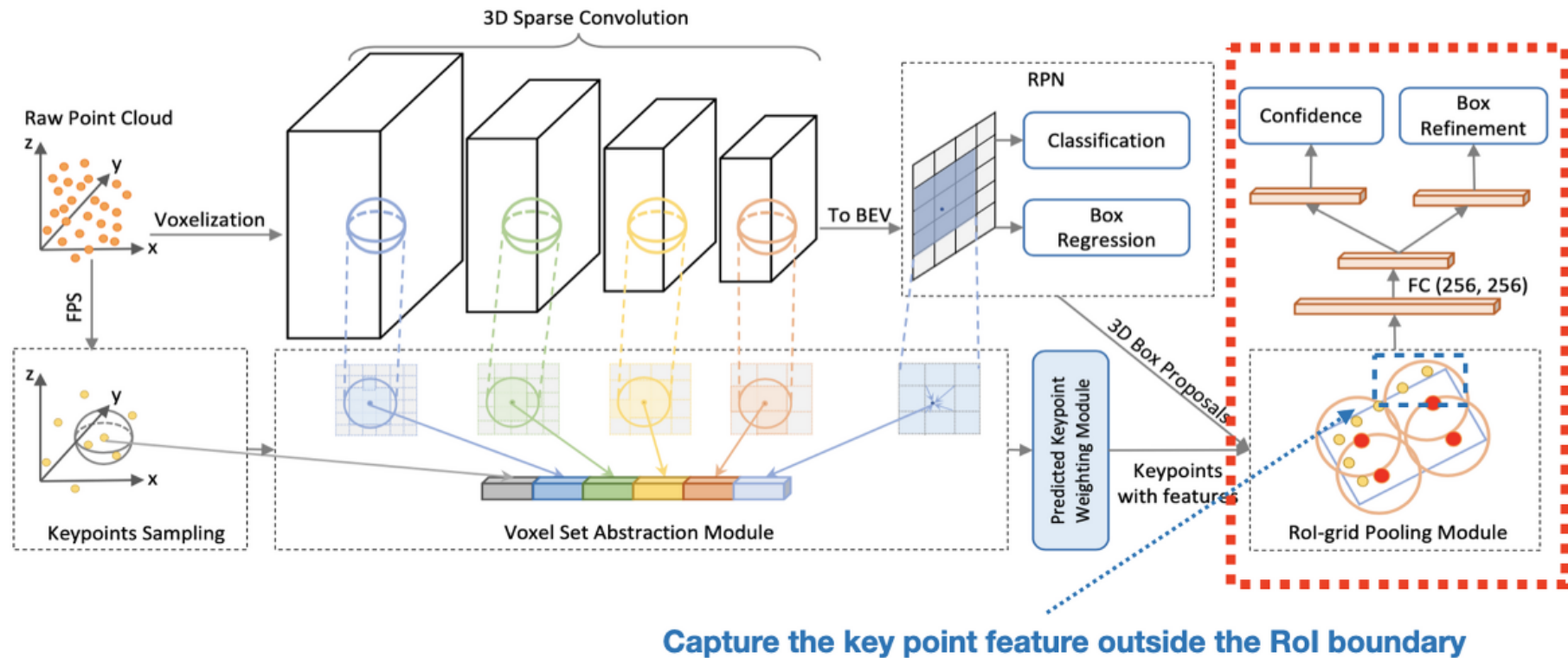
We added a Point Cloud Segmentation network to calculate the confidence of whether a given point is in the [Foreground or Background](#), which is then given as a weight.

The points were trained as foreground if they were inside the ground truth 3D box and as background if they were outside, without separate segmentation labels.

In actual operation, we multiplied the learned network's foreground confidence by the keypoint feature to ensure that keypoints in the foreground have a greater impact on refinement.

During training, we addressed issues of unbalanced distribution in our training set using focal loss.

3.3 Keypoint-to-grid RoI Feature Abstraction for Proposal Refinement



- ROI-Grid Pooling via Set Abstraction

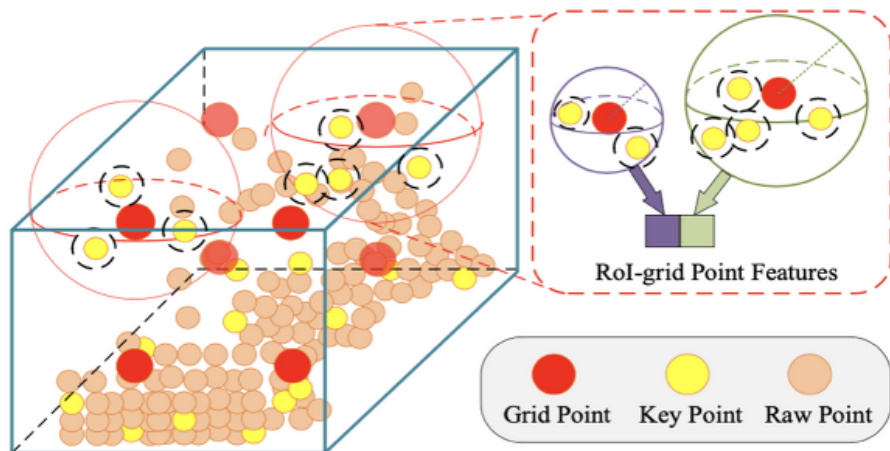


Figure 4. Illustration of RoI-grid pooling module. Rich context information of each 3D RoI is aggregated by the set abstraction operation with multiple receptive fields.

$$\tilde{\Psi} = \left\{ \left[\tilde{f}_j^{(p)}; p_j - g_i \right]^T \mid \begin{array}{l} \|p_j - g_i\|^2 < \tilde{r}, \\ \forall p_j \in \mathcal{K}, \forall \tilde{f}_j^{(p)} \in \tilde{\mathcal{F}} \end{array} \right\},$$

keypoint-wise feature vector Local relative coordinate Radius

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

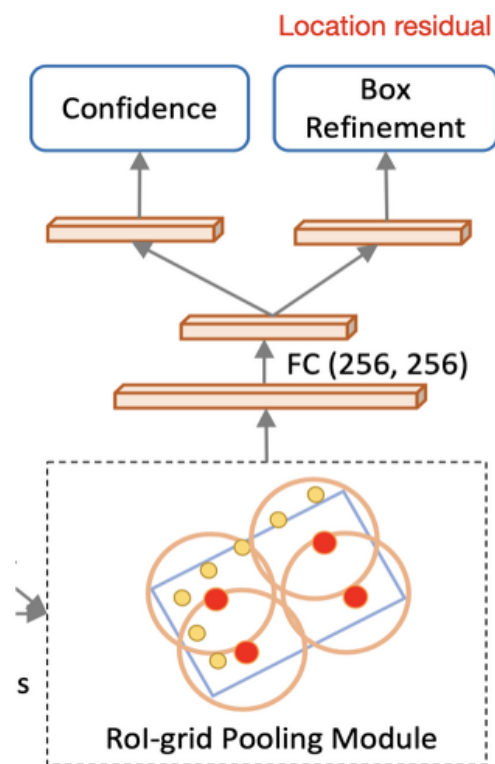
Pointnet block

i Additional explanation

The red dots are grid points, the yellow dots are key points, and the small peach-like dots are raw points. From each grid point, a key point set is formed through various distances (receptive fields).

From this collected key point feature set, T samples are taken. A grid point feature is generated through a multilayer-perceptron layer and a max pooling layer.

- 3D Proposal Refinement and Confidence Prediction
 - The grid point features of each 3D proposal obtained earlier are processed through a 2-layer MLP to create an RoI feature vector with a dimension of 256 features.
 - Subsequently, through a network composed of two branches, we calculate confidence and box refinement.



$$y_k = \min(1, \max(0, 2\text{IoU}_k - 0.5)),$$

IoU guided score

$$L_{\text{iou}} = -y_k \log(\tilde{y}_k) - (1 - y_k) \log(1 - \tilde{y}_k),$$

Predicted score

Notation

$$y_k = \min(1, \max(0, 2\text{IoU}_k - 0.5))$$

Additional explanation

The first equation is used to calculate the confidence of each candidate RoI during training through IOU, not simply having a confidence of 0 or 1.

This allows for a more nuanced learning process where better candidates are predicted to have a confidence closer to 1, and worse candidates closer to 0.

This can result in a network that can better judge which box proposal is superior during testing, compared to a conventional binary classification into 0 or 1.

$$L_{iou} = -y_k \log(\tilde{y}_k) - (1 - y_k) \log(1 - \tilde{y}_k)$$


Additional explanation

The second equation is the loss function for the confidence branch, which minimizes cross-entropy loss.

In the box refinement branch, location residuals (errors) such as center, size, and orientation are predicted and optimized through a smooth L1 loss function.

3.4 Training losses

$$\mathcal{L}_{total} = L_{rpn} + L_{seg} + L_{renn}$$

 The overall training loss are the sum of these three details

$$L_{rpn} = L_{cls} + \beta \sum_{r \in \{x, y, z, l, h, w, \theta\}} + \mathcal{L}_{smooth-L1}(\widehat{\Delta r^a}, \Delta r^a)$$

$$L_{renn} = L_{iou} + \sum_{r \in \{x, y, z, l, h, w, \theta\}} + \mathcal{L}_{smooth-L1}(\widehat{\Delta r^a}, \Delta r^a)$$

Additional explanation

The region proposal loss L_{rpn} is the loss from the two branches in the Region Proposal Network (RPN).

L_{cls} is the anchor classification loss, and $smooth - L_1 loss$ refers to the box regression branch's loss.

L_{seg} is the segmentation network's loss from the PKW module we examined earlier.

The proposal refinement loss L_{rcnn} represents the losses from both confidence branch and box refinement branch that we've discussed previously

4. Experiments

- Training Detail
 - For KITTI **8 GTX 1080 Ti GPUs**, for Waymo **32 GTX 1080 Ti GPUs** 🙄

Method	Reference	Modality	Car - 3D Detection			Car - BEV Detection			Cyclist - 3D Detection			Cyclist - BEV Detection		
			Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
MV3D [1]	CVPR 2017	RGB + LiDAR	74.97	63.63	54.00	86.62	78.93	69.80	-	-	-	-	-	-
ContFuse [17]	ECCV 2018	RGB + LiDAR	83.68	68.78	61.67	94.07	85.35	75.88	-	-	-	-	-	-
AVOD-FPN [11]	IROS 2018	RGB + LiDAR	83.07	71.76	65.73	90.99	84.82	79.62	63.76	50.55	44.93	69.39	57.12	51.09
F-PointNet [22]	CVPR 2018	RGB + LiDAR	82.19	69.79	60.59	91.17	84.67	74.77	72.27	56.12	49.01	77.26	61.37	53.78
UberATG-MMF [16]	CVPR 2019	RGB + LiDAR	88.40	77.43	70.22	93.67	88.21	81.99	-	-	-	-	-	-
SECOND [34]	Sensors 2018	LiDAR only	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 [12]	CVPR 2019	LiDAR only	82.58	74.31	68.99	90.07	86.56	82.81	77.10	58.65	51.92	79.90	62.73	55.58
PointRCNN [25]	CVPR 2019	LiDAR only	86.96	75.64	70.70	92.13	87.39	82.72	74.96	58.82	52.53	82.56	67.24	60.28
3D IoU Loss [39]	3DV 2019	LiDAR only	86.16	76.50	71.39	91.36	86.22	81.20	-	-	-	-	-	-
Fast Point R-CNN [2]	ICCV 2019	LiDAR only	85.29	77.40	70.24	90.87	87.84	80.52	-	-	-	-	-	-
STD [37]	ICCV 2019	LiDAR only	87.95	79.71	75.09	94.74	89.19	86.42	78.69	61.59	55.30	81.36	67.23	59.35
Patches [13]	Arxiv 2019	LiDAR only	88.67	77.20	71.82	92.72	88.39	83.19	-	-	-	-	-	-
Part- A^2 [26]	Arxiv 2019	LiDAR only	87.81	78.49	73.51	91.70	87.79	84.61	-	-	-	-	-	-
PV-RCNN (Ours)	-	LiDAR only	90.25	81.43	76.82	94.98	90.65	86.14	78.60	63.71	57.65	82.49	68.89	62.41
<i>Improvement</i>	-	-	<i>+1.58</i>	<i>+1.72</i>	<i>+1.73</i>	<i>+0.24</i>	<i>+1.46</i>	<i>-0.28</i>	<i>-0.06</i>	<i>+2.12</i>	<i>+2.35</i>	<i>-0.07</i>	<i>+1.65</i>	<i>+2.13</i>

Difficulty	Method	3D mAP (IoU=0.7)				3D mAPH (IoU=0.7)				BEV mAP (IoU=0.7)				BEV mAPH (IoU=0.7)			
		Overall	0-30m	30-50m	50m-Inf	Overall	0-30m	30-50m	50m-Inf	Overall	0-30m	30-50m	50m-Inf	Overall	0-30m	30-50m	50m-Inf
LEVEL_1	PointPillar [12]	56.62	81.01	51.75	27.94	-	-	-	-	75.57	92.1	74.06	55.47	-	-	-	-
	MVF [40]	62.93	86.30	60.02	36.02	-	-	-	-	80.40	93.59	79.21	63.09	-	-	-	-
	PV-RCNN (Ours)	70.30	91.92	69.21	42.17	69.69	91.34	68.53	41.31	82.96	97.35	82.99	64.97	82.06	96.71	82.01	63.15
	<i>Improvement</i>	<i>+7.37</i>	<i>+5.62</i>	<i>+9.19</i>	<i>+6.15</i>	<i>-</i>	<i>-</i>	<i>-</i>	<i>-</i>	<i>+2.56</i>	<i>+3.76</i>	<i>+3.78</i>	<i>+1.88</i>	<i>-</i>	<i>-</i>	<i>-</i>	<i>-</i>
LEVEL_2	PV-RCNN (Ours)	65.36	91.58	65.13	36.46	64.79	91.00	64.49	35.70	77.45	94.64	80.39	55.39	76.60	94.03	79.40	53.82

Table 5. Performance comparison on the Waymo Open Dataset with 202 validation sequences for the vehicle detection. Note that the

Table 1. Performance comparison on the KITTI test set. The results

Method	Reference	Modality	3D mAP
MV3D [1]	CVPR 2017	RGB + LiDAR	62.68
ContFuse[17]	ECCV 2018	RGB + LiDAR	73.25
AVOD-FPN [11]	IROS 2018	RGB + LiDAR	74.44
F-PointNet [22]	CVPR 2018	RGB + LiDAR	70.92
VoxelNet [41]	CVPR 2018	LiDAR only	65.46
SECOND [34]	Sensors 2018	LiDAR only	76.48
PointRCNN [25]	CVPR 2019	LiDAR only	78.63
Fast Point R-CNN [2]	ICCV 2019	LiDAR only	79.00
STD [37]	ICCV 2019	LiDAR only	79.80
PV-RCNN (Ours)	-	LiDAR only	83.90

Table 2. Performance comparison on the moderate level car class of KITTI val split with mAP calculated by 11 recall positions

Method	RPN with 3D Keypoints RoI-grid			Easy Mod. Hard		
	Voxel CNN	Encoding	Pooling			
RPN Baseline	✓			90.46	80.87	77.30
Pool from Encoder	✓		✓	91.88	82.86	80.52
PV-RCNN	✓	✓	✓	92.57	84.83	82.69

Table 6. Effects of voxel-to-keypoint scene encoding strategy and RoI-grid pooling on performance

🔗 Reference

- <https://hblog.tistory.com/7>
- <https://jaehoon-daddy.tistory.com/57>
- <https://velog.io/@treeboy2762/Week-6-PV-RCNN>