Graph R-CNN: Towards Accurate 3D Object Detection with Semantic-Decorated Local Graph

Honghui Yang[1], Zili Liu[1,3], Xiaopei Wu[1], Wenxiao Wang[2(✉)], Wei Qian[3], Xiaofei He[1,3], and Deng Cai[1]

[1] State Key Lab of CAD&CG, Zhejiang University
[2] School of Software Technology, Zhejiang University
[3] Fabu Inc.
{yanghonghui,wuxiaopei,wenxiaowang,dcai}@zju.edu.cn
{zililiuzju}@gmail.com
{qianwei,hexiaofei}@fabu.ai

Abstract. Two-stage detectors have gained much popularity in 3D object detection. Most two-stage 3D detectors utilize grid points, voxel grids, or sampled keypoints for RoI feature extraction in the second stage. Such methods, however, are inefficient in handling unevenly distributed and sparse outdoor points. This paper solves this problem in three aspects. 1) Dynamic Point Aggregation. We propose the patch search to quickly search points in a local region for each 3D proposal. The dynamic farthest voxel sampling is then applied to evenly sample the points. Especially, the voxel size varies along the distance to accommodate the uneven distribution of points. 2) RoI-graph Pooling. We build local graphs on the sampled points to better model contextual information and mine point relations through iterative message passing. 3) Visual Features Augmentation. We introduce a simple yet effective fusion strategy to compensate for sparse LiDAR points with limited semantic cues. Based on these modules, we construct our Graph R-CNN as the second stage, which can be applied to existing one-stage detectors to consistently improve the detection performance. Extensive experiments show that Graph R-CNN outperforms the state-of-the-art 3D detection models by a large margin on both the KITTI and Waymo Open Dataset. And we rank first place on the KITTI BEV car detection leaderboard.

Keywords: 3D object detection, point clouds, multiple sensors

1 Introduction

In autonomous driving, 3D object detection is an essential task that has received substantial attention from industry [1,12,38] and academia [24,39,47]. Among the existing 3D detection methods, two-stage detectors [25,42] outperform most single-stage detectors [12,38] in accuracy due to the proposal refinement stage. Previous two-stage methods [7,14,17,24,25,26,40] have explored different RoI pooling methods to capture better proposal features for refinement. PointRCNN [25] and its subsequent works [14] sample keypoints from original

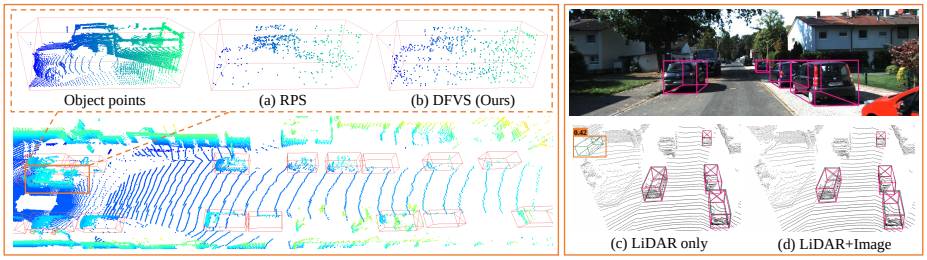

Fig. 1. Illustration of different sampling strategies: (a) random point sampling (RPS) in existing works and the proposed (b) dynamic farthest voxel sampling (DFVS). And the comparison of results using (c) LiDAR and (d) LiDAR and Image. We show the ground truth in pink bounding boxes and our detected objects in green bounding boxes.

point clouds near the 3D proposal and extract the features of the sampled points. Part-A^2 Net [26] divides each 3D proposal into regular voxel grids and applies sparse convolutions to capture the features of voxel grids. PV-RCNN [24] and its variants [7,17,35] sample grid points within each 3D proposal and use the set abstraction [22] to aggregate the features of grid points. The methods that utilize the sampled keypoints show more flexibility than others since they directly process raw points and avoid the predefined voxel grids [16,40] or grid points [6,31] as intermediaries for RoI feature extraction.

Nevertheless, existing methods relied on sampled points still have some problems: **1)** They ignore that points are unevenly distributed in different parts of an object, thus yield sub-optimal sampling strategy. As Fig. 1(a) shows, points for some parts are too sparse to preserve the structure information, which will hinder the prediction of the object's size. **2)** The point interrelation is not adequately utilized to model the contextual information of sparse points for object detection. **3)** Sparse LiDAR points in a single proposal provide limited semantic cues, which easily leads to a series of points that resemble a part of an object yielding high classification scores. Fig. 1(c) shows that a wall corner is wrongly detected as a car. To surmount the above challenges, we introduce three modules:

1) Dynamic Point Aggregation (DPA). To efficiently and effectively group and sample context and object points for 3D proposals, we propose patch search (PS) and dynamic farthest voxel sampling (DFVS). We will start with PS to speed up grouping then move on to DFVS to solve the uneven distribution problem during sampling.

Previous methods [5,25,26] group points by searching all points to determine whether they belong to a proposal, which is time-consuming since the theoretical time complexity is $O(NM)$, where N and M are the number of points and proposals, respectively. Especially for the detection on Waymo Open Dataset [28], there are often about 180K points and 500 proposals per frame that need to be processed. In contrast, PS only searches the points falling in patches occupied by the proposal to group corresponding context and object points. Then, DFVS sample keypoints from the grouped points to well retain the objects' structure,

as shown in Fig. 1(b). Specifically, instead of sampling points directly [14,23,25], DFVS splits each proposal into evenly distributed voxels and iteratively samples the most distant non-empty voxels (i.e., voxels involving at least one point). Further, to ensure the efficiency and accuracy of sampling, we resort to dynamic voxel size. That is, for nearby objects with many points, we use a large voxel size to reduce the sampling complexity. While for distant objects with sparse points, we use a relatively small voxel size to preserve the geometric details.

2) RoI-graph Pooling (RGP). To alleviate the missing detection, we utilize the graph neural network (GNN) to build connections among points to better model contextual information through iterative message passing. Compared with previous point-based methods [21,22], the GNN allows more complex features to be determined along the edges and avoids grouping and sampling the points repeatedly. Specifically, RGP constructs local graphs in each 3D proposal, which treats the sampled points as graph nodes. To compensate for the information loss caused by downsampling, we use PointNet [21] to encode neighbor points of each node into the initial features. Then, RGP iteratively aggregates messages from its neighbors on a k-NN graph to mine the relations among nodes. Finally, we propose multi-level attentive fusion (MLAF) to capture abundant spatial features from multi-level nodes with different receptive fields and fully exploit graph nodes to extract robust RoI features.

3) Visual Features Augmentation (VFA). Though LiDAR points provide accurate depth information, the lack of sufficient semantic features makes it difficult to distinguish objects with similar geometric structures. Thus, a simple yet effective fusion method is used to fuse geometric features from LiDAR and semantic features from images for suppressing the false positives, as shown in Fig. 1(d). We decorate local graphs with image features by bilinear interpolation since graph nodes serve as a natural bridge between the LiDAR and image. We train the two streams in an end-to-end manner and show the complex multi-modality cut-and-paste augmentation [30,45] is not necessary for our framework.

Based on the three modules, we present our Graph R-CNN that can replace the second stage of other two-stage detectors or supplement any one-stage detector for further improvement. Extensive experiments have been conducted on several detection benchmarks to verify the effectiveness of our approach. We consistently improve existing 3D detectors by a large margin and achieve new state-of-the-art results on both Waymo Open Dataset (WOD) [28] and KITTI [8].

Our contributions can be summarized as follows:

- We fully consider the uneven distribution of point clouds and propose dynamic point aggregation (DPA).
- We introduce RoI-graph Pooling (RGP) to capture the robust RoI features by iterative graph-based message passing.
- We demonstrate a simple yet effective fusion strategy (VFA) to fuse image features with point features during the refinement stage.
- We present an accurate and efficient 3D object detector (Graph R-CNN) that can be applied to existing 3D detectors. Extensive experiments are conducted to verify the effectiveness of our methods.

2 Related Works

3D Object Detection Using Point Cloud. Current 3D detectors can be mainly divided into two streams: one-stage and two-stage methods. One-stage detectors jointly predict an output class and location of objects at the projected volumetric grids or downsampled points. SECOND [38] rasterizes point cloud into 3D voxels and accelerates VoxelNet [50] by exploiting sparse 3D convolution. PointPillars [12] partitions points into pillars rather than voxels. 3DSSD [39] proposes a fusion farthest point sampling strategy by utilizing both feature and geometry distance for better classification performance.

Two-stage detectors first use a region proposal network (RPN) to generate coarse object proposals and then use a dedicated per-region head to classify and refine them. PointRCNN [25] generates RoIs based on foreground points from the scene and conducts canonical 3D box refinement after point cloud region pooling. PV-RCNN [24] incorporates the advantage from 3D voxel Convolutional Neural Network and Point-based set abstraction to learn discriminative point cloud features. Voxel R-CNN [7] proposes a voxel RoI pooling to extract RoI features directly from voxel features to refine proposals in the second stage. CenterPoint [42] detects centers of objects using a keypoint detector and refines these estimates using additional point features on the object.

3D Object Detection Using Multi-modality Fusion. Recently, much progress has been made to exploit the advantages of the camera and LiDAR sensors. MV3D [4] generates 3D proposals from the bird's eye view and fuses multi-view features via region-based representation. EPNet [11] proposes LI-Fusion module to fuse the deep features of point clouds and camera images in a point-wise paradigm. However, insufficient multi-modality augmentation makes these methods perform only marginally better or sometimes worse than approaches that only use point cloud. Recent works [30,45] overcome the constraint by extending the cut-and-paste augmentation [38] to multi-modality methods. But a complex process is needed to avoid collisions between objects in both point cloud and 2D imagery domain. PointPainting [29] augments LiDAR points with segmentation scores, which are suboptimal to cover color and textures in images.

3D Object Detection Using Graph Neural Networks. Graph Neural Networks [9] are introduced to model intrinsic relationships of graph-structured data. Since they are suitable for processing 3D point clouds, some works have adopted GNNs for 3D object detection. 3DVID [41] explores spatial relations among different grid regions by treating the non-empty pillar grids as graph nodes to enhance pillar features. Object DGCNN [33] uses DGCNN [34] to construct a graph between the queries for incorporating neighborhood information in object detection estimates. DOPS [19] creates a graph where the points are connected to those with similar center predictions for consolidating the per-point object predictions. Point-GNN [27] encodes point clouds in a fixed radius near-neighbors graph and predicts the category and shape of the object that each node in the graph belongs to. Our work differs from previous works by constructing local graphs during the

refinement stage, which greatly saves computational and memory overhead since the k-nearest neighbor algorithm can be parallelly applied in each 3D proposal, and numerous background points can be avoided to build graphs.

3 Methods

In this section, we present the design of Graph R-CNN, as shown in Fig. 2. We first introduce dynamic point aggregation in Sec. 3.1. Next, we will demonstrate RoI-graph pooling in Sec. 3.2. Then, we will illustrate how to incorporate semantic features from the image into our framework in Sec. 3.3. Finally, we will show the definition of the loss function in Sec. 3.4.

3.1 Dynamic Point Aggregation

In this section, we present a differentiable dynamic point aggregation (DPA) to efficiently and effectively group and sample points and their features for each proposal. We first enlarge each proposal's size by σ to wrap enough object and context points. Then, DPA uses patch search (PS) to quickly group the points in each enlarged proposal and dynamic farthest voxel sampling (DFVS) to evenly sample the grouped points.

Patch Search. Unlike previous works [14,23,25] that need to search all points to determine whether they belong to an enlarged proposal, we divide the entire scene into patches and only search the points falling in patches occupied by the proposal, as shown in Fig. 3. PS consists of three major steps: 1) We turn the rotated box into an axis-aligned box to make it easier to find the occupied patches. 2) We build `point2patch` and `patch2box` index arrays, which store the point and patch indices as keys, and the corresponding patch and box indices as values, respectively. 3) We finally group the points for each proposal according to the `point2patch` and `patch2box` index arrays, as shown in Fig. 3(a). We note that all the steps can be conducted in parallel on GPUs. In this way, we reduce the theoretical time complexity from $O(NM)$ to $O(QK)$, where Q is the number of points that fall in all occupied patches, and K is the predefined maximum number of boxes per patch since the same patch may be occupied by multiple boxes. Notably, Q and K are much smaller than N and M, respectively.

Dynamic Farthest Voxel Sampling. Since the number of raw points in a box is usually far more than that of the sampled points (e.g., 70112 vs. 256, as Fig. 4(b) shows), it's nontrivial to ensure every part of an object is sampled. Therefore, we propose DFVS to balance sampling efficiency and accuracy. To be specific, DFVS partitions proposals into evenly distributed voxels and then iteratively sample the most distant non-empty voxels. Considering the number of points varies with the distance of the box, as Fig. 4(a) shows, the voxel size should be changed dynamically according to the distance to ensure the sampling efficiency

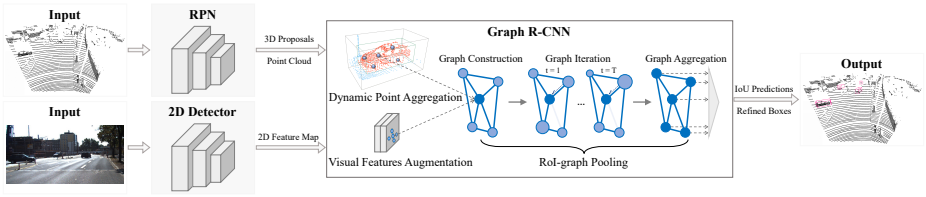

Fig. 2. The overall architecture. We take 3D proposals and points from the region proposal network (RPN) and 2D feature map from the 2D detector as inputs. We propose dynamic point aggregation to sample context and object points and visual features augmentation to decorate the points with 2D features. RoI-graph pooling serves sampled points as graph nodes to build local graphs for each 3D proposal. We iterate the graph for T times to mine the geometric features among the nodes. Finally, each node is fully utilized through graph aggregation to produce robust RoI features.

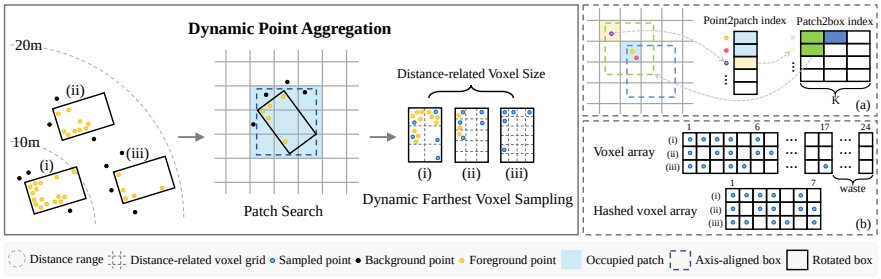

Fig. 3. Illustration of dynamic point aggregation, which includes (a) patch search and (b) dynamic farthest voxel sampling. In (a), we use different colors to represent different keys and values. In (b), we flatten the voxel grids of each proposal for better display.

of nearby objects and accuracy of distant objects, as shown in Fig. 3. Formally, the voxel size V_i of the box b_i can be calculated by:

$$V_i = \lambda \cdot e^{-\frac{\sqrt{x_i^2+y_i^2+z_i^2}}{\delta}} \tag{1}$$

where (x_i, y_i, z_i) is the i-th box's center, and λ and δ determine the relationship between the voxel size and the distance from the box to the LiDAR sensor.

Assume we have grouped the points in the box b_i by patch search and obtained $\mathcal{P}_i = \{p_j^i = [x_j^i, y_j^i, z_j^i, r_j^i] \in \mathbb{R}^4 : j = 1, \cdots, N\}$, where (x_j^i, y_j^i, z_j^i) and r_j^i indicate j-th point's coordinate in the i-th box's canonical coordinate system and the reflectance intensity. We assign each point to evenly divided voxel grids, and the grid index of the point p_j^i is represented as $\{g_j^i = (\lfloor \frac{x_j^i}{V_i} \rfloor, \lfloor \frac{y_j^i}{V_i} \rfloor, \lfloor \frac{z_j^i}{V_i} \rfloor)\}$. Each non-empty voxel can be represented by a randomly selected point in the voxel. Next, farthest point sampling (FPS) [22] is applied to iteratively sample the most distant non-empty voxels.

A potential problem with DVFS lies in that, for the distant box, a small voxel size will divide the box into numerous voxels, of which the non-empty voxels only

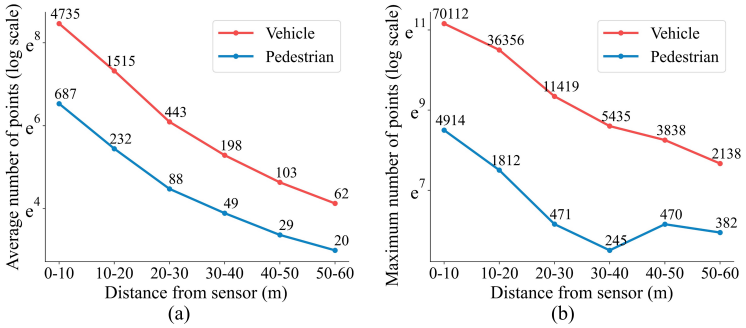

Fig. 4. The statistical plot of the (a) average and (b) maximum number of points in each ground truth on Waymo Open Dataset for vehicle and pedestrian.

occupy a small part. And to utilize the parallel computation of GPUs, voxel grids of other boxes need to be padded to the largest grid number, which will increase the memory overhead, as shown in Fig. 3(b). Since we only care about non-empty voxels, we use a hash table [18] to record the hashed grid indices of non-empty voxels and quadratic probing to resolve the collisions in the hash table.

3.2 RoI-graph Pooling

In this section, we describe the process of RoI-graph pooling, as shown in Fig. 2, which treats sampled points as nodes to build local graphs in 3D proposals. It consists of graph construction, iteration, and aggregation.

Graph Construction. Given sampled points $\overline{\mathcal{P}} = \{p_j = [x_j, y_j, z_j, r_j] \in \mathbb{R}^4 : j = 1, \cdots, T\}$ for each proposal b (we drop the i subscript for ease of notation), we construct a local graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where node $v_j \in \mathcal{V}$ represents a sampled point $p_j \in \overline{\mathcal{P}}$, and edge $e_j^k \in \mathcal{E}$ indicates the connection between node v_j and v_k. To reduce the computational overhead, we define $\mathcal{G}$ as a k-nearest neighbor (k-NN) graph, which is built from the geometric distance among different nodes. Despite efficient, building graphs on down-sampled points inevitably loss fine-grained features. Thus, we use PointNet [21] to encode original neighbor points within a radiu r for each node. We note that neighbor query only induces a marginal computational overhead because it is only conducted for each proposal.

The same graph nodes may be wrapped by different proposals, which will result in the same pooling features and thus introduce ambiguity in the refinement stage [14,26]. Inspired by [23], we add the 3D proposal's local corners (i.e., the corners are transformed to the proposal's canonical coordinate system) for each node to make them have the ability to discriminate differences. In our experiments, we found that two diagonal corners are sufficient. Formally, the initial state s_j^0 of each node v_j at iteration step $t = 0$ can be represented by:

$$s_j^0 = [x_j, y_j, z_j, r_j, f_j, u_j, w_j]\,, \tag{2}$$

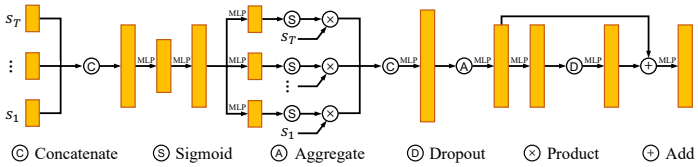

Fig. 5. Illustration of multi-level attentive fusion.

where $[\cdot,\cdot]$ is concatenation function, f_j is the features from PointNet, and u_j and w_j are two diagonal corners of the 3D proposal.

Graph Iteration. To mine the rich geometric relations among nodes, we iteratively pass the message on $\mathcal{G}$ and update the node's state at each iteration step. Concretely, at step t, a node v_j aggregates information from all the neighbor nodes $v_k \in \mathcal{N}_{v_j}$ in the k-NN graph. Following [2,33,41], we use EdgeConv [34] to update the state s_j^{t+1}:

$$s_j^{t+1} = \max_{v_k \in \mathcal{N}_{v_j}} \phi_\theta([s_k^t - s_j^t, s_j^t]), \tag{3}$$

where ϕ_θ is parameterized by a Multilayer Perceptron (MLP).

Graph Aggregation. To capture robust RoI features, we propose multi-level attentive fusion (MLAF) to fuse the nodes' features, as shown in Fig. 5. Specifically, we concatenate the nodes' features $[s_j^1, \cdots, s_j^T]$ from different iterations and feed them into several MLPs to learn the channel-wise weights. Then, we reweight $[s_j^1, \cdots, s_j^T]$ to enhance the features for final detection. After that, it's nontrivial to fully utilize every graph node for the proposal refinement. We explore several aggregation operations, e.g., the channel-wise Transformer [23], Set Transformer [13], attention sum, average pooling, and max pooling. Our final model uses max pooling as it provides the best empirical performance. Then, Dropout is used in later MLPs to reduce overfitting, and a shortcut connection is added to fuse more features without adding much cost.

3.3 Visual Features Augmentation

Cut-and-paste augmentation (CPA) [38] is widely used for 3D object detection to increase the training samples, which could speed up training convergence and improve the detection performance. Since our Graph R-CNN extracts RoI features directly from raw point clouds, we doubt whether CPA is required for our model. We carefully study its influence in Sec. 4.4 and find that our model does not depend on it. Thus, CPA is only used to pretrain the RPN and disabled when training the whole framework.

For the camera image, we extract high-level semantic features using a pretrained 2D detector. Then, we apply two 1×1 convolutional kernels to reduce the dimensionality of the output feature, as Fig. 2 shows. The benefit brought

by it is twofold. Firstly, it can learn to select features that contribute greatly to the performance of the refinement. Secondly, it can ease the optimization to fuse low-dimensionality point features with high-dimensionality image features. Then, we project the graph node to the location in the camera image and collect the feature vector at that pixel in the camera image through bilinear interpolation. Lastly, the features will be appended to s_j^0 for each node v_j.

3.4 Loss Functions

Classification Loss. For class-agnostic confidence score prediction, we follow [24,42] to use a score target I_i guided by the box's 3D IoU with the corresponding ground-truth bounding box:

$$I_i = \min\left(1, \max\left(0, 2 \times \mathrm{IoU}_i - 0.5\right)\right), \tag{4}$$

where IoU_i is the IoU between the i-th proposal box and the ground truth. The training is supervised with a binary cross entropy loss:

$$\mathcal{L}_{cls} = \frac{1}{B}\sum_{i=1}^{B} -I_i \log\left(\hat{I}_i\right) - (1 - I_i)\log\left(1 - \hat{I}_i\right), \tag{5}$$

where $\hat{I}_i$ is the predicted confidence score, and B is the number of sampled region proposals at the training stage.

Regression Loss. For box prediction, we transform the 3D proposal $b_i = (x_i, y_i, z_i, l_i, w_i, h_i, \theta_i)$ and the corresponding 3D ground-truth bounding box $b_i^{gt} = (x_i^{gt}, y_i^{gt}, z_i^{gt}, l_i^{gt}, w_i^{gt}, h_i^{gt}, \theta_i^{gt})$ from the global reference frame to the canonical coordinate system of 3D proposal:

$$\begin{aligned} \tilde{b}_i &= (0, 0, 0, l_i, w_i, h_i, 0), \\ \tilde{b}_i^{gt} &= \left(x_i^{gt} - x_i, y_i^{gt} - y_i, z_i^{gt} - z_i, l_i^{gt}, w_i^{gt}, h_i^{gt}, \Delta\theta_i\right), \end{aligned} \tag{6}$$

where $\Delta\theta_i = \theta_i^{gt} - \theta_i$. Then, the regression targets for center t_i^c, size t_i^s, and orientation t_i^o can be defined as:

$$\begin{aligned} t_i^c &= \left(x_i^{gt} - x_i, y_i^{gt} - y_i, z_i^{gt} - z_i\right), \\ t_i^s &= \left(l_i^{gt} - l_i, w_i^{gt} - w_i, h_i^{gt} - h_i\right), \\ t_i^o &= \Delta\theta_i - \lfloor \frac{\Delta\theta_i}{\pi} + 0.5 \rfloor \times \pi. \end{aligned} \tag{7}$$

Having all the targets $t_i = (t_i^c, t_i^s, t_i^o)$, our regression loss is defined as:

$$\mathcal{L}_{reg} = \frac{1}{B_+}\sum_{i=1}^{B_+} \mathrm{L1}\left(o_i - t_i\right), \tag{8}$$

where o_i is the output of the model's regression branch, and B_+ is the number of positive samples.

Total Loss. Finally, the overall loss is formulated as:

$$\mathcal{L} = \mathcal{L}_{cls} + \alpha \mathcal{L}_{reg}, \tag{9}$$

where α is a hyperparameter to balance the loss, which is 1 by default.

4 Experiments

4.1 Datasets

Waymo Open Dataset. is a large-scale autonomous driving dataset consisting of 798 scenes for training and 202 scenes for validation. The evaluation protocol consists of average precision (AP) and average precision weighted by heading (APH). It includes two difficulty levels: LEVEL_1 denotes objects containing more than 5 points, and LEVEL_2 denotes objects containing at least 1 point.

KITTI. contains 7481 training samples and 7518 testing samples in autonomous driving scenes. We follow [4,7,24] to divide the training data into a *train* set with 3712 samples and a *val* set with 3769 samples. The performance on the *val* set and the test leaderboard are reported.

4.2 Implementation Settings

Implementation Details. The codebase of CenterPoint is used for WOD. Then, we replace the second stage of CenterPoint-Voxel with our method (i.e., Graph-Ce) and train the network separately. For the dynamic point aggregation, we sample 256 points for each proposal and set $\sigma = 0.4$. In dynamic farthest voxel sampling, we set hash size as 4099 and λ and δ as 0.18 and 50, respectively. In patch search, the K and patch size are set to 32 and 1.0, respectively. For the RoI-graph pooling, we set $r = 0.4$ and the embedding channels to $[16, 16]$ in PointNet. We update the graph with $T = 3$, and the output dimensions of the three iterations are $[32, 32, 64]$. The number k of nearest neighbors is set as 8. In MLAF, the embedding dimension is 256, and the dropout ratio is 0.1.

For KITTI, the codebase of OpenPCDet is used. We propose Graph-Pi, Graph-Vo, and Graph-Po that use the **pi**llar-based PointPillars, the **vo**xel-based SECOND, and the **po**int-based 3DSSD as their region proposal networks, respectively. We incorporate the image branch in Graph-Vo (i.e., Graph-VoI) to compare with previous multi-modality methods. For 2D detector, we use the CenterNet [48] with DLA-34 [44] backbone, which takes images with a resolution of 1280×384 as input. The dimension of the output features will be reduced to 32 by the feature reduction layer.

Training Details. For WOD, we use the same training schedules and assignment strategies as CenterPoint-Voxel. The second stage is trained for 6 epochs on 4 GTX 1080Ti GPUs with 8 batch size per GPU.

Table 1. Vehicle detection results on WOD validation sequences. CenterPoint-Voxel† is reproduced by us based on the officially released code. CenterPoint-Voxel‡ is the first stage of CenterPoint-Voxel†.

Difficulty	Methods	3D AP (IoU=0.7)				3D APH (IoU=0.7)				BEV AP (IoU=0.7)				BEV APH (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	MVF [49]	62.93	86.30	60.02	36.02	-	-	-	-	80.40	93.59	79.21	63.09	-	-	-	-
	Pillar-od [32]	69.80	88.53	66.50	42.93	-	-	-	-	87.11	95.78	84.74	72.12	-	-	-	-
	PV-RCNN [24]	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
	VoTr-TSD [18]	74.95	92.28	73.36	51.09	74.25	91.73	72.56	50.01	-	-	-	-	-	-	-	-
	Voxel R-CNN [7]	75.59	92.49	74.09	53.15	-	-	-	-	88.19	**97.62**	87.34	77.70	-	-	-	-
	LiDAR R-CNN [14]	76.00	92.10	74.60	54.50	75.50	91.60	74.10	53.40	90.10	97.00	89.50	78.90	89.30	96.50	88.60	77.40
	Pyramid-PV [17]	76.30	92.67	74.91	54.54	75.68	92.20	74.21	53.45	-	-	-	-	-	-	-	-
	CenterPoint-Voxel† [42]	76.86	92.27	75.31	54.10	76.33	91.81	74.74	53.35	91.61	97.19	91.05	82.06	90.85	96.69	90.23	80.59
	CenterPoint-Voxel‡ [42]	74.78	91.51	73.25	50.67	74.24	91.04	72.67	49.93	90.94	97.02	90.26	80.78	90.12	96.51	89.39	79.20
	Graph-Ce (Ours)	**80.77**	**93.59**	**79.68**	**60.41**	**80.28**	**93.20**	**79.16**	**59.62**	**92.69**	97.56	**92.15**	**84.31**	**92.01**	**97.15**	**91.43**	**82.94**
LEVEL 2	PV-RCNN [24]	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
	VoTr-TSD [18]	65.91	-	-	-	65.29	-	-	-	-	-	-	-	-	-	-	-
	Voxel R-CNN [7]	66.59	91.74	67.89	40.80	-	-	-	-	81.07	**96.99**	81.37	63.26	-	-	-	-
	Pyramid-PV [17]	67.23	-	-	-	66.68	-	-	-	-	-	-	-	-	-	-	-
	LiDAR R-CNN [14]	68.30	91.30	68.50	42.40	67.90	90.90	68.00	41.80	81.70	94.30	82.30	65.80	81.00	93.90	81.50	64.50
	CenterPoint-Voxel† [42]	69.09	91.41	69.43	42.40	68.59	90.96	68.89	41.78	85.43	96.35	86.44	70.06	84.66	95.86	85.63	68.66
	CenterPoint-Voxel‡ [42]	66.66	90.63	66.90	39.50	66.17	90.16	66.36	38.90	84.87	96.21	85.69	69.08	84.04	95.69	84.81	67.58
	Graph-Ce (Ours)	**72.55**	**92.75**	**73.74**	**47.84**	**72.10**	**92.36**	**73.25**	**47.19**	**86.56**	96.79	**87.59**	**72.06**	**85.86**	**96.38**	**86.86**	**70.72**

Table 2. Vehicle, pedestrian, and cyclist results on WOD validation sequences.

Difficulty	Methods	Vehicle				Pedestrian				Cyclist			
		3D AP	3D APH	BEV AP	BEV APH	3D AP	3D APH	BEV AP	BEV APH	3D AP	3D APH	BEV AP	BEV APH
LEVEL 1	CenterPoint-Voxel‡ [42]	74.78	74.24	90.94	90.12	75.95	69.75	82.01	75.05	72.27	71.12	75.95	74.70
	Graph-Ce (Ours)	**80.77**	**80.28**	**92.69**	**92.01**	**82.35**	**76.64**	**86.75**	**80.51**	**75.28**	**74.21**	**77.42**	**76.30**
LEVEL 2	CenterPoint-Voxel‡ [42]	66.66	66.17	84.87	84.04	68.42	62.67	75.06	68.46	69.69	68.59	73.24	72.03
	Graph-Ce (Ours)	**72.55**	**72.10**	**86.56**	**85.86**	**74.44**	**69.02**	**79.50**	**73.45**	**72.52**	**71.49**	**74.64**	**73.56**

For KITTI, we use the same training configuration as PV-RCNN and train the whole model end-to-end for 80 epochs on 4 GTX 1080Ti GPUs with 4 batch size per GPU, and the pretrained RPN and 2D detector are frozen during training. For 2D detector, we pretrain CenterNet on WOD for 24 epochs and finetune it on KITTI for 12 epochs. We use Adam optimizer with one-cycle policy and set batch size to 2 and learning rate to 0.00025.

4.3 Comparison with State-of-the-Art Methods

Waymo Open Dataset. We compare Graph-Ce for the vehicle class at different distances on the full WOD validation with previous methods. Table 1 shows that Graph-Ce achieves the state-of-the-art results in both level 1 and level 2 among all the published papers with a single frame LiDAR input. In Table 2, we present our results for the vehicle, pedestrian, and cyclist classes. Compared with our baseline, i.e., CenterPoint-Voxel‡, our method improves the 3D APH in level 2 for the vehicle, pedestrian, and cyclist by 5.93%, 6.35%, and 2.90%, respectively.

KITTI. We compare Graph-Pi, Graph-Vo, Graph-Po, and Graph-VoI with previous methods listed in Table 3. Graph-Pi achieves the fastest inference speed among all two-stage methods. Compared with methods using only LiDAR as input, Graph-Po ranks the 1^{st} place in 3D AP and BEV AP with competitive inference speed. Graph-VoI outperforms all previous multi-modality methods by a large margin (+2.08% for 3D easy AP and +2.6% for 3D moderate AP). Table 4 shows our method could consistently improve PointPillars, SECOND, and 3DSSD by a large margin, demonstrating the efficacy of the method.

Table 3. Performance comparison on the KITTI testing sever for 3D car detection. L and I represent the LiDAR point cloud and the camera image, respectively.

Methods	Modality	3D AP			BEV AP			FPS
		Easy	Moderate	Hard	Easy	Moderate	Hard	(Hz)
One-stage:								
Point-GNN [27]	L	88.33	79.47	72.29	93.11	89.17	83.90	1.7
3DSSD [39]	L	88.36	79.57	74.55	92.66	89.02	85.86	26.3
SA-SSD [10]	L	88.75	79.79	74.16	95.03	91.03	85.96	25.0
CIA-SSD [46]	L	89.59	80.28	72.87	93.74	89.84	82.39	**32.5**
SASA [3]	L	88.76	82.16	77.16	92.87	89.51	86.35	27.8
Two-stage:								
PV-RCNN [24]	L	90.25	81.43	76.82	94.98	90.65	86.14	12.5
Voxel R-CNN [7]	L	90.90	81.62	77.06	94.85	88.83	86.13	25.2
CT3D [23]	L	87.83	81.77	77.16	92.36	88.83	84.07	14.3
Pyramid-PV [17]	L	88.39	82.08	77.49	92.19	88.84	86.21	7.9
VoTr-TSD [18]	L	89.90	82.09	**79.14**	94.03	90.34	86.14	7.2
SPG [37]	L	90.50	82.13	78.90	94.33	88.70	85.98	6.4
PointPainting [29]	L+I	82.11	71.70	67.08	92.45	88.11	83.36	2.5
PI-RCNN [36]	L+I	84.37	74.82	70.03	91.44	85.81	81.00	10.0
MMF [15]	L+I	88.40	77.43	70.22	93.67	88.21	81.99	12.5
EPNet [11]	L+I	89.81	79.28	74.59	94.22	88.47	83.69	10.0
3D-CVF [43]	L+I	89.20	80.05	73.11	93.52	89.56	82.45	13.3
CLOCs_PVCas [20]	L+I	88.94	80.67	77.15	93.05	89.80	86.57	10.0
Graph-Pi (Ours)	L	90.94	82.42	77.00	95.06	91.52	86.42	**28.5**
Graph-Vo (Ours)	L	91.29	82.77	77.20	95.27	91.72	86.51	25.6
Graph-Po (Ours)	L	91.79	83.18	77.98	**95.79**	**92.12**	**87.11**	16.1
Graph-VoI (Ours)	L+I	**91.89**	**83.27**	77.78	95.69	90.10	86.85	13.3

Table 4. Performance of our model on the KITTI *val* set with AP calculated by 40 recall positions for car class. † indicates our reproduced results.

Methods	3D AP			BEV AP		
	Easy	Moderate	Hard	Easy	Moderate	Hard
Pointpillars† (**Pi**llar-based) [12]	89.67	80.38	78.80	93.56	89.53	88.57
Graph-Pi (Ours)	93.16	85.87	83.29	96.18	91.84	89.46
SECOND† (**Vo**xel-based) [38]	92.15	82.43	79.26	95.78	91.26	88.57
Graph-Vo (Ours)	93.33	86.12	83.29	96.35	92.16	91.54
Graph-VoI (Ours)	**95.67**	**86.87**	**84.09**	96.28	**92.68**	**92.11**
3DSSD† (**Po**int-based) [39]	91.68	82.72	79.74	96.04	91.45	88.89
Graph-Po (Ours)	93.27	86.50	83.87	**96.64**	92.45	89.92

4.4 Ablation Study

Analysis of the Dynamic Point Aggregation. The third and fourth rows in Table 5 show that the dynamic point aggregation (DPA) contributes an improvement of 1.06% and 0.87% APH at level 2 for vehicle and pedestrian, respectively. Especially, Table 6 shows that DPA improves the baseline by 1.49% APH at 0-30m since the nearby objects suffer more from the uneven distribution problem. The first and second rows in Table 8 show that the patch search (PS) is 35× faster than the baseline, i.e., point cloud region pooling (PR) [25].

We also explore several sampling strategies in Table 8 to solve the uneven distribution problem, i.e., voxel sampling (VS), dynamic voxel sampling (DVS), dynamic farthest voxel sampling (DFVS), and farthest point sampling (FPS). We note that VS is a special case of DVS when δ is large, and FPS is a special case of DFVS when λ is small. Table 8 shows that DFVS achieves the best trade-off between accuracy and efficiency. Further, we explore the use of DFVS on point-based 3D object detectors as an alternative to FPS. Table 7 shows that DFVS achieves similar results with FPS but costs less runtime.

Table 5. Ablation study of every module: RoI-graph pooling (RGP), multi-level attentive fusion (MLAF), dynamic point aggregation (DPA), PointNet (PN), and diagonal corners (DC). We show the 3D APH at level 2 on the WOD validation set.

w/ RGP	w/ MLAF	w/ DPA	w/ PN	w/ DC	Vehicle	Pedestrian
					66.17	62.67
✓					69.48	66.79
✓	✓				69.77	67.00
✓	✓	✓			70.83	67.87
✓	✓	✓	✓		71.04	67.96
✓	✓	✓	✓	✓	**72.10**	**69.02**

Table 6. Ablation study of the performance of DPA at different distances. We show the level 2 APH on the WOD validation set for vehicle class.

w/ DPA	Overall	0-30m	30-50m	50m-Inf
	69.77	90.25	71.21	**45.40**
✓	**70.83**	**91.74**	**71.36**	**45.40**

Table 7. Ablation study of FPS and DFVS based on 3DSSD. We report the mAP on KITTI *val* set for car class and the runtime of sampling.

Methods	3D mAP	BEV mAP	Runtime (ms)
FPS	81.73	**88.57**	29.6
DFVS	**81.75**	88.55	**20.3**

Table 8. Ablation study of different sampling and searching strategies. † is tested by us based on the officially released code.

w/ PR†	w/ PS	w/ VS	w/ DVS	w/ DFVS	w/ FPS	APH	Runtime (ms)
✓						69.77	69.7
	✓					69.77	**1.9**
	✓	✓				70.39	2.2
	✓		✓			70.67	2.2
	✓			✓		**70.83**	2.8
	✓				✓	**70.83**	7.3

Analysis of the RoI-graph Pooling. The first and second rows in Table 5 show that the RoI-graph pooling (RGP) raises the APH at level 2 for vehicle and pedestrian by 3.31% and 4.12%, respectively. In Table 9, we study the effect of the number of iterations on the detection accuracy, where the number of neighbors is set to 8 by default to save GPU memory. This result suggests it is beneficial to iterate more times to mine geometric relations. Besides, we find accuracy gains for distant objects are greater than for nearby objects since the contextual information is better modeled to alleviate the missing detection of distant objects. For graph construction, we investigate the components used in the initial state of the graph node. The fourth and fifth rows in Table 5 show that using PointNet (PN) raises 0.21% APH for vehicle class since the downsampling introduces the loss of fine-grained details, and the fifth and sixth rows show that adding two diagonal corners (DC) for each node raises 1.06% APH for both vehicle and pedestrian. For graph aggregation, the second and third rows in Table 5 show that multi-level attentive fusion (MLAF) boosts the performance by 0.29% APH for vehicle class. Furthermore, we study influences of different aggregation methods in Table 10, i.e., channel-wise Transformer (CT), Set Transformer (ST), attention sum (AS), average pooling (AP), and max pooling (MP). Transformer does not achieve better results, probably because it has more parameters leading to overfitting.

Table 9. Ablation study of the number of iterations to update the graph.

# iterations	Overall	0-30m	30-50m	50m-Inf
T=1	68.76	89.84	70.19	44.03
T=2	69.07	89.95	70.60	44.49
T=3	**69.48**	**89.97**	**71.02**	**45.18**

Table 10. Ablation study of different methods to aggregate nodes' features.

Methods	CT	ST	AS	AP	MP
Vehicle	71.67	71.82	71.86	71.85	**72.10**
Pedestrian	68.82	68.74	68.66	68.76	**69.02**

Table 11. Ablation study of image features and CPA. † and ‡ indicate CPA is used in RPN and refinement, respectively.

w/ CPA†	w/ CPA‡	w/ RGB	w/ Seg.	w/ Feat.	3D AP	BEV AP
					85.38	91.48
✓	✓				86.12	92.16
✓					86.11	92.23
✓		✓			86.20	92.35
✓			✓		86.38	92.59
✓				✓	**86.87**	**92.68**

Analysis of the Visual Features Augmentation. By comparing the fourth and fifth rows in Table 4, we observe that the image feature raises the detection results by 2.34%, 0.75%, and 0.8% 3D AP respectively in terms of easy, moderate, and hard. We carefully analyze the effect of the cut-and-paste augmentation (CPA) at different stages in Table 11. We find that the refinement stage is hardly affected by CPA. Thus, we can conveniently train the LiDAR branch and the image branch end-to-end without the help of CPA. We also provide the study of different image features, i.e., the RGB of the input image, the segmentation scores, and the output features of the 2D detector. We find that using the 2D features achieves the best results.

5 Conclusions

We present an accurate and efficient 3D object detector Graph R-CNN that can be applied to existing 3D detectors. Our framework can handle the unevenly distributed and sparse point clouds by utilizing the dynamic point aggregation and the semantic-decorated local graph.

Acknowledgments. This work was supported in part by The National Key Research and Development Program of China (Grant Nos: 2018AAA0101400), in part by The National Nature Science Foundation of China (Grant Nos: 62036009, U1909203, 61936006, 62133013), in part by Innovation Capability Support Program of Shaanxi (Program No. 2021TD-05).

References

1. Bewley, A., Sun, P., Mensink, T., Anguelov, D., Sminchisescu, C.: Range conditioned dilated convolutions for scale invariant 3d object detection. In: Conference on Robot Learning (2020)
2. Chai, Y., Sun, P., Ngiam, J., Wang, W., Caine, B., Vasudevan, V., Zhang, X., Anguelov, D.: To the point: Efficient 3d object detection in the range image with graph convolution kernels (2021)
3. Chen, C., Chen, Z., Zhang, J., Tao, D.: SASA: semantics-augmented set abstraction for point-based 3d object detection. In: Proceedings of the AAAI Conference on Artificial Intelligence (2022)
4. Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3d object detection network for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017)
5. Chen, Y., Liu, S., Shen, X., Jia, J.: Fast point r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision (2019)
6. Cheng, B., Sheng, L., Shi, S., Yang, M., Xu, D.: Back-tracing representative points for voting-based 3d object detection in point clouds (2021)
7. Deng, J., Shi, S., Li, P., Zhou, W., Zhang, Y., Li, H.: Voxel r-cnn: Towards high performance voxel-based 3d object detection. In: Proceedings of the AAAI Conference on Artificial Intelligence (2021)
8. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the KITTI vision benchmark suite. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2012)
9. Gori, M., Monfardini, G., Scarselli, F.: A new model for learning in graph domains. In: Proceedings of the IEEE International Joint Conference on Neural Networks (2005)
10. He, C., Zeng, H., Huang, J., Hua, X.S., Zhang, L.: Structure aware single-stage 3d object detection from point cloud. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020)
11. Huang, T., Liu, Z., Chen, X., Bai, X.: Epnet: Enhancing point features with image semantics for 3d object detection. In: Proceedings of the European Conference on Computer Vision (2020)
12. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2019)
13. Lee, J., Lee, Y., Kim, J., Kosiorek, A.R., Choi, S., Teh, Y.W.: Set transformer: A framework for attention-based permutation-invariant neural networks. In: Proceedings of the International Conference on Machine Learning (2019)
14. Li, Z., Wang, F., Wang, N.: Lidar r-cnn: An efficient and universal 3d object detector. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2021)
15. Liang, M., Yang, B., Chen, Y., Hu, R., Urtasun, R.: Multi-task multi-sensor fusion for 3d object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2019)
16. Liu, Z., Xu, G., Yang, H., Liu, H., Cai, D.: Sparsepoint: Fully end-to-end sparse 3d object detector. CoRR **abs/2103.10042** (2021)
17. Mao, J., Niu, M., Bai, H., Liang, X., Xu, H., Xu, C.: Pyramid r-cnn: Towards better performance and adaptability for 3d object detection. In: Proceedings of the IEEE International Conference on Computer Vision (2021)

18. Mao, J., Xue, Y., Niu, M., Bai, H., Feng, J., Liang, X., Xu, H., Xu, C.: Voxel transformer for 3d object detection. In: Proceedings of the IEEE International Conference on Computer Vision (2021)
19. Najibi, M., Lai, G., Kundu, A., Lu, Z., Rathod, V., Funkhouser, T.A., Pantofaru, C., Ross, D.A., Davis, L.S., Fathi, A.: DOPS: learning to detect 3d objects and predict their 3d shapes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020)
20. Pang, S., Morris, D.D., Radha, H.: Clocs: Camera-lidar object candidates fusion for 3d object detection. In: International Conference on Intelligent Robots and Systems (2020)
21. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2017)
22. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in Neural Information Processing Systems (2017)
23. Sheng, H., Cai, S., Liu, Y., Deng, B., Huang, J., Hua, X.S., Zhao, M.J.: Improving 3d object detection with channel-wise transformer. In: Proceedings of the IEEE International Conference on Computer Vision (2021)
24. Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., Li, H.: Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020)
25. Shi, S., Wang, X., Li, H.: Pointrcnn: 3d object proposal generation and detection from point cloud. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2019)
26. Shi, S., Wang, Z., Shi, J., Wang, X., Li, H.: 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)
27. Shi, W., Rajkumar, R.: Point-gnn: Graph neural network for 3d object detection in a point cloud. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020)
28. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., Anguelov, D.: Scalability in perception for autonomous driving: Waymo open dataset. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020)
29. Vora, S., Lang, A.H., Helou, B., Beijbom, O.: Pointpainting: Sequential fusion for 3d object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020)
30. Wang, C., Ma, C., Zhu, M., Yang, X.: Pointaugmenting: Cross-modal augmentation for 3d object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2021)
31. Wang, J., Lan, S., Gao, M., Davis, L.S.: Infofocus: 3d object detection for autonomous driving with dynamic information modeling. In: Proceedings of the European Conference on Computer Vision (2020)
32. Wang, Y., Fathi, A., Kundu, A., Ross, D.A., Pantofaru, C., Funkhouser, T.A., Solomon, J.: Pillar-based object detection for autonomous driving. In: Proceedings of the European Conference on Computer Vision (2020)
33. Wang, Y., Solomon, J.: Object DGCNN: 3d object detection using dynamic graphs. In: Advances in Neural Information Processing Systems (2021)

34. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph CNN for learning on point clouds. ACM Trans. Graph. (2019)
35. Wu, X., Peng, L., Yang, H., Xie, L., Huang, C., Deng, C., Liu, H., Cai, D.: Sparse fuse dense: Towards high quality 3d detection with depth completion. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2022)
36. Xie, L., Xiang, C., Yu, Z., Xu, G., Yang, Z., Cai, D., He, X.: PI-RCNN: an efficient multi-sensor 3d object detector with point-based attentive cont-conv fusion module. In: Proceedings of the AAAI Conference on Artificial Intelligence (2020)
37. Xu, Q., Zhou, Y., Wang, W., Qi, C.R., Anguelov, D.: SPG: unsupervised domain adaptation for 3d object detection via semantic point generation (2021)
38. Yan, Y., Mao, Y., Li, B.: Second: Sparsely embedded convolutional detection. Sensors **18**(10) (2018)
39. Yang, Z., Sun, Y., Liu, S., Jia, J.: 3dssd: Point-based 3d single stage object detector. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020)
40. Yang, Z., Sun, Y., Liu, S., Shen, X., Jia, J.: Std: Sparse-to-dense 3d object detector for point cloud. In: Proceedings of the IEEE International Conference on Computer Vision (2019)
41. Yin, J., Shen, J., Guan, C., Zhou, D., Yang, R.: Lidar-based online 3d video object detection with graph-based message passing and spatiotemporal transformer attention. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2020)
42. Yin, T., Zhou, X., Krähenbühl, P.: Center-based 3d object detection and tracking. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2021)
43. Yoo, J.H., Kim, Y., Kim, J.S., Choi, J.W.: 3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object detection. In: Proceedings of the European Conference on Computer Vision (2020)
44. Yu, F., Wang, D., Shelhamer, E., Darrell, T.: Deep layer aggregation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2018)
45. Zhang, W., Wang, Z., Loy, C.C.: Multi-modality cut and paste for 3d object detection. CoRR **abs/2012.12741** (2020)
46. Zheng, W., Tang, W., Chen, S., Jiang, L., Fu, C.W.: Cia-ssd: Confident iou-aware single-stage object detector from point cloud. In: Proceedings of the AAAI Conference on Artificial Intelligence (2021)
47. Zheng, W., Tang, W., Jiang, L., Fu, C.W.: Se-ssd: Self-ensembling single-stage object detector from point cloud. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2021)
48. Zhou, X., Wang, D., Krähenbühl, P.: Objects as points. CoRR **abs/1904.07850** (2019)
49. Zhou, Y., Sun, P., Zhang, Y., Anguelov, D., Gao, J., Ouyang, T., Guo, J., Ngiam, J., Vasudevan, V.: End-to-end multi-view fusion for 3d object detection in lidar point clouds. In: Conference on Robot Learning (2019)
50. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2018)