Improving 3D Object Detection with Channel-wise Transformer

Hualian Sheng[1,2]* Sijia Cai[2] Yuan Liu[2] Bing Deng[2]
Jianqiang Huang[2] Xian-Sheng Hua[2] Min-Jian Zhao[1]†
[1] College of Information Science and Electronic Engineering, Zhejiang University
[2]DAMO Academy, Alibaba Group
hlsheng@zju.edu.cn, {stephen.csj, alen.ly, dengbing.db}@alibaba-inc.com
jianqiang.jqh@gmail.com, xiansheng.hxs@alibaba-inc.com, mjzhao@zju.edu.cn

Abstract

Though 3D object detection from point clouds has achieved rapid progress in recent years, the lack of flexible and high-performance proposal refinement remains a great hurdle for existing state-of-the-art two-stage detectors. Previous works on refining 3D proposals have relied on human-designed components such as keypoints sampling, set abstraction and multi-scale feature fusion to produce powerful 3D object representations. Such methods, however, have limited ability to capture rich contextual dependencies among points. In this paper, we leverage the high-quality region proposal network and a Channel-wise Transformer architecture to constitute our two-stage 3D object detection framework (CT3D) with minimal hand-crafted design. The proposed CT3D simultaneously performs proposal-aware embedding and channel-wise context aggregation for the point features within each proposal. Specifically, CT3D uses proposal's keypoints for spatial contextual modelling and learns attention propagation in the encoding module, mapping the proposal to point embeddings. Next, a new channel-wise decoding module enriches the query-key interaction via channel-wise re-weighting to effectively merge multi-level contexts, which contributes to more accurate object predictions. Extensive experiments demonstrate that our CT3D method has superior performance and excellent scalability. Remarkably, CT3D achieves the AP of 81.77% in the moderate car category on the KITTI test 3D detection benchmark, outperforms state-of-the-art 3D detectors.

1. Introduction

3D object detection from point clouds is envisioned as an indispensable part of future Autonomous Vehicle (AV). Unlike the developed 2D detection algorithms whose success is mainly due to the regular structure of image pixels, LiDAR point clouds are usually sparse, unordered and unevenly distributed. This makes the CNN-like operations not well suited to process unstructured point clouds directly. To tackle these challenges, many approaches employ voxelization or custom discretization for point clouds. Several methods [28, 15] project point clouds to a birds-eye view (BEV) representation and apply the standard 2D convolutions, however, it will inevitably sacrifice certain geometric details which are vital for generating accurate localization. Other methods [3, 33] rasterize point clouds into a 3D voxel grid and use regular 3D CNNs to perform computation in grid space, but this category of methods suffers from computational bottleneck associated with making the grid finer. A major breakthrough in detection task on point clouds is due to the effective deep architectures for point clouds representation such as volumetric convolution [33] and permutation invariant convolution [22].

Recently, most state-of-the-art methods for 3D object detection adopt a two-stage framework consisting of 3D region proposal generation and proposal feature refinement. Notice that the most popular region proposal network (RPN) backbone [33] has achieved over 95% recall rate on the KITTI 3D Detection Benchmark, whereas this method only achieves 78% Average Precision (AP). The reason for such a gap stems from the difficulty in encoding an object and extracting the robust feature from 3D proposals in cases of occlusion or long-range distance. Therefore, how to effectively model geometric relationships among points and exploit accurate position information during the proposal feature refinement stage is crucial for good performance. An important family of models is PointNet [22] and its variants [23, 19, 25], which use a flexible receptive field to aggregate features by local regions and permutation-invariant network. However, these methods have the drawback of involving plenty of hand-crafted designs, such as the neighbor ball radii and the grid size. Another family of models

[*] This work was done when the author was visiting Alibaba as a research intern. The code is available at https://github.com/hlsheng1/CT3D

[†] Corresponding author.

is the voxel-based methods [33, 27, 39] which use 3D convolutional kernels to gather information from neighboring voxels. But the performance of such methods is not optimal caused by the voxel quantization and sensitive to hyperparameters. Later studies [43, 24, 4, 10] further apply the point-voxel mixed strategy to capture multi-scale features while retaining fine-grained localization but are strongly tied to the specific RPN architectures.

In this paper, we make two major contributions. First, we propose a novel end-to-end two-stage 3D objection detection framework called CT3D. Motivated by the recent Transformer-based 2D detection method DETR [1] that uses CNN backbone to extract features and encoder-decoder Transformer to enhance the RoI region features, we design our CT3D to generate 3D bounding boxes at the first stage, then learn per-proposal representation by incorporating a novel Transformer architecture with channel-wise reweighting mechanism in decoder. The proposed framework exhibits very strong performance in terms of accuracy and efficiency, and thus can be conveniently combined with any high-quality RPN backbones.

The second contribution is the custom Transformer that offers several benefits over the traditional point/voxel-based feature aggregation mechanism. Despite the point-wise or voxel convolutions have the ability of local and global context modelling, there still have been several limitations in increasing receptive field and parameter optimization. In addition, point-cloud based 3D object detectors also have to deal with the challenging missing/noisy detections such as occlusion and distancing patterns with a few points. Self-attention in Transformers has recently emerged as a basic building block for capturing long-range interactions thus is a natural choice in acquiring context information for enriching the faraway objects or increasing the confidence of false negatives. Inspired by this idea, we initially introduce a proposal-to-point embedding to effectively encode the RPN proposal information in the encoder module. Furthermore, we exploit a channel-wise re-weighting approach to augment the standard Transformer decoder in consideration of both global and local channel-wise features for the encoded points. The purpose is to scale the feature decoding space where we can compute attention distribution over each channel dimension of key embeddings thus can enhance the expressiveness of query-key interactions. Extensive experiments show that our proposed CT3D can outperform the state-of-the-art published methods on both the KITTI dataset and the large-scale Waymo dataset.

2. Related Work

Point Cloud Representations for 3d Object Detection. Recently, there has been a lot of progress on learning effective representations for the raw LiDAR point clouds. A noticeable portion of efforts are PointNet series [22] which employed permutation invariant operations to aggregate the point features. F-PointNet [21] generated the region-level features for point clouds within each 3D frustum. PointRCNN [25] used PointNet++ [23] to segment foreground 3D points and refine the proposals with the segmentation features. STD [37] further extended the proposal refinement by transferring sparse point features into dense voxel representation. Moreover, 3DSSD [36] improved the point-based approach with a new sampling strategy based on feature distance. However, PointNet-like architectures still present limited ability to capture local structures for LiDAR data. Another category of methods [3, 13, 34, 35, 28, 15, 12, 16, 17] aimed to voxelize the unstructured point clouds as a regular 2D/3D grid over which conventional CNNs can be easily applied. Pioneer work [3] encoded the point clouds as 2D bird-view feature maps to generate highly accurate 3D candidate boxes, motivating many efficient bird-view representation-based methods. VoxelNet [43] transformed the points to form a compact feature representation. SECOND [33] introduced 3D sparse convolution for efficient 3D voxel processing. These voxel-based methods are still focused on the subdivision of a volume rather than adaptively modelling local geometric structure. Furthermore, various point-voxel based methods have been proposed for multi-scale feature aggregation. SA-SSD [10] presented an auxiliary network on the basis of 3D voxel CNN. PV-RCNN [24] and its variant VoxelR-CNN [4] adopted 3D voxel CNN as RPN to generate high-quality proposals and then utilize PointNet to aggregate the voxel features around the grids. Nevertheless, these hybrid methods require plenty of hand-crafted feature designs.

Transformers for object detection. A new paradigm for object detection has recently evolved due to the success of Transformers in many computer vision fields [1, 44, 5, 9, 6]. Since Transformer models are very effective at learning local context-aware representations, DETR [1] viewed the detection as a set prediction problem and employed Transformer with parallel decoding to detect objects in 2D image. A variant of DETR [44] further developed a deformable attention module to employ cross-scale aggregation. For point clouds, recent methods [9, 6] also explored to use self-attention for classification and segmentation tasks.

3. CT3D for 3D Object Detection

Given proposals generated by the widely used RPN backbones like 3D voxel CNN [33], current state-of-the-art proposal refinement approaches [24, 4] focus on refining the intermediate multi-stage voxel features extracted by the convolution layers, suffering the difficulties of extra hyperparameter optimization and designing generalized models. We believe that the raw points with precise position information are sufficient for refining the detection proposals. Bearing this view in mind, we construct our CT3D frame-

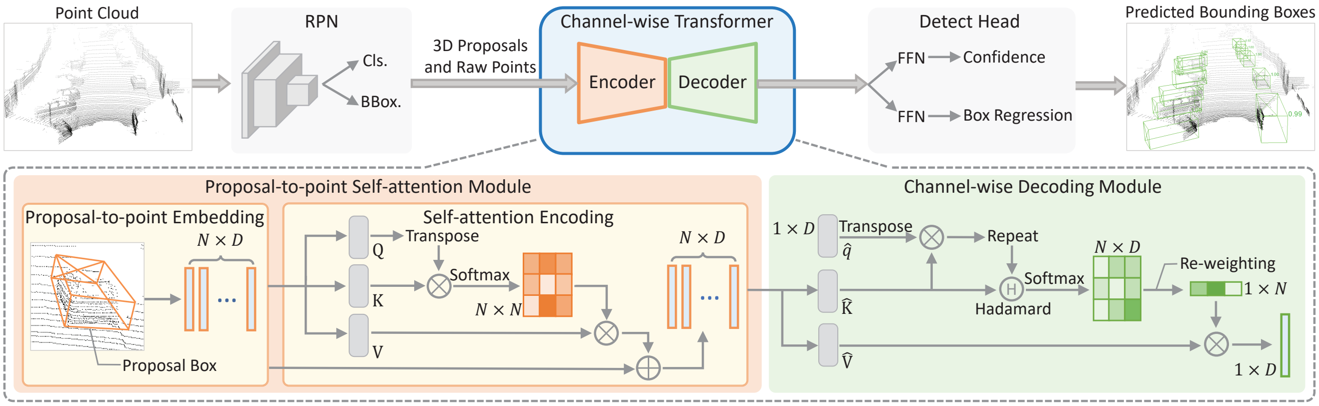

Figure 1. Overview of CT3D. The raw points are first fed into the RPN for generating 3D proposals. Then the raw points along with the corresponding proposals are processed by the channel-wise Transformer composed of the proposal-to-point encoding module and the channel-wise decoding module. Specifically, the proposal-to-point encoding module is to modulate each point feature with global proposal-aware context information. After that, the encoded point features are transformed into an effective proposal feature representation by the channel-wise decoding module for confidence prediction and box regression.

work by deploying a well-designed Transformer on top of a RPN network to directly utilize the raw point clouds. Specifically, the whole CT3D detection framework is composed of three parts, *i.e.,* a RPN backbone for proposal generation, a channel-wise Transformer for proposal feature refinement and a detect head for object predictions. Figure 1 illustrates an overview of our CT3D framework.

3.1. RPN for 3D Proposal Generation

Starting from the point clouds **P** with 3-dimension coordinates and C-dimension point features, the predicted 3D bounding box generated by RPN consists of center coordinate $\boldsymbol{p}^c = [x^c, y^c, z^c]$, length l^c, width w^c, height h^c, and orientation θ^c. In this paper, we adopt the 3D voxel CNN SECOND [33] as our default RPN due to its high efficiency and accuracy. Note that any high-quality RPN should be readily replaceable in our framework and is amenable to training via an end-to-end manner.

3.2. Proposal-to-point Encoding Module

To refine the generated RPN proposals, we adopt a two-step strategy. Specifically, the first proposal-to-point embedding step maps the proposal to point features, then the second self-attention encoding step is to refine point features via modelling the relative relationships among points within the corresponding proposal.

Proposal-to-point Embedding. Given the proposals generated by RPN, we delimit out a scaled RoI area in point clouds according to the proposal. This aims to compensate the deviation between the proposal and the corresponding ground-truth box by wrapping all object points as much as possible. Specifically, the scaled RoI area is a cylindrical with unlimited height and a radius $r = \alpha\sqrt{(\frac{l^c}{2})^2 + (\frac{w^c}{2})^2}$, where α is a hyper-parameter, and l, w denote the length and width of the proposal, respectively. Hereinafter, the randomly sampled $N = 256$ points within the scaled RoIs ($\mathcal{N} = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_N\}$) are taken out for further processing.

At first, we calculate the relative coordinates between each sampled point and the center point of the proposal for unifying the input distance feature, denoted as $\Delta\boldsymbol{p}_i^c = \boldsymbol{p}_i - \boldsymbol{p}^c, \forall \boldsymbol{p}_i \in \mathcal{N}$. A straightforward thought is to directly concatenate the proposal information into each point feature, *i.e.,* $[\Delta\boldsymbol{p}_i^c, l^c, w^c, h^c, \theta^c, f_i^r]$, where f_i^r is the raw point feature such as reflection. However, the size-orientation representation for proposal yields only modest performance as the Transformer encoder might be less effective to reorient in accord with above-mentioned geometric information.

It is noteworthy that the keypoints usually offer more explicit geometry property in detection tasks [41, 14], we propose a novel keypoints subtraction strategy to compute the relative coordinates between each point and the eight corner points of the corresponding proposal. The calculated relative coordinates are $\Delta\boldsymbol{p}_i^j = \boldsymbol{p}_i - \boldsymbol{p}^j, j = 1, \ldots, 8$, where $\boldsymbol{p}^j$ is the coordinate of the j-th corner point. Note that l^c, w^c, h^c and θ^c disappear but are contained in different dimensions of distance information. Through this way, the newly generated relative coordinates $\Delta\boldsymbol{p}_i^j$ can be viewed as a better representation of proposal information. As shown in the left part of Figure 2, for each point $\boldsymbol{p}_i$, the proposal-guided point feature can be expressed as:

$$\boldsymbol{f}_i = \mathcal{A}([\Delta\boldsymbol{p}_i^c, \Delta\boldsymbol{p}_i^1, \ldots, \Delta\boldsymbol{p}_i^8, f_i^r]) \in \mathbb{R}^D, \qquad (1)$$

where $\mathcal{A}(\cdot)$ is a linear projection layer to map point feature into a high-dimensional embedding.

Self-attention Encoding. The embeded point features are then fed into the multi-head self-attention layer, followed

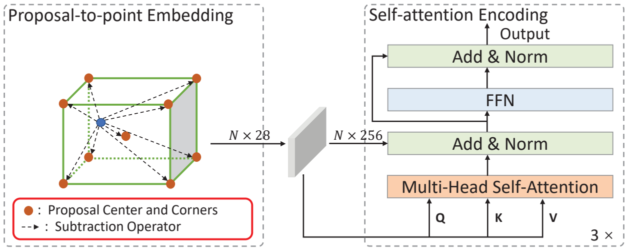
Proposal-to-point Embedding
Self-attention Encoding
Output
Add & Norm
FFN
Add & Norm
Multi-Head Self-Attention
N × 28
N × 256
Q
K
V
3 ×
● : Proposal Center and Corners
--▸ : Subtraction Operator

Figure 2. Proposal-to-point encoding. The location features of raw point clouds are first modulated by the proposal information (center and corners) via subtraction operator. Then, the resulting point features are refined by the proposal-aware encoding module with multi-head self-attention mechanism.

by a feed-forward network (FFN) with residual structure, to encode rich contextual relationships and point dependencies in proposal for refining point features. As shown in the right part of Figure 2, this self-attention encoding scheme shares almost the same structure as the original NLP Transformer encoder, except for the position embedding since it is already included in the point features. Reader can refer to [31] for more details. Denote $\mathbf{X} = [\boldsymbol{f}_1^T, \dots, \boldsymbol{f}_N^T]^T \in \mathbb{R}^{N\times D}$ as the embedded point features with the dimension D, we have $\mathbf{Q} = \mathbf{W}_q\mathbf{X}$; $\mathbf{K} = \mathbf{W}_k\mathbf{X}$; $\mathbf{V} = \mathbf{W}_v\mathbf{X}$, where $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{N\times N}$ are linear projections, and $\mathbf{Q}, \mathbf{K}$ and $\mathbf{V}$ are so-called query, key and value embeddings. These three embeddings are then processed by multi-head self-attention mechanism. In a H-head attention situation, $\mathbf{Q}, \mathbf{K}$ and $\mathbf{V}$ are further divided into $\mathbf{Q} = [\mathbf{Q}_1, \dots, \mathbf{Q}_H]$, $\mathbf{K} = [\mathbf{K}_1, \dots, \mathbf{K}_H]$, and $\mathbf{V} = [\mathbf{V}_1, \dots, \mathbf{V}_H]$, where $\mathbf{Q}_h, \mathbf{K}_h, \mathbf{V}_h \in \mathbb{R}^{N\times D'}, \forall h = 1, \dots, H$, and $D' = \frac{D}{H}$. The output after multi-head self-attention is given by:

$$\mathrm{S}^{(\mathrm{att})}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \left[\sigma\big(\frac{\mathbf{Q}_h\mathbf{K}_h^T}{\sqrt{D'}}\big) \cdot \mathbf{V}_h\right], h = 1, \dots, H, \quad (2)$$

where $\sigma(\cdot)$ is *softmax* function. Hereinafter, applying a simple FFN and residual operator, the result is as follows:

$$\mathrm{S}^{(\mathrm{emb})}(\mathbf{X}) = \mathcal{Z}(\mathcal{F}(\mathcal{Z}(\mathrm{S}^{(\mathrm{att})}(\mathbf{Q}, \mathbf{K}, \mathbf{V})))), \quad (3)$$

where $\mathcal{Z}(\cdot)$ denotes add and normalization operator, $\mathcal{F}(\cdot)$ denotes a FFN with two linear layers and one *Relu* activation. We observe that a stack of 3 identical self-attention encoding modules is ideal for our CT3D framework.

3.3. Channel-wise Decoding Module

In this subsection, we manage to decode all point features (*i.e.,* $\hat{\mathbf{X}}$) from the encoder module into a global representation, which is further processed by FFNs for the final detection predictions. Different from the standard Transformer decoder, which transforms M multiple query embeddings using self- and encoder-decoder attention mechanism, our decoder only manipulates one query embedding according to the following two facts:

- M query embeddings suffer high memory latency, especially for processing with numbers of proposals.
- M query embeddings are usually independently transformed into M words or objects, while our proposal refinement model only needs one prediction.

Generally, the final proposal representation after decoder can be regarded as a weighted sum of all point features, our key motivation is to determine the decoding weights that are dedicated for each point. In below, we first analyze the standard decoding scheme, and then develop an improved decoding scheme to acquire more effective decoding weights.

Standard Decoding. The standard decoding scheme utilizes a learnable vector (*i.e.,* query embedding) of dimension D to aggregate the point features across all channels. As shown in Figure 3(a), the final decoding weight vector for all point features in each attention head is:

$$\boldsymbol{w}_h^{(S)} = \sigma\big(\frac{\hat{\boldsymbol{q}}_h\hat{\mathbf{K}}_h^T}{\sqrt{D'}}\big), h = 1, \dots, H, \quad (4)$$

where $\hat{\mathbf{K}}_h$ is the key embeddings of h-th head computed by the projection of encoder output, and $\hat{\boldsymbol{q}}_h$ is the corresponding query embedding. Note that each value of vector $\hat{\boldsymbol{q}}_h\hat{\mathbf{K}}_h^T$ can be viewed as the global aggregation for individual point (*i.e.*, each key embedding), and the subsequent *softmax* function assigns the decoding value for each point according to the probability in the normalized vector. Consequently, the values in decoding weight vector are derived from simple global aggregation and lack the local channel-wise modelling, which is essential to learn 3D surface structures of point clouds because different channels usually exhibit strong geometric relationships in point clouds.

Channel-wise Re-weighting. In order to emphasize the channel-wise information for key embeddings $\hat{\mathbf{K}}_h^T$, a straightforward solution is to compute the decoding weight vector for points based on all the channels of $\hat{\mathbf{K}}_h^T$. That is, we generate D different decoding weight vectors for each channel to obtain D decoding values. Further, a linear projection is introduced for these D decoding values to form a united channel-wise decoding vector. As shown in Figure 3(b), this new channel-wise re-weighting for decoding weight vector can be summarized as:

$$\boldsymbol{w}_h^{(C)} = \boldsymbol{s} \cdot \hat{\sigma}\big(\frac{\hat{\mathbf{K}}_h^T}{\sqrt{D'}}\big), h = 1, \dots, H, \quad (5)$$

where $\boldsymbol{s}$ is a linear projection that compresses D' number of decoding values into a re-weighting scalar, $\hat{\sigma}(\cdot)$ computes the *softmax* along the N dimension. However, the decoding weights computed by $\hat{\sigma}(\cdot)$ are associated with each channel, and thus ignore the global aggregation of each point. Therefore, we can conclude that the standard decoding scheme

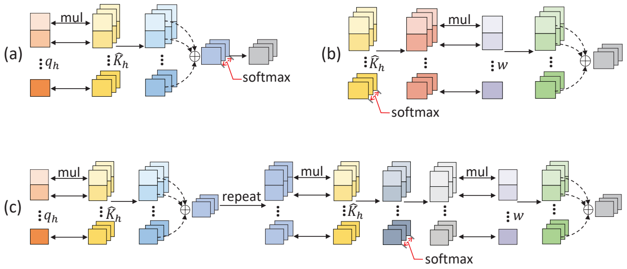

Figure 3. Illustration of the different decoding schemes: (a) Standard decoding; (b) Channel-wise re-weighting; (c) Extended channel-wise re-weighting.

focuses on global aggregation while the channel-wise re-weighting scheme concentrates on the channel-wise local aggregation. To combine their characteristics, we propose an extended channel-wise re-weighting scheme as below.

Extended Channel-wise Re-weighting. Specifically, we first repeat the matrix product of query embedding and key embeddings to spread the spatial information into each channel, and the output is then multiplied element-wise with the key embeddings for keeping the channel differences. As illustrated in Figure 3 (c), this novel extended channel-wise re-weighting scheme generates the following decoding weight vector for all the points:

$$\boldsymbol{w}_h^{(EC)} = \boldsymbol{s} \cdot \hat{\sigma}\big(\frac{\rho(\hat{\boldsymbol{q}}_h \hat{\mathbf{K}}_h^T) \odot \hat{\mathbf{K}}_h^T}{\sqrt{D'}}\big), h = 1, \dots, H, \tag{6}$$

where $\rho(\cdot)$ is a repeat operator makes $\mathbb{R}^{1\times N} \rightarrow \mathbb{R}^{D'\times N}$. In this way, we can not only maintain the global information as compared to the channel-wise re-weighting scheme, but also enrich the local and detailed channel interactions as compared to the standard decoding scheme. Besides, this extended channel-wise re-weighting only brings 1K+ (Bytes) increase as compared to the other two schemes. As a result, the final decoded proposal representation can be described as follows:

$$\boldsymbol{y} = [\boldsymbol{w}_1^{(EC)} \cdot \hat{\mathbf{V}}_1, \dots, \boldsymbol{w}_H^{(EC)} \cdot \hat{\mathbf{V}}_H], \tag{7}$$

where the value embeddings $\hat{\mathbf{V}}$ is the linear projection obtained from $\hat{\mathbf{X}}$.

3.4. Detect head and Training Targets

In the previous steps, the input point features are summarized into a D-dimension vector $\boldsymbol{y}$, which is then fed into two FFNs for predicting the confidence and the box residuals relative to the input 3D proposal, respectively.

To output the confidence, training targets are set as the 3D IoU between the 3D proposals and their corresponding ground-truth boxes. Given the IoU of the 3D proposal and its corresponding ground-truth box, we follow [11, 25, 24] to assign the confidence prediction target, which is shown as:

$$c^t = \min\Big(1, \max\big(0, \frac{\text{IoU} - \alpha_B}{\alpha_F - \alpha_B}\big)\Big), \tag{8}$$

where α_F and α_B are the foreground and background IoU thresholds, respectively. Besides, regression targets (superscript t) are encoded by proposals and their corresponding ground-truth boxes (superscript g), given by:

$$\begin{aligned} x^t &= \frac{x^g - x^c}{d}, y^t = \frac{y^g - y^c}{d}, z^t = \frac{z^g - z^c}{d}, \\ l^t &= \log\big(\frac{l^g}{l^c}\big), w^t = \log\big(\frac{w^g}{w^c}\big), h^t = \log\big(\frac{h^g}{h^c}\big), \\ \theta^t &= \theta^g - \theta^c, \end{aligned} \tag{9}$$

where $d = \sqrt{(l^c)^2 + (w^c)^2}$ is the diagonal of the base of the proposal box.

3.5. Training Losses

We adopt an end-to-end strategy to train CT3D. Hence, the overall training loss is the summation of the RPN loss, the confidence prediction loss, and the box regression loss, which is presented:

$$\mathcal{L} = \mathcal{L}_{\text{RPN}} + \mathcal{L}_{\text{conf}} + \mathcal{L}_{\text{reg}}. \tag{10}$$

Here, the binary cross entropy loss [11, 35] is exploited for the predicted confidence c to compute the IoU-guided confidence loss:

$$\mathcal{L}_{\text{conf}} = -c^t \log(c) - (1 - c^t) \log(1 - c). \tag{11}$$

Moreover, the box regression loss [35, 33] adopts:

$$\mathcal{L}_{\text{reg}} = \mathbb{I}(\text{IoU} \geq \alpha_R) \sum_{\mu \in x,y,z,l,w,h,\theta} \mathcal{L}_{\text{smooth-L1}}(\mu, \mu^t), \tag{12}$$

where $\mathbb{I}(\text{IoU} \geq \alpha_R)$ indicates that only proposals with IoU $\geq \alpha_R$ contribute to the regression loss.

4. Experiments

In this section, we evaluate our CT3D on two public datasets, KITTI [7] and Waymo [18, 42]. Furthermore, we conduct comprehensive ablation studies to verify the effectiveness of each module in CT3D.

4.1. Dataset

KITTI Dataset. KITTI dataset officially contains 7,481 training LiDAR samples and 7,518 testing LiDAR samples. Following the previous work [2], we split the original training data into 3,712 training samples and 3,769 validation samples for experimental studies.

Waymo Dataset. Waymo dataset consists of 798 training sequences with around 158,361 LiDAR samples, and 202 validation sequences with 40,077 LiDAR samples. This large-scale Waymo dataset detection task is more challenging due to its various autonomous driving scenarios [42].

Method	Par.	3D Detection - Car		
	(M)	Easy	Mod.	Hard
LiDAR & RGB				
MV3D, *CVPR 2017* [3]	-	74.97	63.63	54.00
ContFuse, *ECCV 2018* [17]	-	83.68	68.78	61.67
AVOD-FPN, *IROS 2018* [12]	-	83.07	71.76	65.73
F-PointNet, *CVPR 2018* [21]	40	82.19	69.79	60.59
UberATG-MMF, *CVPR 2019* [16]	-	88.40	77.43	70.22
3D-CVF at SPA, *ECCV 2020* [38]	-	89.20	80.05	73.11
CLOCs, *IROS 2020* [20]	-	88.94	80.67	77.15
LiDAR only				
SECOND, *Sensor 2018* [33]	20	83.34	72.55	65.82
PointPillars, *CVPR 2019* [13]	18	82.58	74.31	68.99
STD, *ICCV 2019* [37]	-	87.95	79.71	75.09
PointRCNN, *CVPR 2019* [25]	16	86.96	75.64	70.70
3D IoU Loss, *3DV 2019* [40]	-	86.16	76.50	71.39
Part-A^2, *PAMI 2020* [26]	226	87.81	78.49	73.51
SA-SSD, *CVPR 2020* [10]	40.8	88.75	79.79	74.16
3DSSD, *CVPR 2020* [36]	-	88.36	79.57	74.55
PV-RCNN, *CVPR 2020* [24]	50	90.25	81.43	76.82
Voxel-RCNN, *AAAI 2021* [4]	28	**90.90**	81.62	77.06
CT3D (Ours)	30	87.83	**81.77**	**77.16**

Table 1. Performance comparisons with state-of-the-art methods on the KITTI *test* set. All results are reported by the average precision with 0.7 IoU threshold and 40 recall positions.

4.2. Implementation Details

RPN. We adopt SECOND [33] as our RPN due to its high-quality proposals and fast speed of inference. For the KITTI dataset, the X, Y, Z axis ranges are set as $(0, 70.4)$, $(-40, 40)$, $(-3, 1)$, and the voxel size is set as $(0.05\text{m}, 0.05\text{m}, 0.1\text{m})$ in $(X\text{-axis}, Y\text{-axis}, Z\text{-axis})$. For the Waymo dataset, the corresponding axis ranges are $(-75.2, 75.2)$, $(-75.2, 75.2)$, $(-2, 4)$, and the voxel size is $(0.1\text{m}, 0.1\text{m}, 0.15\text{m})$. $\mathcal{L}_{\text{RPN}}$ consists of the Focal-Loss classification branch and the Smooth-L1-Loss based regression branch. Please refer to OpenPCDet [30] for more details since we conduct our experiments with this toolbox.

Training Details. We use 8 V100 GPUs to train the entire network with batch size 24 for the KITTI dataset and batch size 16 for Waymo dataset. For the encoder and decoder modules of channel-wise transformer, we set $\alpha = 1.2$ and $H = 4$. For training targets, we set $\alpha_F = 0.75, \alpha_B = 0.25, \alpha_R = 0.55$, respectively. The whole CT3D framework is trained end-to-end from scratch with ADAM optimizer for 100 epochs. We adopt cosine annealing learning rate strategy for our learning rate decay, and the maximum of leaning rate is 0.001. In the training stage, only 128 proposals are randomly selected to calculate the confidence loss while 64 (IoU $\geq \alpha_R$) proposals are selected to calculate the regression loss. In the inference stage, top-100 proposals are selected for the final prediction.

4.3. Detection Results on the KITTI Dataset

We compare our CT3D with state-of-the-art methods on both the KITTI *test* and *val* sets with 0.7 IoU threshold. For our test submission, all the released training data is used to train the model. Following [25, 24, 4, 10], the average precision (AP) for *test* set is calculated with 40 recall positions, while the AP for *val* set is calculated with 11 recall positions when compared to the previous methods[1].

Method	3D Detection - Car		
	Easy	Mod.	Hard
LiDAR & RGB			
MV3D, *CVPR 2017* [3]	71.29	62.68	56.56
ContFuse, *ECCV 2018* [17]	-	73.25	-
AVOD-FPN, *IROS 2018* [12]	-	74.44	-
F-PointNet, *CVPR 2018* [21]	83.76	70.92	63.65
3D-CVF at SPA, *ECCV 2020* [38]	89.67	79.88	78.47
LiDAR only			
SECOND, *Sensor 2018* [33]	88.61	78.62	77.22
PointPillars, *CVPR 2019* [13]	86.62	76.06	68.91
STD, *ICCV 2019* [37]	89.70	79.80	**79.30**
PointRCNN, *CVPR 2019* [25]	88.88	78.63	77.38
SA-SSD, *CVPR 2020* [10]	**90.15**	79.91	78.78
3DSSD, *CVPR 2020* [36]	89.71	79.45	78.67
PV-RCNN, *CVPR 2020* [24]	89.35	83.69	78.70
Voxel-RCNN, *AAAI 2021* [4]	89.41	84.52	78.93
CT3D (Ours)	89.54	**86.06**	78.99

Table 2. Performance comparisons with state-of-the-art methods on the KITTI *val* set. All results are reported by the average precision with 0.7 IoU threshold and 11 recall positions.

IoU	BEV Detection			3D Detection		
Thr.	Easy	Mod.	Hard	Easy	Mod.	Hard
0.7	96.14	91.88	89.63	92.85	85.82	83.46

Table 3. Performance of our CT3D on the KITTI *val* set with AP calculated by 40 recall positions for *car* category.

IoU	Pedestrian			Cyclist		
Thr.	Easy	Mod.	Hard	Easy	Mod.	Hard
0.5	65.73	58.56	53.04	91.99	71.60	67.34

Table 4. Performance for *pedestrian* and *cyclist* on the KITTI.

Performance Comparisons. Table 1 illustrates the performance comparisons between our method and state-of-the-art methods on the official KITTI *test* server. It shows CT3D achieves the best performance on *moderate* and *hard* levels for *car* detection on both *LiDAR only* and *Lidar&RGB* modalities, especially for the most important *moderate* level [8]. Compared with the newest released PV-RCNN which shares the same RPN (*i.e.,* SECOND) as ours, CT3D achieves better performance while requiring about 1/3 times of parameters for refinement. Besides, as shown in Figure 4, CT3D presents much better visualization performance as compared to the PV-RCNN. This significant improvement mainly comes from the fact that CT3D processes the raw points in refinement stage rather than relying

[1] The setting of AP calculation is modified from 11 recall positions to 40 recall positions on 08.10.2019. For fair comparison with previous methods, we exploit the 11 recall setting on *val* set.

Difficulty	Method	3D Detection - Vehicle				BEV Detection - Vehicle			
		Overall	0-30m	30-50m	50m-Inf	Overall	0-30m	30-50m	50m-Inf
LEVEL_1	PointPillar, *CVPR 2019* [13]	56.62	81.01	51.75	27.94	75.57	92.10	74.06	55.47
	MVF, *CoRL 2020* [42]	62.93	86.30	60.02	36.02	80.40	93.59	79.21	63.09
	Pillar-OD, *arXiv 2020* [32]	69.80	88.53	66.50	42.93	87.11	95.78	84.87	72.12
	PV-RCNN, *CVPR 2020* [24]	70.30	91.92	69.21	42.17	82.96	97.35	82.99	64.97
	Voxel-RCNN, *AAAI 2021* [4]	75.59	92.49	74.09	53.15	88.19	97.62	87.34	77.70
	CT3D (Ours)	**76.30**	**92.51**	**75.07**	**55.36**	**90.50**	**97.64**	**88.06**	**78.89**
LEVEL_2	PV-RCNN, *CVPR 2020* [24]	65.36	91.58	65.13	36.46	77.45	94.64	80.39	55.39
	Voxel-RCNN, *AAAI 2021* [4]	66.59	91.74	67.89	40.80	81.07	96.99	81.37	63.26
	CT3D (Ours)	**69.04**	**91.76**	**68.93**	**42.60**	**81.74**	**97.05**	**82.22**	**64.34**

Table 5. Performance comparisons with state-of-the-art methods on the Waymo dataset with 202 validation sequences ($\sim$ 40k samples) for vehicle detection.

on human-specified designs and sub-optimal intermediate features. Note the AP on *easy* level of our CT3D is comparatively worse, there might be two reasons. First, we only sample 256 raw point within each proposal for all levels even the proposals in *easy* level usually have a much larger number of points. Second, we observe that KITTI exhibits large distribution differences between *trainval* and *test* sets.

For further validation, we conduct comparisons with previous methods on the KITTI *val* set. It shows that our CT3D outperforms all the other methods with a large margin, leading the state-of-the-art method Voxel-RCNN by 1.54% on *moderate* level, and achieves the competitive result on *easy* level. This improvement also verifies the effectiveness of our method, indicating our CT3D could better model the context information and dependencies as compared to the methods based on multi-scale feature fusion. Our model can also achieve strong performance on *pedestrian* and *cyclist* detection. The *car*-BEV, *pedestrian*-3D and *cyclist*-3D results are presented in Table 3 and Table 4 for reference.

4.4. Detection Results on the Waymo Dataset

As for the Waymo dataset, we train our model on the training set and evaluate it on the validation set. Likewise, the mAP is calculated with 0.7 IoU threshold for vehicle detection. The data is split into two difficulty levels: LELVEL_1 denotes objects containing more than 5 points , LELVEL_2 denotes objects containing $1 \sim 5$ points.

Performance Comparisons. In Table 5, we compare our CT3D with state-of-the-art methods based on official released evaluation tools [29]. It can be seen that our method outperforms all previous methods with remarkable margins on all distance ranges of interest in both LEVEL_1 and LEVEL_2. CT3D achieves 76.30% for the commonly used LEVEL_1 3D mAP evaluation metric, surpassing previous state-of-the-art method Voxel-RCNN by 0.71% on 3D detection, and 2.31% on bird-view detection. This significant improvement also verifies the effectiveness of our CT3D approach on large-scale point cloud feature representation. We report the results of LEVEL_2 difficulty in Table 5, our method outperforms Voxel-RCNN significantly by 2.45% on 3D detection. A contributing factor is that Voxel-RCNN limits the feature interactions via dividing the RoI space into grids, while our proposed CT3D has the obvious advantage of capturing long-range interactions among sparse points.

4.5. Ablation Studies

In this section, we conduct comprehensive ablation studies for the CT3D to verify the effectiveness of each individual component. We report the 3D detection AP metric with 40 recall positions on the KITTI *val* set.

Different RPN Backbones. In Table 6, we validate the effects of our refinement network with "SECOND RPN [33]," and "PointPillar RPN [13]", respectively. It can be seen that the detection performance boosts with +5.47% and +4.82% when compared to the RPN baselines. This benefits from that our two-stage framework CT3D could be integrated on the top of any RPNs to provide strong ability for proposal refinement. We also provide the amount of parameters in Table 6 for reference.

Proposal-to-point Embedding. We investigate the importance of the keypoints subtraction strategy by comparing it with the baseline size-orientation strategy adopted in the proposal-to-point embedding of Sec. 3.2. The 2^{nd} and 3^{rd} rows of Table 7 show that keypoints subtraction approach significantly improves the performance in all three difficulty levels. The rationale behind this strategy is that the relative coordinates between each point and the proposal keypoints could provide more effective geometric information, forming high-quality point location embeddings.

Self-attention Encoding. The 1^{st} and 3^{rd} rows of Table 7 show that removing the self-attention encoding drops performance a lot, which demonstrates that the self-attention enables better feature representation for each point by aggregating the global-aware context information and dependencies. Moreover, we visualize the attention maps of the last self-attention layer of a trained model from different epoch checkpoints. As shown in Figure 5, the points on cars get more attention in epoch 80, even in an extremely sparse case as Figure 5 (c). On the contrary, the background points get less attention with the training process. There-

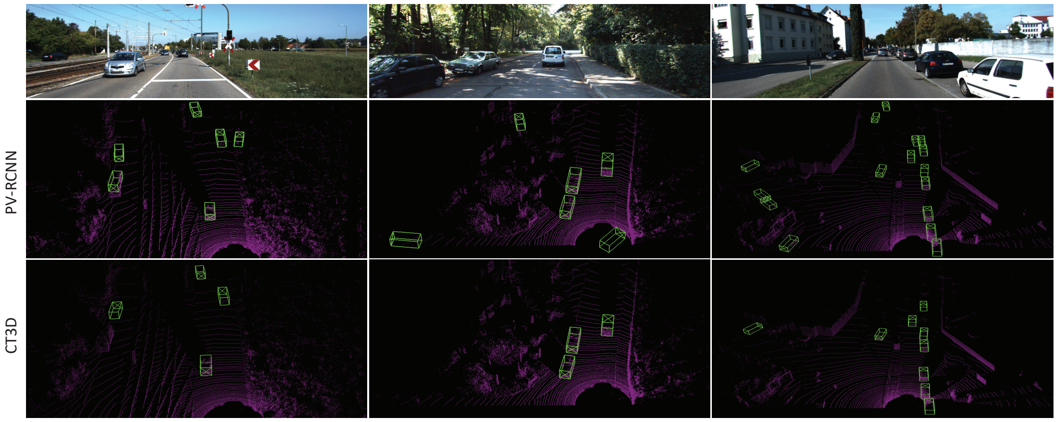

Figure 4. Qualitative comparison results of 3D object detection on the KITTI *test* set. Our CT3D enables more reasonable and accurate detection as compared to the PV-RCNN.

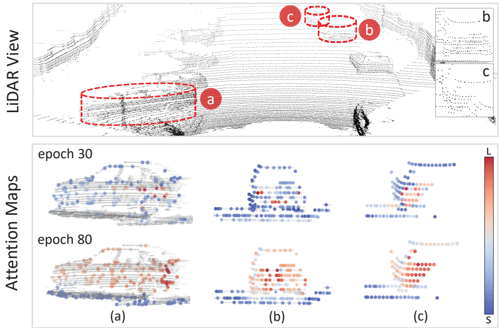

Figure 5. Attention maps generated by the self-attention layer. We visualize the weights of at most 256 sampled points within 3 RoIs (red dotted line) as the 30-th and 80-th epochs.

PointPillar RPN	SECOND RPN	Two-stage refinement	Par. (M)	Moderate AP (%)
✓			18	79.26
✓		✓	28	84.08
	✓		20	80.35
	✓	✓	30	**85.82**

Table 6. Ablation studies for different RPNs on the KITTI *val* set in terms of 3D detection AP metric with 40 recall positions.

K. S.	S. E.	S. D.	C. R.	E. C. R	Easy	Mod.	Hard
✓		✓			90.29	79.20	74.59
	✓	✓			91.92	83.41	81.79
✓	✓	✓			92.09	85.10	82.98
✓	✓		✓		92.56	85.34	83.23
✓	✓			✓	**92.85**	**85.82**	**83.46**

Table 7. Ablation studies for proposal-to-point embedding, self-attention encoding and channel-wise decoding on the KITTI *val* set. "K. S." stands for the keypoints subtraction strategy, "S. E." stands for the self-attention encoding, "S. D.", "C. R." and "E. C. R." represent the standard decoding, channel-wise re-weighting, and our extended channel-wise re-weighting, respectively.

fore, CT3D pays more attention to foreground points, and thus achieves considerable performance.

Channel-wise Decoding. As shown in the 3^{rd}, 4^{th} and 5^{th} rows of Table 7, the extended channel-wise re-weighting outperforms both the standard decoding and channel-wise re-weighting with a large margin. This benefits from the integration of the standard decoding and the channel-wise re-weighting for both global and channel-wise local aggregation, generating more effective decoding weights.

5. Conclusion

In this paper, we present a two-stage 3D object detection framework CT3D with a novel channel-wise Transformer architecture. Our method first encodes the proposal information into each raw point via an efficient proposal-to-point embedding, followed by self-attention to capture the long-range interactions among points. Subsequently, we transform the encoded point features into a global proposal-aware representation by an extended channel-wise re-weighting scheme which could obtain effective decoding weights for all points. The CT3D provides a flexible and highly-effective framework which is particularly helpful for point cloud detection tasks. Experimental results on both the KITTI dataset and the large-scale Waymo dataset also verify that CT3D could achieve significant improvement over the state-of-the-art methods.

Acknowledgements

This work was supported by Alibaba Innovative Research (AIR) progamm and Major Scientifc Research Project of Zhejiang Lab (No. 2019DB0ZX01).

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 213–229, 2020.

[2] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals for accurate object class detection. *Advances in Neural Information Processing Systems (NIPS)*, 28:424–432, 2015.

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

[4] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. *arXiv preprint arXiv:2012.15712*, 2020.

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[6] Nico Engel, Vasileios Belagiannis, and Klaus Dietmayer. Point transformer. *arXiv preprint arXiv:2011.00931*, 2020.

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

[8] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361, 2012.

[9] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *arXiv preprint arXiv:2012.09688*, 2020.

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

[11] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–799, 2018.

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

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

[14] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 734–750, 2018.

[15] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle detection from 3d lidar using fully convolutional network. *arXiv preprint arXiv:1608.07916*, 2016.

[16] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7345–7353, 2019.

[17] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 641–656, 2018.

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

[19] Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3d object detection with pointformer. *arXiv preprint arXiv:2012.11409*, 2020.

[20] Su Pang, Daniel Morris, and Hayder Radha. Clocs: Camera-lidar object candidates fusion for 3d object detection. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

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

[22] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 652–660, 2017.

[23] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems (NIPS)*, 30:5099–5108, 2017.

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

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

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

[27] Shuran Song and Jianxiong Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 808–816, 2016.

[28] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 945–953, 2015.

[29] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2446–2454, 2020.

[30] OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. `https://github.com/open-mmlab/OpenPCDet`, 2020.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008, 2017.

[32] Yue Wang, Alireza Fathi, Abhijit Kundu, David Ross, Caroline Pantofaru, Tom Funkhouser, and Justin Solomon. Pillar-based object detection for autonomous driving. *arXiv preprint arXiv:2007.10323*, 2020.

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

[34] Bin Yang, Ming Liang, and Raquel Urtasun. Hdnet: Exploiting hd maps for 3d object detection. In *Conference on Robot Learning*, pages 146–155, 2018.

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

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

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

[38] Jin Hyeok Yoo, Yecheol Kim, Ji Song Kim, and Jun Won Choi. 3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object detection. *arXiv preprint arXiv:2004.12636*, 3, 2020.

[39] Wu Zheng, Weiliang Tang, Sijin Chen, Li Jiang, and Chi-Wing Fu. Cia-ssd: Confident iou-aware single-stage object detector from point cloud. *arXiv preprint arXiv:2012.03015*, 2020.

[40] Dingfu Zhou, Jin Fang, Xibin Song, Chenye Guan, Junbo Yin, Yuchao Dai, and Ruigang Yang. Iou loss for 2d/3d object detection. In *2019 International Conference on 3D Vision (3DV)*, pages 85–94, 2019.

[41] Xingyi Zhou, Jiacheng Zhuo, and Philipp Krahenbuhl. Bottom-up object detection by grouping extreme and center points. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 850–859, 2019.

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

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

[44] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.