Voxel Set Transformer: A Set-to-Set Approach to 3D Object Detection from Point Clouds

Chenhang He, Ruihuang Li, Shuai Li, Lei Zhang*
The Hong Kong Polytechnic University
{csche, csrhli, csshuaili, cshzeng}@comp.polyu.edu.hk

Abstract

Transformer has demonstrated promising performance in many 2D vision tasks. However, it is cumbersome to compute the self-attention on large-scale point cloud data because point cloud is a long sequence and unevenly distributed in 3D space. To solve this issue, existing methods usually compute self-attention locally by grouping the points into clusters of the same size, or perform convolutional self-attention on a discretized representation. However, the former results in stochastic point dropout, while the latter typically has narrow attention fields. In this paper, we propose a novel voxel-based architecture, namely Voxel Set Transformer (VoxSeT), to detect 3D objects from point clouds by means of set-to-set translation. VoxSeT is built upon a voxel-based set attention (VSA) module, which reduces the self-attention in each voxel by two cross-attentions and models features in a hidden space induced by a group of latent codes. With the VSA module, VoxSeT can manage voxelized point clusters with arbitrary size in a wide range, and process them in parallel with linear complexity. The proposed VoxSeT integrates the high performance of transformer with the efficiency of voxel-based model, which can be used as a good alternative to the convolutional and point-based backbones. VoxSeT reports competitive results on the KITTI and Waymo detection benchmarks. The source codes can be found at https://github.com/skyhehe123/VoxSeT.

1. Introduction

Object detection from 3D point cloud has been receiving extensive attention as it empowers many applications like autonomous driving, robotics and virtual reality. Unlike 2D images, 3D point clouds are naturally sparse and unevenly distributed in continuous space, impeding the CNN layers from being directly applied. To resolve this issue, some approaches [5, 10, 42, 49, 51] first transform the point cloud into a discrete representation and then apply CNN models to extract high dimensional features. Another class of approaches [27, 33, 41, 44, 45] model the point cloud in continuous space, where the multi-scale features are extracted through interleaved grouping and aggregation steps.

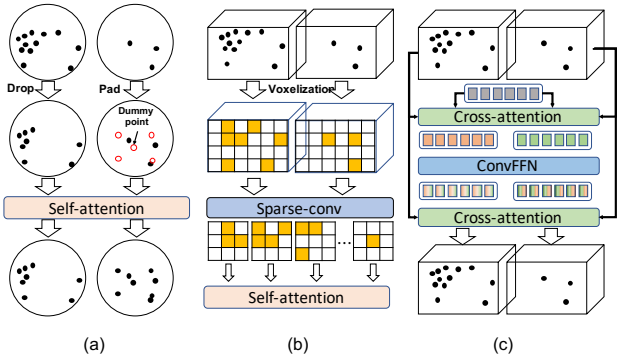

Figure 1. The illustrations of (a) grouping-based, (b) convolutional-based and (c) our proposed induced set-based attention mechanisms.

Beyond the above two schemes, transformer-based models [9, 21–23, 31, 47] have recently attracted great interest in processing point cloud data as the self-attention used in transformers is invariant to permutation and cardinality of the input components, which makes transformer an appropriate choice for point cloud processing. The main limitation of transformer models, however, lies in that the self-attention computation is quadratic. Each token has to be updated by using all the other tokens from previous layers, making self-attention intractable for long sequence point clouds. Point Transformer [47] builds transformers upon a PointNet [28] architecture, which hierarchically groups the point cloud data into different clusters and computes self-attention in each cluster. CT3D [31] presents a two-stage point cloud detector, where 3D RoIs are extracted to group the raw points in the first stage and transformers are applied to the grouped points in the second stage.

However, since the distribution of point clouds is extremely uneven, the number of points in each cluster varies a lot. To enable the self-attention to run in parallel, cur-

*Corresponding author.

rent approaches [23, 31, 47] balance the token number in each cluster by stochastically dropping points or padding dummy points (see Figure 1(a)). This results in unstable detection results and redundant computations. Besides, each operation of grouping n points to m clusters will cost $\mathcal{O}(nm)$ complexity, which is relatively intensive. Alternatively, Voxel Transformer [21] performs self-attention on a discrete voxel grid, as depicted in Figure 1(b). It computes self-attention in a convolutional manner and hence is as efficient as sparse convolution with $\mathcal{O}(n)$ complexity. However, since convolutional attention is a point-wise operation, to save the memory, the attention field of the convolutional kernel is typically small, thus hindering the voxel transformer to model long-range dependencies. It is worth mentioning that though Group-free [20] and 3DETR [22] present a promising solution by computing self-attention on a reduced set of seed points, this solution is only applicable to indoor scenes, where the point clouds are relatively dense and concentrated. Considering that the point clouds of outdoor scenes are typically sparse, large-scale (*e.g.*, $> 20k$), and unevenly distributed, the scale and coverage of seed points remain an issue.

To address the above issues, we introduce a voxel-based set attention (VSA) module. For each VSA, we divide the whole scene into non-overlapping 3D voxels and compute the voxel indices of the input point with instant efficiency. We use these voxels to determine the attentive region which is analogous to the window attention in SwinTransformer [19]. Unlike image, LiDAR has irregular structures, and the resulting attention groups have different lengths, which hinders the parallelization of the model.

Inspired by the induced set transformer [13], we assign a group of trainable "latent codes" to each voxel. These latent codes build a fixed-length bottleneck for the point cloud, through which the information from input points within the voxel can be compressed to a static hidden space. This formulation is based on the key observation that the self-attention matrix is typically low-rank, and hence we can decompose an intensive full self-attention into two consecutive cross-attention modules. As shown in Figure 1(c), VSA first transforms the latent codes, which serve as queries, to a hidden space by attending to the projected features, *i.e.*, keys and values, from the input points. The transformed hidden features, which encode the context information of the input points in each voxel, are enriched by a convolutional feed-forward network, in which the features across voxels exchange their information in spatial domain. After that, the hidden features are attentively fused with input, producing output features of the input resolution. By leveraging the latent codes, the cross-attention performed in all voxels can be vectorized, making VSA a highly parallel module. Given n d-dimensional input features and k latent codes, VSA has a complexity of $\mathcal{O}(nkd)$ and it can be implemented with general matrix multiplications.

With VSA, we propose a Voxel Set Transformer (VoxSeT) to detect 3D objects by learning point cloud features in a set-to-set translation process. VoxSeT is composed of VSA modules, MLP layers and a shallow CNN for Birds-Eye-View (BEV) feature extraction. To verify the effectiveness of the proposed model, we conduct experiments on two 3D detection benchmarks, KITTI and Waymo open dataset. VoxSeT achieves competitive performance with current state-of-the-arts. In addition, the proposed VSA module can be seamlessly adopted into point-based detectors such as PointRCNN [33], and demonstrates advantages over the set abstraction module.

In summary, in this work we first invent a voxel-based set attention module, which can model long-range dependencies from the token cluster of arbitrary size, bypassing the limitation of current grouped-based and convolution-based attention modules. We then present a Voxel Set Transformer to learn point cloud features effectively by leveraging the superiority of transformer on large-scale sequential data. Our work provides a novel alternative to the current convolutional and point-based backbones for 3D point cloud data processing.

2. Related work

2.1. 3D object detection from point clouds

Early approaches on 3D object detection from point cloud can be categorized into two classes. The first class of methods transform the point cloud into more compact representations, *e.g.*, Birds-Eye-View (BEV) images [3, 11, 34], frontal-view range images [2, 7, 18], and volumetric features [14, 43, 51]. Yan *et al.* [42] developed a sparse convolutional backbone to efficiently process the point clouds by encoding the point clouds into a 3D sparse tensor. Lang *et al.* [12] further accelerated the detection rate by stacking the voxel features as a "pillar" and using 2D CNN to process. Another class of methods [25, 27, 33, 44, 45] process the point cloud in a continuous space by employing a PointNet [29] architecture. The point-wise features in multi-scales are extracted in stages with interleaved grouping and sampling operations. Shi *et al.* [33] and Yang *et al.* [45] proposed to generate 3D RoIs from PointNet outputs and apply the RoIs to group point-wise features for further refinement. Qi *et al.* [26] proposed a deep voting method to cluster the points from objects' surface to detect the object with insufficient points. Unlike compact representations, point-wise features preserve more details and fine-grained structures of original point clouds. Based on this fact, some approaches [10, 32] employ a hybrid representation in both point and voxel spaces to achieve more reliable detection outputs.

Our proposed architecture is largely motivated by voxel-

based approaches. We partition the point cloud into voxel grid and execute self-attention locally, endowing our model with inductive bias and computational efficiency.

2.2. Transformer in point cloud analysis

Recently, Transformer [38] has demonstrated its great success in many computer vision tasks such as image classification [6, 37], 2D object detection [1, 19], and other dense prediction tasks [30, 48]. For point cloud analysis, Zhao *et al.* [47] proposed a novel subtraction attention based operator for point cloud classification and segmentation. Guo *et al.* [9] investigated a dual attention to process the point clouds in feature and edge space. Misra *et al.* [22] and Liu *et al.* [20] used transformer to process point clouds as sequential data, preventing the models from stacking hierarchical grouping and sampling modules. Miao *et al.* [21] embeded self-attention into a sparse convolutional kernel. Sheng *et al.* [31] built the transformer on top of a two-stage detector and operated attention on the points grouped by RoIs.

Unlike the above approaches that perform self-attention on a fixed-size token cluster, our proposed Voxel Set Transformer leverages the idea of induced set transformer [13] to decompose self-attention into two cross-attentions, making it possible to perform self-attention on the token clusters of arbitrary size.

3. Methodology

3.1. Preliminary

It is prohibitive to directly apply self-attention on point cloud data due to its quadratic computational complexity. To bypass the issue, an induced set attention block was proposed in [13], where the full self-attention in a set was approximated by two reduced cross-attentions induced by a group of latent codes. Given an input set $X \in \mathcal{R}^{n\times d}$ of size n with dimension d and k latent codes $L \in \mathcal{R}^{k\times d}$, the output set $O \in \mathcal{R}^{n\times d}$ from the induced set attention block can be formulated as

$$H = \text{CrossAttention}(L, X) \in \mathcal{R}^{k\times d}, \tag{1}$$

$$\hat{H} = \text{FFN}(H) \in \mathcal{R}^{k\times d}, \tag{2}$$

$$O = \text{CrossAttention}(X, \hat{H}) \in \mathcal{R}^{n\times d}. \tag{3}$$

The first cross attention transforms the latent features L into hidden features H by attending to the input set. This step costs $\mathcal{O}(nkd)$ complexity, which is linear to n as the number of latent codes k is fixed and usually very small. The transformed hidden features contain information about the input set X and then they are updated by a point-wise feed-forward network (FFN). This point-wise operation costs $\mathcal{O}(k)$ complexity and it learns highly semantic features from the input set. The second cross attention attends the input set to the resulting hidden features, which costs $\mathcal{O}(nkd)$ complexity, producing an output set of length n. The induced set attention is based on the assumption that the self-attention can be approximated with low-rank projections, thus the self-attention can be regarded as performing a k-clustering on the inputs where the latent codes serve as cluster centers. This is also analogous to the clustered attention [39] and *Linformer* [40], where the input set is explicitly reduced with linear projection.

3.2. Voxel-based Set Attention (VSA)

Unlike images, point clouds are widely distributed and have weak semantic associations in scene level, while they have strong structural details in the local region. Instead of compressing all the input points into a hidden space, we modify the above induced set attention to be performed locally. Specifically, we partition the scene into a voxel grid and assign a set of latent codes to each voxel. We refer to the module as *Voxel-based Set Attention* (VSA).

Scatter kernel function. As mentioned before, VSA is a highly parallel module, where the operations across voxels can be vectorized. This vectorization can be achieved by the *scatter* function[1], which is a cuda kernel library that performs symmetric reduction, *e.g.*, sum, max and mean, on different segments of a matrix. In our case, we regard the input set as a single matrix, each row of which corresponds to a point-wise feature, and its belonging voxel can be indexed by a table of voxel coordinates.

Let $\{p_i = (x_i, y_i, z_i) : i = 1, ...n\}$ denote the coordinates of point cloud and $[d_x, d_y, d_z]$ be the voxel size in three dimensions. The voxel coordinates $\mathcal{V}$ can be computed by $\mathcal{V} = \{\mathcal{V}_i = (\lfloor \frac{x_i}{d_x} \rfloor, \lfloor \frac{y_i}{d_y} \rfloor, \lfloor \frac{z_i}{d_z} \rfloor) : i = 1, ..., n\}$, where $\lfloor \cdot \rfloor$ is the floor function. Hence, given point-wise input features $\{X_i : 1 = 1, ...n\}$, their reduced voxel-wise form $\{Y_j : j = 1, ..., m\}$ after a symmetric function $\text{F}(\cdot)$ can be represented as:

$$Y = \{\text{F}(\{X_i : \mathcal{V}_i = j\}) : j = 1, ..., m\} \tag{4}$$

where m is the number of non-empty voxels. With $scatter$ function $\text{F}_{scatter}$, the above equation can be written in a vectorized form, *i.e.*,

$$Y = \text{F}_{scatter}(X, \mathcal{V}). \tag{5}$$

By deploying VSA, we do not need to stochastically drop or pad the points in each voxel and the complexity of the model is linear.

In Figure 2, we illustrate the VSA in a matrix-multiplication form for ease of comprehension. As can be seen, the module is analogous to an encoder-decoder architecture, where the input set is encoded to a hidden space, then the hidden features are refined through a ConvFFN and finally decoded to produce the output set.

[1] https://github.com/rusty1s/pytorch_scatter

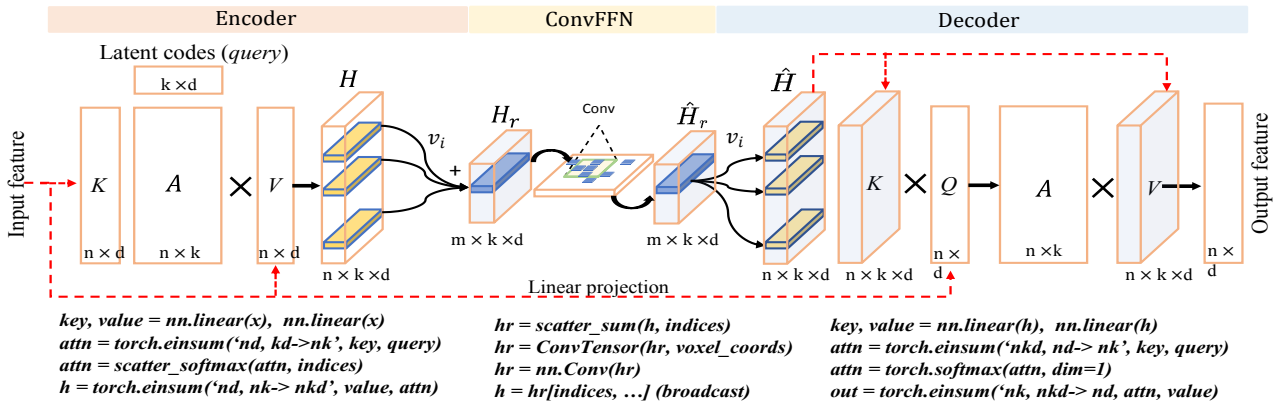

Figure 2. The matrix multiplication used in implementing the voxel-based set attention (VSA) module. The pseudo codes in PyTorch-stype of each step are presented bellow the diagram.

Encoder. In the encoder, we first project the input features from the previous module to the key $K \in \mathcal{R}^{n\times d}$ and the value $V \in \mathcal{R}^{n\times d}$ with linear projections, respectively. Next, we perform cross attention between the key and the latent codes (query) $L \in \mathcal{R}^{k\times d}$, producing the attention matrix $\mathcal{A} \in \mathcal{R}^{n\times k\times d}$. The attention matrix $\mathcal{A}$ is then normalized voxel-wisely to obtain $\tilde{\mathcal{A}}$, and multiplied with the value, producing hidden features $\tilde{H}$. The calculation of $\tilde{H}$ can be formulated as:

$$\tilde{\mathcal{A}} = \text{Softmax}_{scatter}(\mathcal{A}, \mathcal{V}),\ \mathcal{A} = KL^T, \tag{6}$$

$$H = \tilde{\mathcal{A}}^T V. \tag{7}$$

After that, we perform voxel-wise reduction on the hidden features based on the voxel indices $\mathcal{V}$:

$$H_r = \text{Sum}_{scatter}(H, \mathcal{V}). \tag{8}$$

The overall computations in the encoder include two GEMMs and two *scatter* operations. The overall complexity of the encoder is $\mathcal{O}(2n(k+1)d)$.

It is worth mentioning that the cross-attention based encoding scheme can be viewed as an extension of voxel feature encoding (VFE) used in [12, 42, 51]. The difference is that VFE encodes the points within a voxel into a single feature vector, while our scheme encodes the points based on a *codebook* consisting of the latent features. Owing to the high expressive power of VSA, we can use a relatively large voxel size to capture the features in a wide range.

Convolutional feed-forward network. The core idea of VSA is to encode the region-wise features into a hidden space using latent codes. The hidden features work as a bottleneck, through which we apply a ConvFFN to achieve more flexible and complex information update. Unlike conventional FFN that only performs point-wise token update, ConvFFN enables the information exchange accross voxels, which is especially important for dense prediction. To adaptively integrate voxel features with global dependencies, we scatter the reduced hidden features into a 3D sparse tensor based on their voxel coordinates C_r, and then conduct two depth-wise convolutions (DwConv) on them to enforce the feature interaction in spatial domain. Given convolutional weights W_1 and W_2, the enriched hidden features $\hat{H}_r \in \mathcal{R}^{m\times k\times d}$ from ConvFFN can be written as:

$$\hat{H}_r = \text{DwConv}(\sigma(\text{DwConv}(\mathcal{T}(H_r, C_r); W_1)); W_2), \tag{9}$$

where σ denotes the non-linear activation, $\mathcal{T}$ denotes the formulation of sparse tensor, and the number of groups in DwConv equals to that of latent features. This operation costs $\mathcal{O}(\frac{HWDkd}{d_x d_y d_z})$ complexity, where H, W, D refer to the point cloud range in three directions, respectively, and $[d_x, d_y, d_z]$ specifies the voxel size. The ConvFFN plays an important role in VSA as it introduces desirable inductive bias and global context to the module. More studies on ConvFNN will be discussed in Sec 4.5.

Decoder. The decoder reconstructs the output set from the enriched hidden features $\hat{H}_r$. Specifically, we first broadcast the hidden features based on the voxel indices $\mathcal{V}$, producing $\hat{H} \in \mathcal{R}^{n\times k\times d}$ which has the same length as the input set. Then we generate the query, key-value pair from the input set and the hidden features, respectively, with linear projections. Given matrices of query $Q \in \mathcal{R}^{n\times d}$, key $K \in \mathcal{R}^{n\times k\times d}$ and value $V \in \mathcal{R}^{n\times k\times d}$, the decoder output O can be calculated as:

$$\mathcal{A} = [\mathcal{A}_1, ..., \mathcal{A}_n] = [K_1 Q_1^T, ..., K_n Q_n^T], \tag{10}$$

$$\tilde{\mathcal{A}} = [\text{Softmax}(\mathcal{A}_1), ..., \text{Softmax}(\mathcal{A}_n)], \tag{11}$$

$$O = [O_1, ..., O_n] = [\tilde{\mathcal{A}}_1^T V_1, ..., \tilde{\mathcal{A}}_n^T V_n] \tag{12}$$

The overall computational complexity of the decoder is $\mathcal{O}(2nkd)$. Owing to the flexibility of cross-attention mechanism, VSA formulates the point cloud processing as a set-to-set translation problem.

Relative position embedding. As discussed in [23, 29], preserving the local structure of point cloud is crucial to

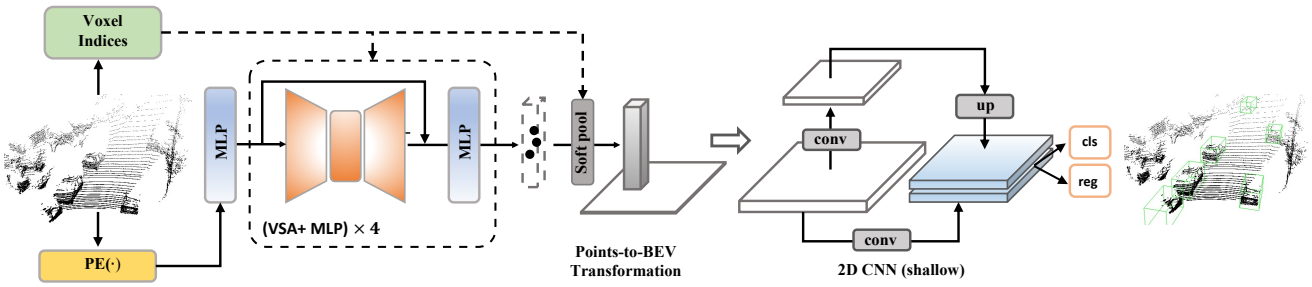

Figure 3. The overall architecture of the proposed Voxel Set Transformer (VoxSeT).

improve performance. Therefore, we introduce a Positional Embedding (PE) module to encode the local coordinates of the point clouds within a voxel to a high dimensional feature and inject them into each VSA module. Specifically, the PE module applies the Fourier parameterization to take values $[sin(f_k \pi x), cos(f_k \pi, x)]$, given the normalized local coordinates $x \in [0, 1]$ and the k^{th} frequency f_k with bandwidth L. The resulted Fourier embedding, which has a dimension of $3L$, is further mapped to the input dimension of the first MLP module through a learnable linear layer.

3.3. Voxel Set Transformer (VoxSeT)

The overall architecture of VoxSeT is illustrated in Figure 3. Following the traditional transformer paradigm, the VoxSeT backbone is composed of inter-connected multi-layer perception (MLP) and VSA modules. We use batch norm as the normalization layer and wrap each VSA module into a residual block for optimal gradient flow.

Unlike grouping-based approaches [25, 29] that progressively downsample and aggregate point-wise features for context extraction, our backbone extracts point cloud features as a set-to-set translation process. The semantic level of the features is controlled by the size of voxels in the VSA module. We empirically found that applying large voxels can learn richer context information, and demonstrate better understanding of the objects with sparse points, especially pedestrian and cyclist instances. We will present the settings of VSA modules in Sec. 4.1.

Birds-eye-view feature encoding. In point cloud detection, a common phenomenon is that models using dense birds-eye-view (BEV) features [4, 5, 42] generally achieve higher recall than those using sparse point-wise features [33, 45]. In this regard, we encode the point-wise features from the backbone into a BEV representation and apply a shallow CNN to increase the feature density. The CNN has only two strides, each involving three convolutions. The convolutional features of two strides are finally concatenated and passed to the detection head for bounding-box prediction. To generate BEV features, we aggregate the point-wise features within a pillar of size 0.36m×0.36m and apply a "soft-pooling" operation to produce features in BEV. Given point-wise output features $X^j \in \mathcal{R}^{k \times d}$ in the j^{th} pillar, the pillar-wise features after pooling F^j can be formulated as:

$$F^j = \sum_{m=1}^{k} X_m^j * w_m^j, \quad w_m^j = \frac{e^{X_m^j}}{\sum_{m=1}^{k} e^{X_m^j}}. \tag{13}$$

Detection head and training objectives. To enhance the expressiveness of VoxSeT backbone, we follow PointRCNN [33] to apply the foreground segmentation loss $\mathcal{L}_{seg}$ to the output features. This forces VoxSeT to capture contextual information for generating accurate bounding-boxes. The detection head follows the traditional anchor-based design [12, 42]. The final loss then becomes:

$$\mathcal{L} = \mathcal{L}_{seg} + \frac{1}{N_p}(\mathcal{L}_{cls} + \mathcal{L}_{reg}) + \mathcal{L}_{dir}), \tag{14}$$

where N_p is the number of positive samples whose IoU with anchors lies between $[\sigma_1, \sigma_2]$. $\mathcal{L}_{\text{cls}}$ is the focal loss for bounding-box classification and $\mathcal{L}_{\text{reg}}$ is the Smooth-L_l loss for bounding-box offsets regression. $\mathcal{L}_{dir}$ is a binary entropy loss for bounding-box orientation prediction. The readers are referred to [33] and [42] for details.

Two-stage model. It is worth noting that VoxSeT can be extended to a two-stage detector, in which we employ the efficient RoI head from LiDAR-RCNN [15] as our second-stage module. To more clearly illustrate our contribution, we also report the performance of a single-stage detector, and our VoxSeT demonstrates superior performance over the current single-stage baselines.

4. Experiments

In this section, we evaluate our proposed VoxSeT on two public detection datasets, KITTI [8] and Waymo [35]. We first introduce the training details of VoxSeT and the evaluation settings, and then compare our models with state-of-the-art detection models. Finally, we conduct an in-depth analysis of each component of VoxSeT.

4.1. Implementation details

Model setup. On the KITTI dataset, we select the LiDAR points that fall into the ranges [0m, 70.4m], [-40m,

Table 1. Performance comparison with state-of-the-art methods on the Waymo dataset with 202 validation sequences ($\sim 40k$ samples) for vehicle detection.

Method	Backbone	3D mAP				BEV mAP			
		Overall	0-30m	30-50m	50m-inf	Overall	0-30m	30-50m	50m-inf
LEVEL_1 (IoU=0.7):									
PointPillar [12] (CVPR19)	CNN	56.62	81.01	51.75	27.94	75.57	92.10	74.06	55.47
MVF [50] (CoRL20)	CNN	62.93	86.30	60.02	36.02	80.40	93.59	79.21	63.09
PV-RCNN [32] (CVPR20)	SpCNN	70.30	91.92	69.21	42.17	82.96	97.35	82.99	64.97
Voxel-RCNN [5] (AAAI21)	SpCNN	75.59	92.49	74.09	53.15	88.19	97.62	87.34	77.70
VoTR-TSD [21] (ICCV21)	Transformer	74.95	92.28	73.36	51.09	-	-	-	-
CT3D [31] (ICCV21)	SpCNN	76.30	92.51	75.07	**55.36**	**90.50**	**97.64**	88.06	**78.89**
VoxSeT (ours)	Transformer	76.02	91.13	75.75	54.23	89.12	95.12	87.36	77.78
VoxSeT + CT3D (RoI head)	Transformer	**77.82**	**92.78**	**77.21**	54.41	90.31	96.11	**88.12**	77.98
LEVEL_2 (IoU=0.7):									
PV-RCNN [32] (CVPR20)	SpCNN	65.36	91.58	65.13	36.46	77.45	94.64	80.39	55.39
Voxel-RCNN [5] (AAAI21)	SpCNN	66.59	91.74	67.89	40.80	81.07	96.99	81.37	63.26
VoTR-TSD [21] (ICCV21)	Transformer	65.91	-	-	-	-	-	-	-
CT3D [31] (ICCV21)	SpCNN	69.04	91.76	68.93	42.60	**81.74**	**97.05**	**82.22**	**64.34**
VoxSeT (ours)	Transformer	68.16	91.03	67.13	42.23	76.13	94.13	81.78	58.13
VoxSeT + CT3D (RoI head)	Transformer	**70.21**	**92.05**	**70.10**	**43.20**	80.56	96.79	80.44	62.37

Table 2. Performance comparison with traditional single-stage baseline models on the KITTI validation set. The results are reported by the mAP with 11 recall points.

Method	Vehicle			Pedestrian			Cyclist		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
SECOND [42]	88.61	78.62	77.22	56.55	52.98	47.73	80.58	67.15	63.10
PointPillars [12]	86.46	77.28	74.65	57.75	52.29	47.90	80.04	62.61	59.52
VoxSeT (single-stage)	88.45	78.48	77.07	60.62	54.74	50.39	84.07	68.11	65.14
Improvements	-0.16	-0.14	-0.15	2.87	1.76	2.49	3.49	0.96	2.04

40m], [-3m, 1m] along X, Y, Z axes, respectively, and abandon those points with the frontal view projections out of image. On the Waymo dataset, the points that lie between [-75.2m, 75m] in the X and Y axes, and [-2m, 4m] in the Z axis are selected. The voxel size of the first VSA layer is [0.32m, 0.32m, 4m] on KITTI and [0.32m, 0.32m, 6m] on Waymo. The voxel size is doubled along the X and Y axes in the next VSA block. The feature dimensions of the four VSA blocks are 16, 32, 64 and 128, respectively. The number of latent codes in each VSA block is 8 and the bandwidth L of the Positional Embedding (PE) module is 64.

Training and inference. The network is trained end-to-end on four RTX Quodra 8000 GPUs for 100 epochs with the Adam optimizer. The batch size, learning rate, and weight decay are set to 4, 0.003 and 0.01, respectively. The learning rate is decayed with the *onecycle* policy, where the momentum has a damping range of $[85\%, 95\%]$.

We apply the anchor settings in SECOND [42] in our single stage model. For the two-stage model, we sample 512 RoIs in training and 128 RoIs in inference. In the post-processing phase, the bounding-boxes are filtered by NMS with an IoU threshold of 0.1, and those having confidence over 0.3 are selected as final predictions. Data augmentations [4,42,51] are applied to improve the model generalization performance. For other default settings, the readers are referred to the OpenPCDet toolbox [36] used in this work.

4.2. Dataset and evaluation metrics

KITTI dataset [8]. KITTI contains 7,481 training samples and 7,518 testing samples. Following the common protocol [3], we split the labeled data into a training set with 3,712 samples and a validation set with 3,769 samples. We conduct experiments on the commonly used car category whose detection IoU threshold is 0.7, and report the results on three difficulty levels (*easy*, *moderate* and *hard*) according to the object size, occlusion state and truncation level.

Waymo open dataset [35]. This dataset consists of 798 training sequences and 202 validation sequences, where there are 158,361 samples and 40,077 samples, respectively. The evaluation metrics used are 3D mean Average Precision (mAP) with IoU threshold of 0.7 on the vehicle category. The measures are reported based on the distances from objects to sensor, *i.e.*, $0-30$m, $30-50$m and >50m, respectively. Two difficulty levels, LEVEL 1 (boxes with more than five LiDAR points) and LEVEL 2 (boxes with at least one LiDAR point) are considered.

Table 3. Performance comparison with state-of-the-art methods on the KITTI test set. The results are reported by the mAP with 0.7 IoU threshold and 40 recall points.

Method	*3D*		
	Easy	Moderate	Hard
LiDAR + RGB:			
MV3D [3] (CVPR17)	74.97	63.63	54.00
ContFuse [17] (ECCV18)	83.68	68.78	61.67
AVOD-FPN [11] (IROS18)	83.07	71.76	65.73
F-PointNet [27] (CVPR18)	82.19	69.79	60.59
MMF [16] (CVPR19)	88.40	77.43	70.22
3D-CVF [46] (ECCV20)	89.20	80.05	73.11
CLOCs [24] (IROS20)	88.94	80.67	77.15
LiDAR only:			
VoxelNet [51] (CVPR18)	77.47	65.11	57.73
SECOND [42] (Sensor18)	83.34	72.55	65.82
PointPillars [12] (CVPR19)	82.58	74.31	68.99
STD [45] (ICCV19)	87.95	79.71	75.09
PointRCNN [33] (CVPR19)	86.96	75.64	70.70
SA-SSD [10] (CVPR20)	88.75	79.79	74.16
3DSSD [45] (CVPR20)	88.36	79.57	74.55
PV-RCNN [45] (CVPR20)	**90.25**	81.43	76.82
Voxel-RCNN [45] (AAAI21)	87.95	79.71	75.09
CT3D [31] (ICCV21)	87.83	81.77	77.16
VoTR-TSD [21] (ICCV21)	89.90	**82.09**	**79.14**
VoxSeT (ours)	88.53	82.06	77.46

Table 4. Performance comparison with state-of-the-art methods on the KITTI validation set. The results are reported by the mAP with 0.7 IoU threshold and 11 recall points.

Method	*3D*		
	Easy	Moderate	Hard
LiDAR + RGB:			
MV3D [3] (CVPR17)	71.29	62.68	56.56
F-PointNet [27] (CVPR18)	83.76	70.92	63.65
3D-CVF [46] (ECCV20)	89.67	79.88	78.47
LiDAR only:			
SECOND [42] (Sensor18)	88.61	78.62	77.22
PointPillars [12] (CVPR19)	86.62	76.06	68.91
STD [45] (ICCV19)	89.70	79.80	**79.30**
PointRCNN [33] (CVPR19)	88.88	78.63	77.38
SA-SSD [10] (CVPR20)	**90.15**	79.91	78.78
3DSSD [45] (CVPR20)	89.71	79.45	78.67
PV-RCNN [45] (CVPR20)	89.35	83.69	78.70
Voxel-RCNN [45] (AAAI21)	89.41	84.52	78.93
CT3D [31] (ICCV21)	89.54	86.06	78.99
VoTR-TSD [21] (ICCV21)	89.04	84.04	78.68
VoxSeT (ours)	89.21	**86.71**	78.56

4.3. Results on the Waymo open dataset

We first evaluate the performance of VoxSeT on the Waymo open dataset. The results are summarized in Table 1. VoxSeT outperforms most of CNN-based models, leading PV-RCNN [32] by 5% LEVEL_1 mAP and VoxelRCNN [5] by 2.4% LEVEL_2 mAP. As one of the few transformer based models, our VoxSeT achieves better performance than its transformer-based competitor VoTR-TSD [21], which brings 0.9% and 1.4% improvements on LEVEL_1 and LEVEL_2 mAP, respectively. VoxSeT achieves comparable performance to the state-of-the-art method CT3D [31]. It should be noted that, however, CT3D actually employs a heavy transformer based RoI head, which has three self-attention encoding layers. By adopting this RoI head into VoxSeT, our model achieves better results, outperforming CT3D by 1.5 % LEVEL_1 mAP and 1.2% LEVEL_2 mAP. This demonstrates that as a new transformer based backbone network, VoxSeT surpasses Sparse CNN based networks. VoxSeT works especially well in the range of 30-50m, which indicates that transformer modules are better in capturing the context information in long-range areas.

4.4. Results on the KTTI Dataset

We then conduct experiments on the KITTI dataset to evaluate the performance of VoxSeT as a single-stage detection model. Our competitors are SECOND [42] and PointPillars [12], which represent two widely used baseline feature extractors. All the three methods use the same detection head and hyper-parameters in training. As shown in Table 2, VoxSeT achieves comparable performance to SECOND on vehicle class, but much better performance on Pedestrian and Cyclist classes. We believe this is because VoxSeT has a wider effective receptive field through the VSA conditioned on the large voxel, which is crucial to detecting the objects with sparse points.

As a two-stage detection model, VoxSeT achieves better performance than CT3D by 0.7% (Easy), 0.4% (Moderate) and 0.3% (Hard) mAP, respectively, as shown in Table 3. Compared with VoTR-TSD which relies on multi-scale backbone features, VoxSeT can still achieve comparable performance by using only singular point-wise features. We also evaluate VoxSeT on KITTI val. One can see that VoxSeT achieves leading accuracy on "Moderate" level but slightly lower accuracy on "Easy" and "Hard" levels. We believe this is because KITTI has a long-tailed distribution, and hence the "Moderate" samples dominate the hidden space of the VSA module.

It should be pointed out that both CT3D and VoTR employ convolutional architectures, and their performances on KITTI and Waymo datasets are not consistent. Specifically, CT3D works better on Waymo but its performance drops much on KITTI, while VoTR works better on KITTI but its performance on Waymo is much worse. In contrast, our VoxSeT exhibits consistently superior performance on both datasets, demonstrating its good generalization capacity.

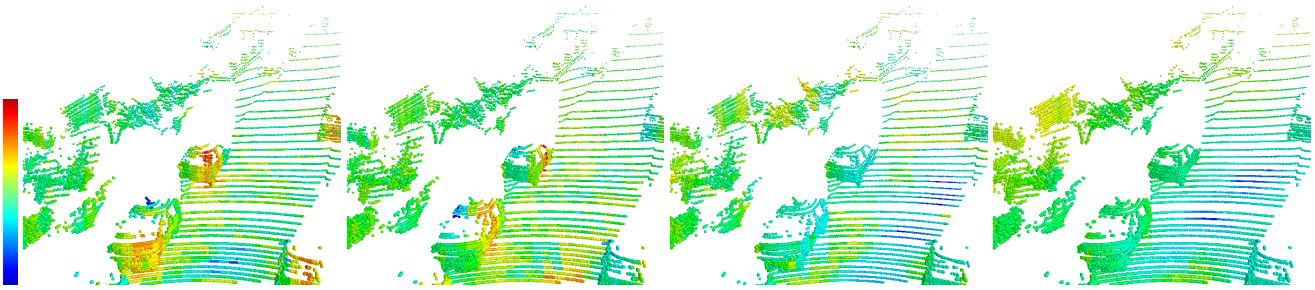

Figure 4. Visualization of the spatial attention maps induced by different latent codes in the last VSA module.

Table 5. Effects of enabling token interactions by convolutional feed-forward network. The mAP (11 recall points) of RPN on KITTI *val* are reported.

Settings	Easy	Moderate	Hard
default	88.31	79.56	77.84
ConvFFN→FFN	70.12	69.54	54.23

Table 6. The mAP (11 recall points) of RPN on KITTI *val* by using different number of latent codes (LC) in four VSA modules.

No. of latent codes	4	8	16
mAP	76.63	78.25	78.74

Table 7. Comparison of PointRCNN [33] detectors with VoxSeT and PointNet++ backbones. The mAP (11 recall points) on KITTI *val* are reported.

Settings	Easy	Moderate	Hard
PointRCNN	88.52	78.95	77.81
VSA-*PointRCNN*	89.61	80.14	78.69

Table 8. The latency and runtime memory on the KITTI dataset, tested by NVIDIA 2080Ti GPU.

Models	Latency	Memory (runtime)
SECOND [42]	48 ms	6093MB
PointPillars [12]	22ms	1508 MB
VoxSeT (single-stage)	34 ms	2381 MB

4.5. Ablation study

We conduct a series of ablation experiments to comprehend the roles of different components in VoxSeT.

Convolutional feed-forward network. Table 5 shows that replacing the proposed ConvFFN with the conventional FFN significantly degrades the accuracy, indicating that the local connectivity is crucial to the detection performance.

Effects of number of latent codes. In Table 6, we investigate the number of latent codes used in four VSA modules. We see that more latent codes can encode more context information of point cloud, and enhance the modeling capacity of VoxSeT.

Comparison with PointNet++ backbone. We train a PointRCNN [33] variant by replacing its PointNet++ backbone [33] with our VoxSeT backbone. From Table [29], one can observe obvious performance improvements. We believe this is because VSA module has better modeling power in terms of dynamic learning and large receptive field than the set abstraction (SA) module in PointNet++.

Latency and runtime memory. Table 8 shows that VoxSeT is faster and has less memory consumption compared to the sparse 3D CNN (SECOND). The higher performance than PointPillars and the acceptable runtime cost suggest that VoxSeT can be a good alternative to PointPillars in real-time applications.

Visualization of attention weights. Figure 4 visualizes the spatial attention maps for the latent codes in the last VSA module. We show the attention maps for 4 out of a total of 8 latent codes. One can observe that the VSA module focuses more on the object region and different latent codes encode different contexts of the objects, indicating the high expressiveness of VSA for point cloud data.

5. Conclusion and discussions

We proposed VoxSeT, a novel transformer-based framework for 3D object detection from LiDAR point clouds. In contrast to previous 3D LiDAR detectors, which use sparse CNN and PointNet backbones to learn point cloud features, we made the first attempt to model point cloud processing as set-to-set translation, which preserves the full resolution of raw point cloud at every step of feature extraction. We presented a voxel-based set attention module that performs self-attention on voxel clusters of arbitrary size and encodes point features with more discriminative context information from a large receptive field. Experimental results on the Waymo and KITTI datasets demonstrated that our VoxSeT can achieve competitive performance, making it a good alternative for point cloud modeling.

It should be noted that in VoxSeT, we only explored one possible formulation of liner attention based on the induced latent codes. This limits the expressive power of VoxSeT to represent different point cloud structures and their correlations. By using stronger attention mechanisms, the performance of VoxSeT can be further improved, which will be our future research direction.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020. 3

[2] Yuning Chai, Pei Sun, Jiquan Ngiam, Weiyue Wang, Benjamin Caine, Vijay Vasudevan, Xiao Zhang, and Dragomir Anguelov. To the point: Efficient 3d object detection in the range image with graph convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16000–16009, June 2021. 2

[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*, pages 1907–1915, 2017. 2, 6, 7

[4] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast point r-cnn. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, 2019. 5, 6

[5] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. *AAAI*, 2021. 1, 5, 6, 7

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 3

[7] Lue Fan, Xuan Xiong, Feng Wang, Naiyan Wang, and ZhaoXiang Zhang. Rangedet: In defense of range view for lidar-based 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2918–2927, October 2021. 2

[8] 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. 5, 6

[9] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R. Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, Apr 2021. 1, 3

[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/CVF Conference on Computer Vision and Pattern Recognition*, pages 11873–11882, 2020. 1, 2, 7

[11] 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. IEEE, 2018. 2, 7

[12] 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*, pages 12697–12705, 2019. 2, 4, 5, 6, 7, 8

[13] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3744–3753, 2019. 2, 3

[14] Bo Li. 3d fully convolutional network for vehicle detection in point cloud. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1513–1518. IEEE, 2017. 2

[15] Zhichao Li, Feng Wang, and Naiyan Wang. Lidar r-cnn: An efficient and universal 3d object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 5

[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*, pages 7345–7353, 2019. 7

[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. 7

[18] Zhidong Liang, Zehan Zhang, Ming Zhang, Xian Zhao, and Shiliang Pu. Rangeioudet: Range image based real-time 3d object detector optimized by intersection over union. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7140–7149, June 2021. 2

[19] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, October 2021. 2, 3

[20] Ze Liu, Zheng Zhang, Yue Cao, Han Hu, and Xin Tong. Group-free 3d object detection via transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2949–2958, October 2021. 2, 3

[21] Jiageng Mao, Yujing Xue, Minzhe Niu, Haoyue Bai, Jiashi Feng, Xiaodan Liang, Hang Xu, and Chunjing Xu. Voxel transformer for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3164–3173, October 2021. 1, 2, 3, 6, 7

[22] Ishan Misra, Rohit Girdhar, and Armand Joulin. An end-to-end transformer model for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2906–2917, October 2021. 1, 2, 3

[23] Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3d object detection with pointformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7463–7472, June 2021. 1, 2, 4

[24] Su Pang, Daniel Morris, and Hayder Radha. Clocs: Camera-lidar object candidates fusion for 3d object detection. In *2020*

IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 10386–10393. IEEE, 2020. 7

[25] Charles R. Qi, Or Litany, Kaiming He, and Leonidas J. Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 2, 5

[26] Charles R. Qi, Or Litany, Kaiming He, and Leonidas J. Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, 2019. 2

[27] 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*, pages 918–927, 2018. 1, 2, 7

[28] 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*, pages 652–660, 2017. 1

[29] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017. 2, 4, 5, 8

[30] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 12179–12188, October 2021. 3

[31] Hualian Sheng, Sijia Cai, Yuan Liu, Bing Deng, Jianqiang Huang, Xian-Sheng Hua, and Min-Jian Zhao. Improving 3d object detection with channel-wise transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2743–2752, October 2021. 1, 2, 3, 6, 7

[32] 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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 2, 6, 7

[33] 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*, pages 770–779, 2019. 1, 2, 5, 7, 8

[34] Martin Simony, Stefan Milzy, Karl Amendey, and Horst-Michael Gross. Complex-yolo: an euler-region-proposal for real-time 3d object detection on point clouds. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018. 2

[35] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020. 5, 6

[36] OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. https://github.com/open-mmlab/OpenPCDet, 2020. 6

[37] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, volume 139, pages 10347–10357, July 2021. 3

[38] 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*, pages 5998–6008, 2017. 3

[39] Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. Fast transformers with clustered attention, 2020. 3

[40] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity, 2020. 3

[41] Qian Xie, Yu-Kun Lai, Jing Wu, Zhoutao Wang, Yiming Zhang, Kai Xu, and Jun Wang. Mlcvnet: Multi-level context votenet for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1

[42] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 1, 2, 4, 5, 6, 7, 8

[43] 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*, pages 7652–7660, 2018. 2

[44] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1, 2

[45] 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)*, 2019. 1, 2, 5, 7

[46] Jin Hyeok Yoo, Yecheol Kim, Jisong Kim, and Jun Won Choi. 3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object detection. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16*, pages 720–736. Springer, 2020. 7

[47] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H.S. Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16259–16268, October 2021. 1, 2, 3

[48] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip H.S. Torr, and Li Zhang. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6881–6890, June 2021. 3

[49] Wu Zheng, Weiliang Tang, Sijin Chen, Li Jiang, and Chi-Wing Fu. Cia-ssd: Confident iou-aware single-stage object detector from point cloud. *AAAI*, 2021. 1

[50] 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, 2019. 6

[51] 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*, pages 4490–4499, 2018. 1, 2, 4, 6, 7