Voxel Transformer for 3D Object Detection

Jiageng Mao [1*] Yujing Xue [2*] Minzhe Niu [3] Haoyue Bai [4] Jiashi Feng [2]
Xiaodan Liang [5] Hang Xu [3†] Chunjing Xu [3]

Abstract

We present Voxel Transformer (VoTr), a novel and effective voxel-based Transformer backbone for 3D object detection from point clouds. Conventional 3D convolutional backbones in voxel-based 3D detectors cannot efficiently capture large context information, which is crucial for object recognition and localization, owing to the limited receptive fields. In this paper, we resolve the problem by introducing a Transformer-based architecture that enables long-range relationships between voxels by self-attention. Given the fact that non-empty voxels are naturally sparse but numerous, directly applying standard Transformer on voxels is non-trivial. To this end, we propose the sparse voxel module and the submanifold voxel module, which can operate on the empty and non-empty voxel positions effectively. To further enlarge the attention range while maintaining comparable computational overhead to the convolutional counterparts, we propose two attention mechanisms for multi-head attention in those two modules: Local Attention and Dilated Attention, and we further propose Fast Voxel Query to accelerate the querying process in multi-head attention. VoTr contains a series of sparse and submanifold voxel modules, and can be applied in most voxel-based detectors. Our proposed VoTr shows consistent improvement over the convolutional baselines while maintaining computational efficiency on the KITTI dataset and the Waymo Open dataset.

1. Introduction

3D object detection has received increasing attention in autonomous driving and robotics. Detecting 3D objects from point clouds remains challenging to the research community, mainly because point clouds are naturally sparse and unstructured. Voxel-based detectors transform irregular point clouds into regular voxel-grids and show superior performance in this task. In this paper, we propose Voxel Transformer (VoTr), an effective Transformer-based backbone that can be applied in most voxel-based detectors to further enhance detection performance.

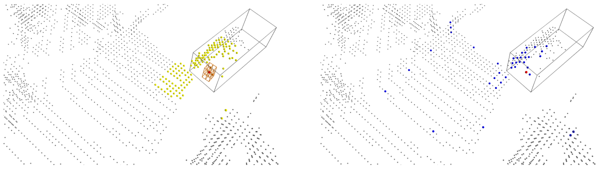

(a) 3D convolutional network (b) Voxel Transformer

Figure 1. Illustration of the receptive field obtained by the 3D convolutional network and our proposed VoTr. In (a), the orange cube denotes a single 3D convolutional kernel, and the yellow voxels are covered by the maximum receptive field centered at the red voxel. In (b), the red voxel denotes a querying voxel, and the blue voxels are the respective attending voxels for this query in voxel attention. Our observation is that a single self-attention layer in VoTr can cover a larger region than the whole convolutional backbone, and it can also maintain enough fine-grained 3D structures.

Previous approaches can be divided into two branches. Point-based approaches [26, 19, 34, 35] directly operate and generate 3D bounding boxes on point clouds. Those approaches generally apply point operators [23, 16] to extract features directly from point clouds, but suffer from the sparse and non-uniform point distribution and the time-consuming process of sampling and searching for neighboring points. Alternatively, voxel-based approaches [43, 33, 37, 5, 36] first rasterize point clouds into voxels and apply 3D convolutional networks to extract voxel features, and then voxels are transformed into a Bird-Eye-View (BEV) feature map and 3D boxes are generated on the BEV map. Compared with the point-based methods which heavily rely on time-consuming point operators, voxel-based approaches are more efficient with sparse convolutions, and can achieve state-of-the-art detection performance.

The 3D sparse convolutional network is a crucial component in most voxel-based detection models. Despite its advantageous efficiency, the 3D convolutional backbones cannot capture rich context information with limited receptive fields, which hampers the detection of 3D objects that have only a few voxels. For instance, with a commonly-used 3D convolutional backbone [33] and the voxel size as $(0.05m, 0.05m, 0.1m)$ on the KITTI dataset, the maximum receptive field in the last layer is only

[*] Equal contribution. [1] The Chinese University of Hong Kong [2] National University of Singapore [3] Huawei Noah's Ark Lab [4] HKUST [5] Sun Yat-Sen University [†] Corresponding author: xu.hang@huawei.com

$(3.65m, 3.65m, 7.3m)$, which can hardly cover a car with the length over $4m$. Enlarging the receptive fields is also intractable. The maximum theoretical receptive field of each voxel is roughly proportional to the product of the voxel size V, the kernel size K, the downsample stride S, and the layer number L. Enlarging V will lead to the high quantization error of point clouds. Increasing K leads to the cubic growth of convoluted features. Increasing S will lead to a low-resolution BEV map which is detrimental to the box prediction, and increasing L will add much computational overhead. Thus it is computationally extensive to obtain large receptive fields for the 3D convolutional backbones. Given the fact that the large receptive field is heavily needed in detecting 3D objects which are naturally sparse and incomplete, a new architecture should be designed to encode richer context information compared with the convolutional backbone.

Recently advances [6, 2, 41] in 2D object classification, detection, and segmentation show that Transformer is a more effective architecture compared with convolutional neural networks, mainly because long-range relationships between pixels can be built by self-attention in the Transformer modules. However, directly applying standard Transformer modules to voxels is infeasible, mainly owing to two facts: 1) Non-empty voxels are sparsely distributed in a voxel-grid. Different from pixels which are densely placed on an image plane, non-empty voxels only account for a small proportion of total voxels, *e.g*., the non-empty voxels normally occupy less than 0.1% of the total voxel space on the Waymo Open dataset [29]. Thus instead of performing self-attention on the whole voxel-grids, special operations should be designed to only attend to those non-empty voxels efficiently. 2) The number of non-empty voxels is still large in a scene, *e.g*., there are nearly $90k$ non-empty voxels generated per frame on the Waymo Open dataset. Therefore applying fully-connected self-attention like the standard Transformer is computationally prohibitive. New methods are thus highly desired to enlarge the attention range while keeping the number of attending voxels for each query in a small value.

To this end, we propose Voxel Transformer (VoTr), a Transformer-based 3D backbone that can be applied upon voxels efficiently and can serve as a better substitute for the conventional 3D convolutional backbones. To effectively handle the sparse characteristic of non-empty voxels, we propose the sparse voxel module and the submanifold voxel module as the basic building blocks of VoTr. The submanifold voxel modules operate strictly on the non-empty voxels, to retain the original 3D geometric structure, while the sparse voxel modules can output features at the empty locations, which is more flexible and can further enlarge the non-empty voxel space. To resolve the problem that non-empty voxels are too numerous for self-attention, we further propose two attention mechanisms: Local Attention and Dilated Attention, for multi-head attention in the sparse and submanifold voxel modules. Local Attention focuses on the neighboring region to preserve detailed information. Dilated Attention obtains a large attention range with only a few attending voxels, by gradually increasing the search step. To further accelerate the querying process for Local and Dilated Attention, we propose Fast Voxel Query, which contains a GPU-based hash table to efficiently store and lookup the non-empty voxels. Combining all the above components, VoTr significantly boosts the detection performance compared with the convolutional baselines, while maintains computational efficiency.

Our main contributions can be summarized as follows:

1) We propose Voxel Transformer, the first Transformer-based 3D backbone for voxel-based 3D detectors.

2) We propose the sparse and submanifold voxel module to handle the sparsity characteristic of voxels, and we further propose special attention mechanisms and Fast Voxel Query for efficient computation.

3) Our VoTr consistently outperforms the convolutional baselines and achieves the state-of-the-art performance with 74.95% LEVEL_1 mAP for vehicle and 82.09% mAP for moderate car class on the Waymo dataset and the KITTI dataset respectively.

2. Related Work

3D object detection from point clouds. 3D object detectors can be divided into 2 streams: point-based and voxel-based. Point-based detectors operate directly on raw point clouds to generate 3D boxes. F-PointNet [21] is a pioneering work that utilizes frustums for proposal generation. PointRCNN [26] generates 3D proposals from the foreground points in a bottom-up manner. 3DSSD [34] introduces a new sampling strategy for point clouds. Voxel-based detectors transform point clouds into regular voxel-grids and then apply 3D and 2D convolutional networks to generate 3D proposals. VoxelNet [43] utilizes a 3D CNN to extract voxel features from a dense grid. SECOND [33] proposes 3D sparse convolutions to efficiently extract voxel features. HVNet [36] designs a convolutional network that leverages the hybrid voxel representation. PV-RCNN [25] uses keypoints to extract voxel features for boxes refinement. Point-based approaches suffer from the time-consuming process of sampling and aggregating features from irregular points, while voxel-based methods are more efficient owing to the regular structure of voxels. Our Voxel Transformer can be plugged into most voxel-based detectors to further enhance the detection performance while maintaining computational efficiency.

Transformers in computer vision. Transformer [30] introduces a fully attentional framework for machine translation. Recently Transformer-based architectures surpass the con-

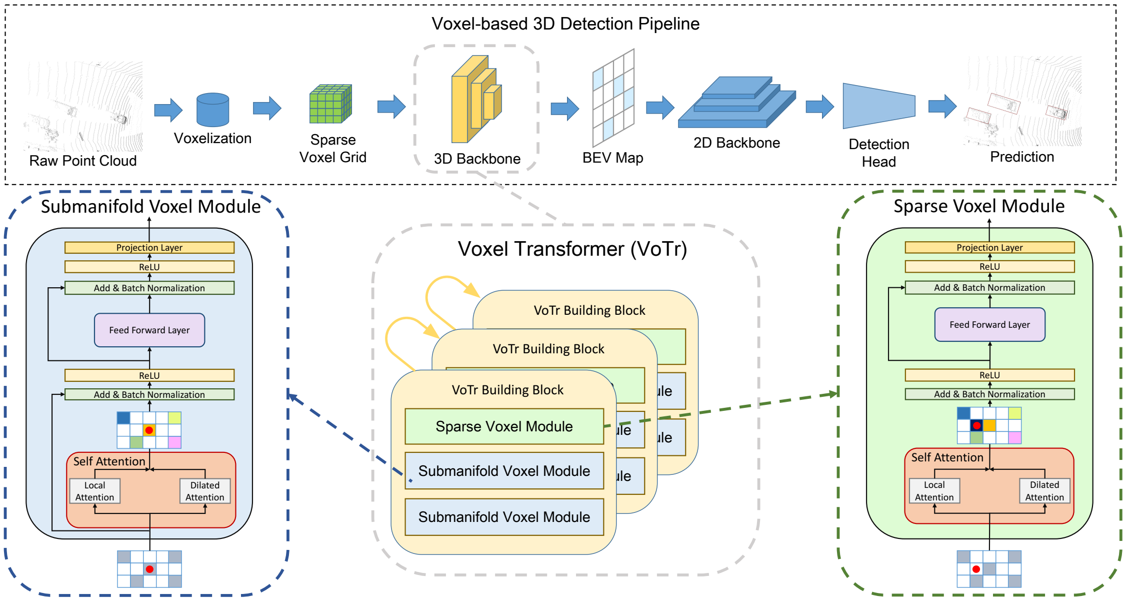


Figure 2. The overall architecture of Voxel Transformer (VoTr). VoTr is a Transformer-based 3D backbone that can be applied in most voxel-based 3D detection frameworks. It contains a series of sparse and submanifold voxel modules. Submanifold voxel modules perform multi-head self-attention strictly on the non-empty voxels, while sparse voxel modules can extract voxel features at empty locations.

volutional architectures and show superior performance in the task of image classification, detection and segmentation. Vision Transformer [6] splits an image into patches and feeds the patches into a Transformer for image classification. DETR [2] utilizes a Transformer-based backbone and a set-based loss for object detection. SETR [41] applies progressive upsampling on a Transformer-based backbone for semantic segmentation. MaX-DeepLab [31] utilizes a mask Transformer for panoptic segmentation. Transformer-based architectures are also used in 3D point clouds. Point Transformer [40] designs a novel point operator for point cloud classification and segmentation. Pointformer [19] introduces attentional operators to extract point features for 3D object detection. Our Voxel Transformer extends the idea of Transformers on images, and proposes a novel method to apply Transformer to sparse voxels. Compared with point-based Transformers, Voxel Transformer benefits from the efficiency of regular voxel-grids and shows superior performance in 3D object detection.

3. Voxel Transformer

In this section, we present Voxel Transformer (VoTr), a Transformer-based 3D backbone that can be applied in most voxel-based 3D detectors. VoTr can perform multi-head attention upon the empty and non-empty voxel positions though the sparse and submanifold voxel modules, and long-range relationships between voxels can be constructed by efficient attention mechanisms. We further propose Fast Voxel Query to accelerate the voxel querying process in multi-head attention. We will detail the design of each component in the following sections.

3.1. Overall Architecture

In this section, we introduce the overall architecture of Voxel Transformer. Similar to the design of the conventional convolutional architecture [33] which contains 3 sparse convolutional blocks and 6 submanifold convolutional blocks, our VoTr is composed of a series of sparse and submanifold voxel modules, as shown in Figure 2. In particular, we design 3 sparse voxel modules which downsample the voxel-grids by 3 times and output features at different voxel positions and resolutions as inputs. Each sparse voxel module is followed by 2 submanifold voxel modules, which keeps the input and output non-empty locations the same, to maintain the original 3D structure while enlarge receptive fields. Multi-head attention is performed in all those modules, and the attending voxels for each querying voxel in multi-head attention are determined by two special attention mechanisms: Local Attention and Dilated Attention, which captures well diverse context in different ranges. Fast Voxel Query is further proposed to accelerate the searching process for the non-empty voxels in multi-head attention.

Voxel features extracted by our proposed VoTr are then projected to a BEV feature map to generate 3D proposals, and the voxels and corresponding features can also be utilized on the second stage for RoI refinement. We note that our proposed VoTr is flexible and can be applied in most voxel-based detection frameworks [33, 25, 5].

3.2. Voxel Transformer Module

In this section, we present the design of sparse and submanifold voxel modules. The major difference between sparse and submanifold voxel modules is that submanifold voxel modules strictly operate on the non-empty voxels and extract features only at the non-empty locations, which maintains the geometric structures of 3D scenes, while sparse voxel modules can extract voxel features at the empty locations, which shows more flexibility and can expand the original non-empty voxel space according to needs. We first introduce self-attention on sparse voxels and then detail the design of sparse and submanifold voxel modules.

Self-attention on sparse voxels. We define a dense voxel-grid, which has N_{dense} voxels in total, to rasterize the whole 3D scene. In practice we only maintain those non-empty voxels with a $N_{sparse} \times 3$ integer indices array $\mathcal{V}$ and $N_{sparse} \times d$ corresponding feature array $\mathcal{F}$ for efficient computation, where N_{sparse} is the number of non-empty voxels and $N_{sparse} \ll N_{dense}$. In each sparse and submanifold voxel module, multi-head self-attention is utilized to build long-range relationships among non-empty voxels. Specifically, given a querying voxel i, the attention range $\Omega(i) \subseteq \mathcal{V}$ is first determined by attention mechanisms, and then we perform multi-head attention on the attending voxels $j \in \Omega(i)$ to obtain the feature f_i^{attend}. Let $f_i, f_j \in \mathcal{F}$ be the features of querying and attending voxels respectively, and $v_i, v_j \in \mathcal{V}$ be the integer indices of querying and attending voxels. We first transform the indices v_i, v_j to the corresponding 3D coordinates of the real voxel centers p_i, p_j by $p = r \cdot (v + 0.5)$, where r is the voxel size. Then for a single head, we compute the query embedding Q_i, key embedding K_j and value embedding V_j as:

$$Q_i = f_i W_q, K_j = f_j W_k + E_{pos}, V_j = f_j W_v + E_{pos}, \quad (1)$$

where W_q, W_k, W_v are the linear projection of query, key and value respectively, and the positional encoding E_{pos} can be calculated by:

$$E_{pos} = (p_i - p_j) W_{pos}. \quad (2)$$

Thus self-attention on voxels can be formulated as:

$$f_i^{attend} = \sum_{j \in \Omega(i)} \sigma(\frac{Q_i K_j}{\sqrt{d}}) \cdot V_j, \quad (3)$$

where $\sigma(\cdot)$ is the softmax normalization function. We note that self-attention on voxels is a natural 3D extension of standard 2D self-attention with sparse inputs and relative coordinates as positional embeddings.

Submanifold voxel module. The outputs of submanifold voxel modules are exactly at the same locations with the input non-empty voxels, which indicates its ability to keep the original 3D structures of inputs. In the submanifold voxel module, two sub-layers are designed to capture the long-range context information for each non-empty voxel. The first sub-layer is the self-attention layer that combines all the attention mechanisms, and the second is a simple feed-forward layer in [30]. Residual connections are employed around the sub-layers. The major differences between the standard Transformer module and our proposed module are as three folds: 1) We append an additional linear projection layer after the feed-forward layer for channel adjustment of voxel features. 2) We replace layer normalization with batch normalization. 3) We remove all the dropout layers in the module, since the number of attending voxels is already small and randomly rejecting some of those voxels hampers the learning process.

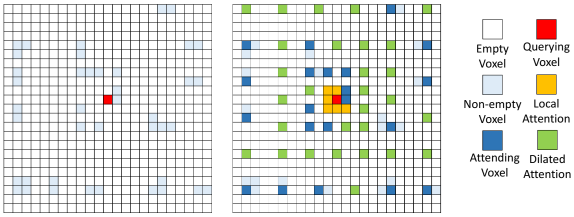


Figure 3. Illustration of Local and Dilated Attention. We note that this is a 2D example and can be easily extended to 3D cases. For each query (red), Local Attention (yellow) focuses on the local region while Dilated Attention (green) searches the whole space with gradually enlarged steps. The non-empty voxels (light blue) which meet the searching locations are selected as the attending voxels (dark blue).

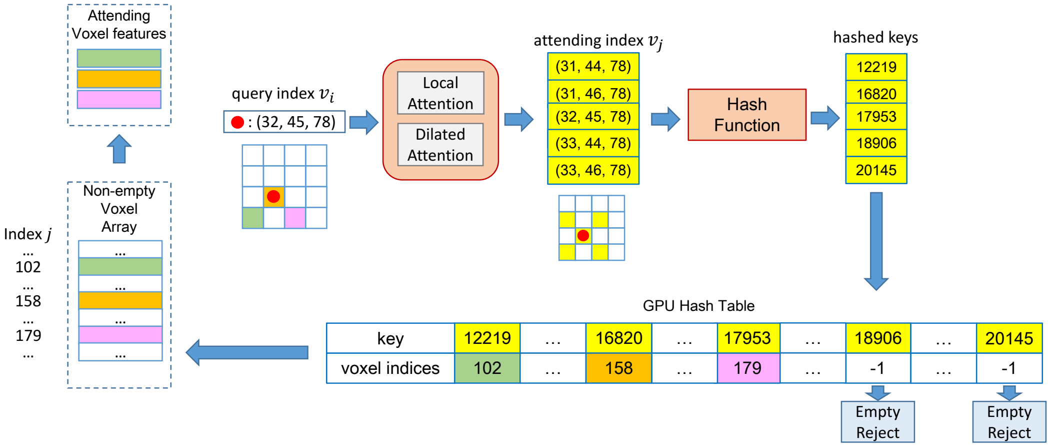
Sparse voxel module. Different from the submanifold voxel module which only operates on the non-empty voxels, the sparse voxel module can extract features for the empty locations, leading to the expansion of the original non-empty space, and it is typically required in the voxel downsampling process [33]. Since there is no feature f_i available for the empty voxels, we cannot obtain the query embedding Q_i from f_i. To resolve the problem, we give an approximation of Q_i at the empty location from the attending features f_j:

$$Q_i = \underset{j \in \Omega(i)}{\mathcal{A}} (f_j), \quad (4)$$

where the function $\mathcal{A}$ can be interpolation, pooling, *etc*. In this paper, we choose $\mathcal{A}$ as the maxpooling of all the attending features f_j. We also use Eq.3 to compute multi-head attention. The architecture of sparse voxel modules is similar to submanifold voxel modules, except that we remove the first residual connection around the self-attention layer, since the inputs and outputs are no longer the same.

3.3. Efficient Attention Mechanism

In this section, we delve into the design of the attention range $\Omega(i)$, which determines the attending voxels for each



key	12219	…	16820	…	17953	…	18906	…	20145
voxel indices	102	…	158	…	179	…	-1	…	-1

Figure 4. Illustration of Fast Voxel Query. For each querying index v_i, an attending voxel index v_j is determined by Local and Dilated Attention. Then we can lookup the non-empty index j in the hash table with hashed v_j as the key. Finally, the non-empty index j is used to gather the attending feature f_j from $\mathcal{F}$ for multi-head attention. Our proposed Fast Voxel Query is efficient both in time and in space and can significantly accelerate the computation of sparse voxel attention.

query i, and is a crucial factor in self-attention on sparse voxels. $\Omega(i)$ is supposed to satisfy the following requirements: 1) $\Omega(i)$ should cover the neighboring voxels to retain the fine-grained 3D structure. 2) $\Omega(i)$ should reach as far as possible to obtain a large context information. 3) the number of attending voxels in $\Omega(i)$ should be small enough, *e.g.* less than 50, to avoid heavy computational overhead. To tackle those issues, we take the inspiration from [39] and propose two attention mechanisms: Local Attention and Dilated Attention to control the attention range $\Omega(i)$. The designs of the two mechanisms are as follows.

Local Attention. We define $\varnothing(start, end, stride)$ as a function that returns the non-empty indices in a closed set $[start, end]$ with the step as $stride$. In the 3D cases, for example, $\varnothing((0,0,0),(1,1,1),(1,1,1))$ searches the set $\{(0,0,0),(0,0,1),(0,1,0),\cdots,(1,1,1)\}$ with 8 indices for the non-empty indices. In Local Attention, given a querying voxel v_i, the local attention range $\Omega_{local}(i)$ parameterized by R_{local} can be formulated as:

$$\Omega_{local}(i) = \varnothing(v_i - R_{local}, v_i + R_{local}, (1,1,1)), \quad (5)$$

where $R_{local} = (1,1,1)$ in our experiments. Local Attention fixes the $stride$ as $(1,1,1)$ to exploit every non-empty voxel inside the local range R_{local}, so that the fine-grained structures can be retained by Local Attention.

Dilated Attention. The attention range $\Omega_{dilated}(i)$ of Dilated Attention is defined by a parameter list $R_{dilated}$: $[(R^{(1)}_{start}, R^{(1)}_{end}, R^{(1)}_{stride}),\cdots,(R^{(M)}_{start}, R^{(M)}_{end}, R^{(M)}_{stride})]$, and the formulation of $\Omega_{dilated}(i)$ can be represented as:

$$\Omega_{dilated}(i) = \bigcup_{m=1}^{M} \varnothing(v_i - R^{(m)}_{end}, v_i + R^{(m)}_{end}, R^{(m)}_{stride}) \setminus \varnothing(v_i - R^{(m)}_{start}, v_i + R^{(m)}_{start}, R^{(m)}_{stride}), \quad (6)$$

where $\setminus$ is the set subtraction operator and the function $\bigcup$ takes the union of all the non-empty voxel sets. We note that $R^{(i)}_{start} < R^{(i)}_{end} \leq R^{(i+1)}_{start}$ and $R^{(i)}_{stride} < R^{(i+1)}_{stride}$, which means that we gradually enlarge the querying step $R^{(i)}_{stride}$ when search for the non-empty voxels which are more distant. This leads to a fact that we preserve more attending voxels near the query while still maintaining some attending voxels that are far away, and $R^{(i)}_{stride} > (1,1,1)$ significantly reduces the searching time and memory cost. With a carefully designed parameter list $R_{dilated}$, the attention range is able to reach more than $15m$ but the number of attending voxels for each querying voxel is still kept less than 50. It is worth noting that Local Attention can be viewed as a special case in Dilated Attention when $R_{start} = (0,0,0)$, $R_{end} = (1,1,1)$ and $R_{stride} = (1,1,1)$.

3.4. Fast Voxel Query

Searching for the non-empty attending voxels for each query is non-trivial in voxel self-attention. The sparse indices array $\mathcal{V}$ cannot arrange 3D sparse voxel indices in order in one dimension N_{sparse}. Thus we cannot directly obtain the index $j \in \Omega(i)$ in $\mathcal{V}$, even if we can easily get the corresponding integer voxel index $v_j \in \mathbb{R}^3$. Iterating all the N_{sparse} non-empty voxels to find the matched j takes $O(N_{sparse})$ time complexity for each querying process, and it is extremely time-consuming since N_{sparse} is normally $90k$ on the Waymo Open dataset. In [5] dense 3D voxel-grids are utilized to store j (or -1 if empty) for all the empty and non-empty voxels, but it is extremely memory-consuming to maintain those dense 3D voxel-grids, where the total number of voxels N_{dense} is more than 10^7. In this paper, we propose Fast Voxel Query, a new method that applies a GPU-based hash table to efficiently look up the attending non-empty voxels with little memory consumption.

An illustration of Fast Voxel Query is shown in Figure 4. Fast Voxel Query consists of four major steps: 1) we build a hash-table on GPUs which stores the hashed non-empty integer voxel indices v_j as keys, and the corresponding indices j for the array $\mathcal{V}$ as values. 2) For each query i, we apply Local Attention and Dilated Attention to obtain the attending voxel indices $v_j \in \Omega(i)$. 3) We look up the respective indices j for $\mathcal{V}$ using the hashed key v_j in the hash table, and v_j is judged as an empty voxel and rejected if the hash value returns -1. 4) We can finally gather the attending voxel indices v_j and features f_j from $\mathcal{V}$ and $\mathcal{F}$ with j for voxel self-attention. We note that all the steps can be conducted in parallel on GPUs by assigning each querying voxel i a separate CUDA thread, and in the third step, the lookup process for each query only costs $O(N_\Omega)$ time complexity, where N_Ω is the number of voxels in $\Omega(i)$ and $N_\Omega \ll N_{sparse}$.

To leverage the spatial locality of GPU memory, we build the hash table as a $N_{hash} \times 2$ tensor, where N_{hash} is the hash table size and $N_{sparse} < N_{hash} \ll N_{dense}$. The first row of the $N_{hash} \times 2$ hash table stores the keys and the second row stores the values. We use the linear probing scheme to resolve the collisions in the hash table, and the atomic operations to avoid the data race among CUDA threads. Compared with the conventional methods [24, 5], our proposed Fast Voxel Query is efficient both in time and in space, and our approach remarkably accelerates the computation of voxel self-attention.

4. Experiments

In this section, we evaluate Voxel Transformer on the commonly used Waymo Open dataset [29] and the KITTI [8] dataset. We first introduce the experimental settings and two frameworks based on VoTr, and then compare our approach with previous state-of-the-art methods on the Waymo Open dataset and the KITTI dataset. Finally, we conduct ablation studies to evaluate the effects of different configurations.

4.1. Experimental Setup

Waymo Open Dataset. The Waymo Open Dataset contains 1000 sequences in total, including 798 sequences (around $158k$ point cloud samples) in the training set and 202 sequences (around $40k$ point cloud samples) in the validation set. The official evaluation metrics are standard 3D mean Average Precision (mAP) and mAP weighted by heading accuracy (mAPH). Both of the two metrics are based on an IoU threshold of 0.7 for vehicles and 0.5 for other categories. The testing samples are split in two ways. The first way is based on the distances of objects to the sensor: $0-30m$, $30-50m$ and $> 50m$. The second way is according to the difficulty levels: LEVEL_1 for boxes with more than five LiDAR points and LEVEL_2 for boxes with at least one LiDAR point.

KITTI Dataset. The KITTI dataset contains 7481 training samples and 7518 test samples, and the training samples are further divided into the *train* split (3712 samples) and the *val* split (3769 samples). The official evaluation metric is mean Average Precision (mAP) with a rotated IoU threshold 0.7 for cars. On the *test* set mAP is calculated with 40 recall positions by the official server. The results on the *val* set are calculated with 11 recall positions for a fair comparison with other approaches.

We provide 2 architectures based on Voxel Transformer: VoTr-SSD is a single-stage voxel-based detector with VoTr as the backbone. VoTr-TSD is a two-stage voxel-based detector based on VoTr.

VoTr-SSD. ***Vo**xel **Tr**ansformer for **S**ingle-**S**tage **D**etector* is built on the commonly-used single-stage framework SECOND [33]. In particular, we replace the 3D sparse convolutional backbone of SECOND, with our proposed Voxel Transformer as the new backbone, and we still use the anchor-based assignment following [33]. Other modules and configurations are kept the same for a fair comparison.

VoTr-TSD. ***Vo**xel **Tr**ansformer for **T**wo-**S**tage **D**etector* is built upon the state-of-the-art two-stage framework PV-RCNN [25]. Specifically, we replace the 3D convolutional backbone on the first stage of PV-RCNN, with our proposed Voxel Transformer as the new backbone, and we use keypoints to extract voxel features from Voxel Transformer for the second stage RoI refinement. Other modules and configurations are kept the same for a fair comparison.

Implementation Details. VoTr-SSD and VoTr-TSD share the same architecture on the KITTI and Waymo dataset. The input non-empty voxel coordinates are first transformed into 16-channel initial features by a linear projection layer, and then the initial features are fed into VoTr for voxel feature extraction. The channels of voxel features are lifted up to 32 and 64 in the first and second sparse voxel module respectively, and other modules keep the input and output channels the same. Thus the final output features have 64 channels. The number of total attending voxels is set to 48 for each querying voxel, and the number of heads is set to 4 for multi-head attention. The GPU hash table size N_{hash} is set to $400k$. We would like readers to refer to supplementary

Methods	LEVEL_1	LEVEL_2	LEVEL_1 3D mAP/mAPH by Distance		
	3D mAP/mAPH	3D mAP/mAPH	0-30m	30-50m	50m-Inf
PointPillars [11]	63.3/62.7	55.2/54.7	84.9/84.4	59.2/58.6	35.8/35.2
MVF [42]	62.93/-	-	86.30/-	60.02/-	36.02/-
Pillar-OD [32]	69.8/-	-	88.5/-	66.5/-	42.9/-
AFDet [7]	63.69/-	-	87.38/-	62.19/-	29.27/-
LaserNet [17]	52.1/50.1	-	70.9/68.7	52.9/51.4	29.6/28.6
CVCNet [3]	65.2/-	-	86.80/-	62.19/-	29.27/-
StarNet [18]	64.7/56.3	45.5/39.6	83.3/82.4	58.8/53.2	34.3/25.7
RCD [1]	69.0/68.5	-	87.2/86.8	66.5/66.1	44.5/44.0
Voxel R-CNN [5]	75.59/-	66.59/-	92.49/-	74.09/-	53.15/-
SECOND* [33]	67.94/67.28	59.46/58.88	88.10/87.46	65.31/64.61	40.36/39.57
VoTr-SSD (ours)	**68.99/68.39**	**60.22/59.69**	**88.18/87.62**	**66.73/66.05**	**42.08/41.38**
PV-RCNN [25]	71.69/71.16	64.21/63.70	91.83/91.37	69.99/69.37	46.26/45.41
VoTr-TSD (ours)	**74.95/74.25**	**65.91/65.29**	**92.28/91.73**	**73.36/72.56**	**51.09/50.01**

Table 1. Performance comparison on the Waymo Open Dataset with 202 validation sequences for the vehicle detection. $\star$: re-implemented by ourselves with the official code.

materials for the detailed design of attention mechanisms.

Training and Inference Details. Voxel Transformer is trained along with the whole framework with the ADAM optimizer. On the KITTI dataset, VoTr-SSD and VoTr-TSD are trained with the batch size 32 and 16 respectively, and with the learning rate 0.01 for 80 epochs on 8 V100 GPUs. On the Waymo Open dataset, we uniformly sample 20% frames for training and use the full validation set for evaluation following [25]. VoTr-SSD and VoTr-TSD are trained with the batch size 16 and the learning rate 0.003 for 60 and 80 epochs respectively on 8 V100 GPUs. The cosine annealing strategy is adopted for the learning rate decay. Data augmentations and other configurations are kept the same as the corresponding baselines [33, 25].

4.2. Comparisons on the Waymo Open Dataset

We conduct experiments on the Waymo Open dataset to verify the effectiveness of our proposed VoTr. As is shown in Table 1, simply switching from the 3D convolutional backbone to VoTr gives 1.05% and 3.26% LEVEL_1 mAP improvements for SECOND [33] and PV-RCNN [25] respectively. In the range of 30-50m and 50m-Inf, VoTr-SSD gives 1.42% and 1.72% improvements, and VoTr-TSD gives 3.37% and 4.83% improvements on LEVEL_1 mAP. The significant performance gains in the far away area show the importance of large context information obtained by VoTr to 3D object detection.

Methods	Modality	AP_{3D} (%)		
		Easy	Mod.	Hard
MV3D [4]	R+L	74.97	63.63	54.00
AVOD-FPN [10]	R+L	83.07	71.76	65.73
F-PointNet [22]	R+L	82.19	69.79	60.59
MMF [13]	R+L	88.40	77.43	70.22
3D-CVF [38]	R+L	89.20	80.05	73.11
CLOCs [20]	R+L	88.94	80.67	77.15
ContFuse [14]	R+L	83.68	68.78	61.67
VoxelNet [43]	L	77.47	65.11	57.73
PointPillars [11]	L	82.58	74.31	68.99
PointRCNN [26]	L	86.96	75.64	70.70
Part-A^2 Net [27]	L	87.81	78.49	73.51
STD [35]	L	87.95	79.71	75.09
Patches [12]	L	88.67	77.20	71.82
3DSSD [34]	L	88.36	79.57	74.55
SA-SSD [9]	L	88.75	79.79	74.16
TANet [15]	L	85.94	75.76	68.32
Voxel R-CNN [5]	L	90.90	81.62	77.06
HVNet [36]	L	87.21	77.58	71.79
PointGNN [28]	L	88.33	79.47	72.29
SECOND [33]	L	84.65	75.96	68.71
VoTr-SSD (ours)	L	**86.73**	**78.25**	**72.99**
PV-RCNN [25]	L	90.25	81.43	76.82
VoTr-TSD (ours)	L	**89.90**	**82.09**	**79.14**

Table 2. Performance comparison on the KITTI *test* set with AP calculated by 40 recall positions for the car category. R+L denotes the methods that combines RGB data and point clouds. L denotes LiDAR-only approaches.

4.3. Comparisons on the KITTI Dataset

We conduct experiments on the KITTI dataset to validate the efficacy of VoTr. As is shown in the Table 2, VoTr-SSD and VoTr-TSD brings 2.29% mAP and 0.66% mAP improvement on the moderate car class on the KITTI *val* split. For the hard car class, VoTr-TSD achieves 79.14% mAP, outperforming all the previous approaches by a large margin, which indicates the long-range relationships between voxels captured by VoTr is significant for detecting 3D objects that only have a few points. The results on the *val* split in Table 3 show that VoTr-SSD and VoTr-TSD outper-

Methods	AP_{3D} (%)		
	Easy	Mod.	Hard
PointRCNN [26]	88.88	78.63	77.38
STD [35]	89.70	79.80	79.30
3DSSD [34]	89.71	79.45	78.67
VoxelNet [43]	81.97	65.46	62.85
Voxel R-CNN [5]	89.41	84.52	78.93
PointPillars [11]	86.62	76.06	68.91
Part-A^2 Net [27]	89.47	79.47	78.54
TANet [15]	87.52	76.64	73.86
SA-SSD [9]	90.15	79.91	78.78
SECOND [33]	87.43	76.48	69.10
VoTr-SSD (ours)	**87.86**	**78.27**	**76.93**
PV-RCNN [25]	89.35	83.69	78.70
VoTr-TSD (ours)	**89.04**	**84.04**	**78.68**

Table 3. Performance comparison on the KITTI *val* split with AP calculated by 11 recall positions for the car category.

form the baseline methods by 1.79% and 0.35% mAP for the moderate car class. Observations on the KITTI dataset are consistent with those on the Waymo Open dataset.

4.4. Ablation Studies

Effects of Local and Dilated Attention. Table 4 indicates that Dilated Attention guarantees larger receptive fields for each voxel and brings 2.79% moderate mAP gain compared to using only Local Attention.

Effects of dropout in Voxel Transformer. Table 5 details the influence of different dropout rates to VoTr. We found that adding dropout layers in each module is detrimental to the detection performance. The mAP drops by 8.52% with the dropout probability as 0.3.

Effects of the number of attending voxels. Table 6 shows that increasing the number of attending voxels from 24 to 48 boosts the performance by 1.19%, which indicates that a voxel can obtain richer context information by involving more attending voxels in multi-head attention.

Comparisons on the model parameters. Table 7 shows that replacing the 3D convolutional backbone with VoTr reduces the model parameters by $0.5M$, mainly because the modules in VoTr only contain linear projection layers, which have only a few parameters, while 3D convolutional kernels typically contain a large number of parameters.

Comparisons on the inference speed. Table 8 shows that with carefully designed attention mechanisms and Fast Voxel Query, VoTr maintains computation efficiency with 14.65 Hz running speed for the single-stage detector. Replacing the convolutional backbone with VoTr only adds about 20 ms latency per frame.

Visualization of attention weights. Figure 5 shows that a querying voxel can dynamically select the features of attending voxels in a very large context range, which benefits the detection of objects that are sparse and incomplete.

Methods	L.A.	D.A.	AP$_{3D}$ (%)
(a)	√		75.48
(b)	√	√	**78.27**

Table 4. Effects of attention mechanisms on the KITTI val split. L.A.: Local Attention. D.A.: Dilated Attention.

Methods	Dropout probability	AP$_{3D}$ (%)
(a)	**0**	**78.27**
(b)	0.1	75.97
(c)	0.2	70.82
(d)	0.3	69.75

Table 5. Effects of dropout probabilities on the KITTI *val* split.

Methods	Number of attending voxels	AP$_{3D}$ (%)
(a)	24	77.08
(b)	32	77.72
(c)	**48**	**78.27**

Table 6. Effects of the number of attending voxels for each querying voxel on the KITTI *val* split.

Methods	Model parameters
SECOND [33]	$5.3M$
VoTr-SSD (ours)	**4.8M**
PV-RCNN [25]	$13.1M$
VoTr-TSD (ours)	**12.6M**

Table 7. Comparisons on the model parameters for different frameworks on the KITTI dataset.

Methods	Inference speed (Hz)
SECOND [33]	20.73
VoTr-SSD (ours)	14.65
PV-RCNN [25]	9.25
VoTr-TSD (ours)	7.17

Table 8. Comparisons on the inference speeds for different frameworks on the KITTI dataset. 48 attending voxels are used.

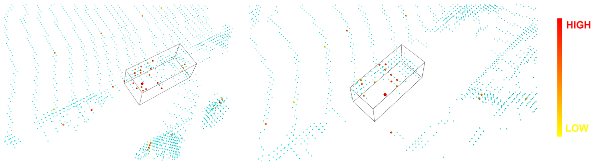


Figure 5. Visualization of attention weights for attending voxels.

5. Conclusion

We present Voxel Transformer, a general Transformer-based 3D backbone that can be applied in most voxel-based 3D detectors. VoTr consists of a series of sparse and submanifold voxel modules, and can perform self-attention on sparse voxels efficiently with special attention mechanisms and Fast Voxel Query. For future work, we plan to explore more Transformer-based architectures on 3D detection.

References

[1] Alex Bewley, Pei Sun, Thomas Mensink, Dragomir Anguelov, and Cristian Sminchisescu. Range conditioned dilated convolutions for scale invariant 3d object detection. *arXiv preprint arXiv:2005.09927*, 2020. 7

[2] 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. 2, 3

[3] Qi Chen, Lin Sun, Ernest Cheung, and Alan L Yuille. Every view counts: Cross-view consistency in 3d object detection with hybrid-cylindrical-spherical voxelization. *Advances in Neural Information Processing Systems*, 33, 2020. 7

[4] 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. 7

[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. *arXiv preprint arXiv:2012.15712*, 2020. 1, 3, 6, 7, 8

[6] 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. 2, 3

[7] Runzhou Ge, Zhuangzhuang Ding, Yihan Hu, Yu Wang, Sijia Chen, Li Huang, and Yuan Li. Afdet: Anchor free one stage 3d object detection. *arXiv preprint arXiv:2006.12671*, 2020. 7

[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. 6

[9] 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. 7, 8

[10] 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. 7

[11] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019. 7, 8

[12] Johannes Lehner, Andreas Mitterecker, Thomas Adler, Markus Hofmarcher, Bernhard Nessler, and Sepp Hochreiter. Patch refinement–localized 3d object detection. *arXiv preprint arXiv:1910.04093*, 2019. 7

[13] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 7345–7353, 2019. 7

[14] 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

[15] Zhe Liu, Xin Zhao, Tengteng Huang, Ruolan Hu, Yu Zhou, and Xiang Bai. Tanet: Robust 3d object detection from point clouds with triple attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11677–11684, 2020. 7, 8

[16] Jiageng Mao, Xiaogang Wang, and Hongsheng Li. Interpolated convolutional networks for 3d point cloud understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1578–1587, 2019. 1

[17] Gregory P Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K Wellington. Lasernet: An efficient probabilistic 3d object detector for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12677–12686, 2019. 7

[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. 7

[19] Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3d object detection with pointformer. *arXiv preprint arXiv:2012.11409*, 2020. 1, 3

[20] Su Pang, Daniel Morris, and Hayder Radha. Clocs: Camera-lidar object candidates fusion for 3d object detection. *arXiv preprint arXiv:2009.00784*, 2020. 7

[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*, pages 918–927, 2018. 2

[22] 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. 7

[23] 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

[24] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*, 2017. 6

[25] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020. 2, 3, 6, 7, 8

[26] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019. 1, 2, 7, 8

[27] 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*, 2020. 7, 8

[28] Weijing Shi and Raj Rajkumar. Point-gnn: Graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1711–1719, 2020. 7

[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/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020. 2, 6

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017. 2, 4

[31] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Max-deeplab: End-to-end panoptic segmentation with mask transformers. *arXiv preprint arXiv:2012.00759*, 2020. 3

[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. 7

[33] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 1, 2, 3, 4, 6, 7, 8

[34] 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*, pages 11040–11048, 2020. 1, 2, 7, 8

[35] 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/CVF International Conference on Computer Vision*, pages 1951–1960, 2019. 1, 7, 8

[36] Maosheng Ye, Shuangjie Xu, and Tongyi Cao. Hvnet: Hybrid voxel network for lidar based 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1631–1640, 2020. 1, 2, 7

[37] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. *arXiv preprint arXiv:2006.11275*, 2020. 1

[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. 7

[39] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *arXiv preprint arXiv:2007.14062*, 2020. 5

[40] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer. *arXiv preprint arXiv:2012.09164*, 2020. 3

[41] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. *arXiv preprint arXiv:2012.15840*, 2020. 2, 3

[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. PMLR, 2020. 7

[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*, pages 4490–4499, 2018. 1, 2, 7, 8