MsSVT: Mixed-scale Sparse Voxel Transformer for 3D Object Detection on Point Clouds

Anonymous Author(s)
Affiliation
Address
email

Abstract

3D object detection from the LiDAR point cloud is fundamental to autonomous driving. Large-scale outdoor scenes usually feature significant variance in instance scales, thus requiring features rich in long-range and fine-grained information to support accurate detection. Recent detectors leverage the power of window-based transformers to model long-range dependencies but, unfortunately, tend to blur out fine-grained details. To mitigate this gap, we present a novel Mixed-scale Sparse Voxel Transformer, named MsSVT, which can well capture both types of information simultaneously by the divide-and-conquer philosophy. Specifically, MsSVT explicitly divides attention heads into multiple groups, each in charge of attending to information within a particular range. The output from all the groups is merged to obtain the final mixed-scale features. Moreover, we provide a novel chessboard sampling strategy to reduce the computational complexity of applying a window-based transformer in 3D voxel space and implement the voxel sampling and gathering operations sparsely with a hash map to improve efficiency. Endowed by the powerful capability and high efficiency of modeling mixed-scale information, our single-stage detector built on top of MsSVT surprisingly outperforms state-of-the-art two-stage detectors on Waymo. Code will soon be available.

1 Introduction

3D object detection has received increasing attention due to its successful autonomous driving applications. Unlike 2D images with a regular structure of pixels, LiDAR point clouds are naturally irregular and unordered. Hence directly applying CNN-like operations [10, 11] to them can be difficult. To solve this, many researchers have rasterized point clouds into regular voxel grids [21] and employed 3D CNNs to extract high-dimensional voxel features. With the recent rise of vision transformer (ViT) [35] on 2D images, some attempts have been made to generalize global or more efficient window-based transformers to 3D voxels [20] or pillars [6]. These methods successfully seek long-range context by utilizing transformers' powerful abilities in modeling long-range information. However, they ignore that blindly increasing receptive fields would easily blur fine-grained details, especially in sparse 3D space, crucial to accurate object recognition and localization.

Standard window-based transformers update the features of queries in a local window by attending to keys from the same window. Hence simultaneously aggregating long-range context and fine-grained details require enlarging the window size to embrace local and distant voxels. Nevertheless, directly gathering all the voxels within the window as keys suffer a cubical growth in computational cost w.r.t. the window size. Some attempts [20] alleviate this by sampling only a certain number of key voxels. While a trivial sampling strategy quickly leads to sparse sampling of local voxels (Fig. 1 b), thus bias mainly on long-range context. To mitigate the above, we try to set up multiple key windows of varying sizes centered on a query window and sample the same number of local and distant key

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

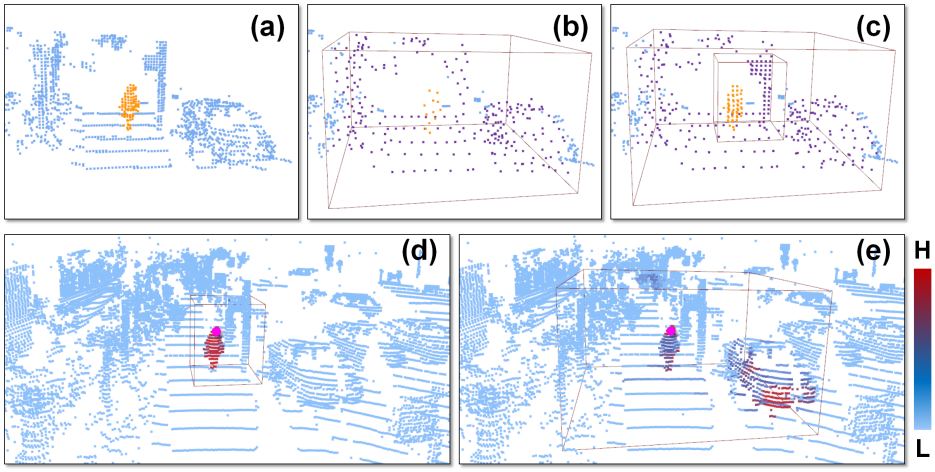

Figure 1: **Top**: In contrast to sampling key voxels from **b)** a single-scale 3D window in **a)** raw point clouds, our MsSVT samples key voxels from **c)** multi-scale windows, thus keeps finer granularity on the target object while covering large-scale neighborhood. **Bottom**: Different head groups accept keys sampled from windows of different scales, and are respectively responsible for obtaining **d)** fine-grained details and **e)** long-range context (reflected by higher attention weights), thus together contribute to accurate object detection collaboratively.

voxels from the smaller and larger windows separately. As a result, we can keep finer granularity in the local region to retain fine-grained details while collecting distant voxels roughly to enlarge the receptive field (Fig. 1 (c)).

With the sampled voxels ready, the next question is how to attend to voxels from different windows effectively and capture long-range context and fine-grained details simultaneously. We argue that the divide-and-conquer philosophy can satisfactorily resolve this issue. Specifically, inspired by the recent findings [35, 48, 23] that transformers learn different levels of self-attention by different heads, we propose a novel Mixed-scale Sparse Voxel Transformer (MsSVT), which explicitly divides the transformer heads into multiple groups. Different head groups accept voxels sampled from windows of different sizes, such that they are each in charge of capturing information of a particular scale. By combining the outputs from all the head groups, mixed-scale information, *i.e.*, long-range context, and fine-grained details can be well captured simultaneously. We also design a novel scale-aware relative position encoding strategy to adaptively adjust the position encoding used in each head group according to the range of keys. We provide some resulting attention maps by two different head groups (Fig. 1 (d), (e)). It is also worth mentioning that the mixed-scale attention meanwhile enables information exchange across local windows, which makes MsSVT more compact by saving additional shift window operation that is commonly required by window-based transformers [18, 6].

Moreover, to improve the efficiency of applying transformers in 3D voxel space, we strive to reduce computational costs in two ways. First, we propose a novel chessboard sampling (CBS) strategy to reduce the number of query voxels that need to be sampled within the query window, to reduce computational costs without losing information. Specifically, we partition the query window into chess-like spaced, and termed as "$\times$", "$\bigcirc$", "$\triangle$", "$\square$" positions separately. During each attention layer, only one specific position of voxels is sampled and updated by serving as queries, and the updates of the other voxels can be obtained by interpolation. Four positions are selected in the circular. Thus, we can update all the voxels without introducing deviation. Second, we take advantage of non-empty voxels' sparsity by performing mixed-scale window-based attention solely on non-empty sites in 3D space and parallelizing the search and feature gathering for non-empty voxels using hash mapping for further acceleration.

We build a 3D detector by replacing the original sparse 3D CNN backbone in SECOND [39] with our MsSVT and conduct extensive experiments on the large-scale Waymo open dataset [32]. Benefiting from the powerful capability of abstracting voxel features of mixed-scales, our single-stage detector based on MsSVT surprisingly outperforms state-of-the-art two-stage detectors. We summarize our contributions as follows:

- We present a novel Mixed-scale Sparse Voxel Transformer (MsSVT), which abstracts voxel features with both long-range context and fine-grained details simultaneously.
- We design an efficient chessboard sampling strategy to vastly reduce the computational cost of applying a voxel-based transformer in 3D space and sparsely implement all operations to improve efficiency.
- Our single-stage detector based on MsSVT even outperforms state-of-the-art two-stage detectors on Waymo.

2 Related work

Outdoor 3D detection on point cloud. The mainstream outdoor 3D detection models are based on voxel [2, 40, 41, 31, 15, 13, 49, 39] or pillar [14]. VoxelNet [49] utilizes a point net to aggregate the features within each voxel and is further processed by sparse 3D convolution and 2D convolution to generate detection results. SECOND [39] investigates an improved sparse convolution method, which dramatically improves the model's speed. Pointpillar [14] utilizes a pillar (voxel with infinite length in the vertical direction) to quantify the point cloud, so it can fully apply the mature 2D CNN, considering both efficiency and performance. Two-stage models [17, 29, 30, 26] aggregate the original point cloud or voxel features on the proposal of the single-stage model, which is used to adjust the bounding box. At present, they have state-of-the-art performance.

Vision transformer. Inspired by the great success of Transformer [35, 4] in NLP, some works employ it in the field of computer vision [5, 1, 18, 16, 50, 42, 36]. Swin-transformer [18] increases efficiency by limiting self-attention computation to non-overlapping local windows while allowing for cross-window connection. SSA [24] divides the attention head into multiple groups and aggregates image features with different granularity, respectively, which has achieved excellent performance. In the field of point cloud, Guo et al. [8] and Zhao et al. [47] introduce transformer paradigm for point cloud classification and segmentation. Recently, many methods [37, 46, 22, 9, 19] apply local self-attention mechanism on voxels to learn richer feature representations. Our model extends window attention to 3D voxels and flexibly combines different window sizes to capture multi-scale and multi-granularity features while shunting them to different attention heads in SSA.

Voxel transformer backbone in 3D detection. VoTr [20] introduces two kinds of sparse voxel attention, including local attention and extended attention. Each voxel serves as a query and attends with the neighbor voxels. We optimize the sparse operation and introduce window attention, which significantly improves efficiency and performance. The recent SST [6] adopts a single-stride design and swin-transformer structure, which performs well on small objects. However, SST is implemented based on a pillar. The single window size is not conducive to capturing multi-scale features, resulting in the low performance of vehicles when multiple categories are detected simultaneously. In contrast, our MsSVT can capture fine and global features and better performance on large and small objects.

3 Method

In this section, We first detail the pipeline of the Mixed-scale Sparse Voxel Transformer (MsSVT), then move to the novel chessboard sampling strategy and efficient sparse implementation, and build a complete 3D detector based on the above components.

3.1 Mixed-scale Sparse Voxel Transformer

Fig. 2 illustrates the overall framework of our proposed MsSVT, consisting of two core modules: a balanced multi-window sampling module and a scale-aware head attention module. We first get the query, key, and value voxels balanced sampled from multiple windows through the former and then feed them into our scale-aware head attention to simultaneously capture mixed-scale information. We also add the scale-aware relative position encoding to the attention mechanism so that even the same relative position information can play various roles in different scales.

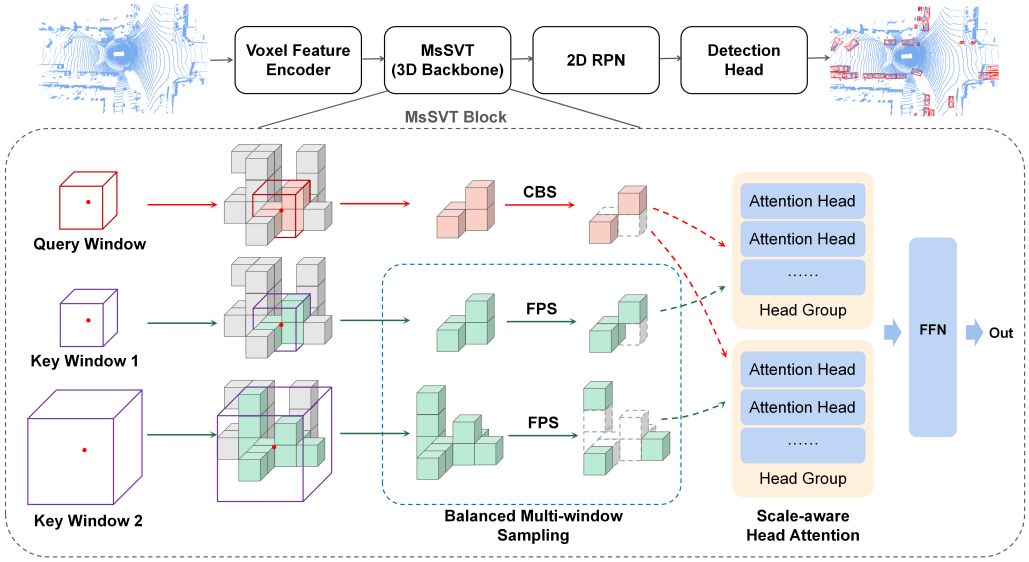

Figure 2: **Top**: overall architecture of our detection network. **Bottom**: details of our MsSVT block. In MsSVT block, we gather the voxels with the query window size (e.g. $\boldsymbol{s}_0 = (2, 2, 2)$) and the key window sizes (e.g. $\boldsymbol{s}_1 = (2, 2, 2)$ and $\boldsymbol{s}_2 = (4, 4, 4)$), respectively. Then, we apply CBS to obtain sampled queries, while employ FPS to get sampled keys and values of same number (e.g. $N_K = 3$) among all key windows. Finally, we feed the same query and the different keys into different attention head groups, respectively.

3.1.1 Balanced Multi-window Sampling

Given a series of window sizes $\{\boldsymbol{s}_k | \boldsymbol{s}_k \in \mathbb{Z}^3\}_{k=0}^{M}$, where $\boldsymbol{s}_0$ denotes the query window size and $\boldsymbol{s}_{1,...,M}$ denote M different key window sizes from small to large. Let $\boldsymbol{\mathcal{V}} = \{\boldsymbol{v}_i | \boldsymbol{v}_i = (\boldsymbol{x}_i, \boldsymbol{f}_i)\}_{i=1}^{|\mathcal{V}|}$ be the input voxel set, where $\boldsymbol{x}_i \in \mathbb{Z}^3$ denotes voxel coordinates and $\boldsymbol{f}_i \in \mathbb{R}^C$ denotes voxel features. We first partition the voxel set into non-overlap 3D windows $\boldsymbol{s}_0$ by finding non-empty window centers $\{\boldsymbol{c}_i | \boldsymbol{c}_i \in \mathbb{Z}^3\}_{i=0}^{L}$, where L is the number of non-empty windows. To get query voxels, we can simply gather all the non-empty voxels $\boldsymbol{\mathcal{V}}_{\boldsymbol{c}_i, \boldsymbol{s}_0}$ centered on $\boldsymbol{c}_i$ within query window $\boldsymbol{s}_0$ to guarantee that every voxel can serve as a query and be updated after one attention layer. We also prsent another novel chessboard sampling strategy for the query voxel sampling which will be discussed in Section 3.2. For easy understanding, readers can regrard all the non-empty voxels in each query window as the queries.

As for the key voxels, instead of sampling in a single large window at once like previous methods [20], which inevitably bias on local or distant voxels, we simultaneously research the neighboors for each center $\boldsymbol{c}_i$ with different window size $\boldsymbol{s}_k$ and gather no more than m non-empty voxels $\boldsymbol{\mathcal{V}}_{\boldsymbol{c}_i, \boldsymbol{s}_k} = \{\boldsymbol{v}_j | -\boldsymbol{s}_k < \boldsymbol{x}_j - \boldsymbol{c}_i < \boldsymbol{s}_k\}_{j=1}^{m}$, where m is a pre-defined number for all key windows. Furthermore, to reduce the computational cost and keep balanced sampling, we adopt farthest point sampling (FPS) to uniformly sample N_K voxels for each window $\boldsymbol{s}_k$ to obatain the final key voxels $\boldsymbol{\mathcal{V}}_{\boldsymbol{c}_i, \boldsymbol{s}_k}^{fps}, k = 1, ..., M$ at different scales, where N_K is a predefined maximum number of sampled voxels. Benefit from the multi-window sampling combined with the uniform FPS operation, we can achieve more balanced key voxels sampling from various scales which is crucial to capturing mixed-scale information.

3.1.2 Scale-aware Head Attention

Once receiving the query voxels $\boldsymbol{\mathcal{V}}_{\boldsymbol{c}_i, \boldsymbol{s}_0} = (\boldsymbol{X}_0, \boldsymbol{F}_0)$ and sampled key voxels $\boldsymbol{\mathcal{V}}_{\boldsymbol{c}_i, \boldsymbol{s}_k}^{fps} = (\boldsymbol{X}_k, \boldsymbol{F}_k), k = 1, ..., M$ at different scales, where $\boldsymbol{X}_0 \in \mathbb{Z}^{N_Q \times 3}$, $\boldsymbol{F}_0 \in \mathbb{R}^{N_Q \times C}$ denote query voxel coordinates and features, $\boldsymbol{X}_k \in \mathbb{Z}^{N_K \times 3}$, $\boldsymbol{F}_k \in \mathbb{R}^{N_K \times C}$ denote key voxel coordinates and features, we first get queries $\boldsymbol{Q}$, keys $\{\boldsymbol{K}_k\}_{k=1}^{M}$ and values $\{\boldsymbol{V}_k\}_{k=1}^{M}$ as

$$\boldsymbol{Q},\ \boldsymbol{K}_k,\ \boldsymbol{V}_k = \boldsymbol{F}_0 \boldsymbol{W}^Q,\ \boldsymbol{F}_k \boldsymbol{W}^K,\ \boldsymbol{F}_k \boldsymbol{W}^V, \quad k = 1, ..., M \tag{1}$$

where $\boldsymbol{W}^Q$,$\boldsymbol{W}^K$,$\boldsymbol{W}^V \in \mathbb{R}^{C\times C}$ are linear projection. To make our attention module aware to different scales, we divide the attention heads into M multiple groups and assign keys from different windows to different head groups so that the queries $\boldsymbol{Q}$ should only attend $\boldsymbol{K}_k$ in the k-th head group. We also split the feature channel into M parts and explicitly make each part represent the information of a specific scale. We denote $\boldsymbol{Q}^k = \boldsymbol{Q}[:,\ (k-1)\times C/M : k\times C/M],\ k = 1, ..., M$ as the k-th scale features for queries and the same is true for $\boldsymbol{K}$ and $\boldsymbol{V}$. So for the k-th head group, we can get the attended feature as:

$$\tilde{\boldsymbol{Y}}^{\boldsymbol{k}} = \mathbf{MHA}(\boldsymbol{Q}^k, \boldsymbol{K}^k, \boldsymbol{V}^k, \mathbf{RPE}(\boldsymbol{X}_0, \boldsymbol{X}_k)), \tag{2}$$

where $\mathbf{MHA}(\cdot)$ denotes multi-head-group attention and $\mathbf{RPE}(\cdot)$ stands for our relative position encoding which will be discussed in Section 3.1.3. Each window size corresponds to a "head group" that includes one or more attention heads. Finally, we concatenate the output of all heads group$\{\tilde{\boldsymbol{Y}}^k\}_{k=1}^M$ to $\tilde{\boldsymbol{Y}} \in \mathbb{R}^{L\times N_Q\times C}$, and feed into FFN, where L denotes the total number of non-empty query windows. After that, we obtain the mixed-scale features $\boldsymbol{Y}$:

$$\tilde{\boldsymbol{Y}} = \mathbf{CAT}(\tilde{\boldsymbol{Y}}^1, ..., \tilde{\boldsymbol{Y}}^M), \tag{3}$$

$$\boldsymbol{Y} = \mathbf{MLP}(\mathbf{LN}(\tilde{\boldsymbol{Y}}))) + \tilde{\boldsymbol{Y}}, \tag{4}$$

3.1.3 Scale-aware Relative Position Encoding

The 3D point cloud feature generally contains the original coordinates information, which voxels will inherit. However, the fine-grained location information may be blurred with the deepening of the network, so the relative position encoding is necessary. Furthermore, since MsSVT can extract multi-scale features, the position-coding should differ at different scales, even for the same relative position. In light of these, we adopt an adaptive, scale-aware relative position encoding strategy inspired by [25, 38, 44].

Specifically, according to the size of the maximum key window, we establish a learnable embedding table $\boldsymbol{T}^k \in \mathbb{R}^{D\times R}$ for k-th head group like [18], where R is the number of possible relative position pairs and D is the feature dimension. Then we can get the query relative position bias $\boldsymbol{B}_Q^k \in \mathbb{R}^{L\times N_Q\times N_K}$ as:

$$\boldsymbol{B}_Q^k = \mathcal{G}(\boldsymbol{Q}^k\boldsymbol{T}^k, \boldsymbol{I}^k), \tag{5}$$

where $\boldsymbol{I}^k \in \mathbb{Z}^{L\times N_Q\times N_K}$ means the table indices corresponding to the actual relative position of the queries and keys, $\mathcal{G}(\cdot)$ means the operation of gather features via indices. We get the $\boldsymbol{B}_K^k$ in the same way. Finally, the bias acts directly on the attention, and Eq. (2) can be rewrited as:

$$\tilde{\boldsymbol{Y}}^{\boldsymbol{k}} = \sigma(\frac{\boldsymbol{Q}^k\boldsymbol{K}^{k\mathsf{T}}}{\sqrt{D}} + \boldsymbol{B}_Q^k + \boldsymbol{B}_K^k)\boldsymbol{V}^k, \tag{6}$$

where $\sigma(\cdot)$ represents softmax. In this way, position embeddings can be adjusted adaptively according to different scales to guide our scale-ware head attention better.

3.2 Chessboard Sampling

In Section 3.1, the prerequisite for us to sample key voxels to a specific number is that we do not need to preserve all the key voxels and only need to select some representative ones. However, things are different for queries. On the one hand, we should maintain and update every query voxel after an attention layer; otherwise, the information could be irreversibly lost, and sampling queries seems impossible. On the other hand, the computational cost grows cubically with window size, and it would bring unacceptable memory costs if we can not reduce the number of queries.

However, since the non-empty voxel positions are not changed during a sequential of attention blocks, an effective way is to iteratively sample a subset of voxels from all the queries and then use the sampled voxels to update those unsampled. In the light of this, we present Chessboard Sampling (CBS) as shown in Fig. 3. In an attention module, we select one "×", "◯", "△", "□" as the sampling location. The sampled non-empty voxels are sent to the subsequent network for updating. The updates of other non-empty voxels can be obtained by linear interpolation with their top k nearest neighbors (default by 3). Four positions will be selected in turn in stacked modules. In this way, we retain the original structure and cover all voxels as comprehensively as possible. Note that we can apply the interval to any axis to obtain the sampling rate of 1/2, 1/4, or 1/8. Generally, we apply it to the horizontal xy plane (like Fig. 3).

3.3 Sparse Implementation

We implement all our window center searching, window gathering, and balanced window sampling sparsely in cuda operations to leverage the natural sparsity of point clouds and improve efficiency. These operations are mainly based on a hash map which establishes the mapping from coordinate space to voxel index as in [20]. For example, for window gathering, we query each possible position wrt the given center within the window and retrieve the feature if the position is a valid key in our pre-built hash map. More details are available in the supplementary materials.

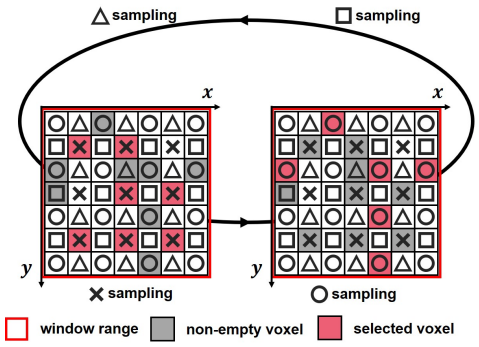

Figure 3: Diagram of chessboard sampling.

3.4 Detector Establishment

As shown in Fig 2, we build our 3D backbone by stacking Mixed-scale Sparse Voxel Transformer (MsSVT) blocks. It is worth noting that both the query and key window size in the last block of MsSVT are set to $(1, 1, \infty)$ so as to compressing the 3D voxels into 2D feature map, where the query is the average of all the voxel features within the pillar window. We replace the 3D backbone with our MsSVT in SECOND [39] and keep the other architecture the same except for abandoning the down-sampling process because MsSVT can already capture features at different scales. The input point cloud is first converted into regular voxels and fed into our MsSVT to get the mixed-scale voxel features. Next, features are compressed vertically and sent to the following 2D RPN and detection head, outputting detection results.

4 Experiments

In this section, we provide architectural details of MsSVT and compare our model with recent state-of-the-art detectors on Waymo Open Dataset [32] and KITTI [7]. Thorough ablation studies and in-depth analysis are further provided to validate our design choices.

4.1 Architectural Details

MsSVT includes four regular MsSVT attention blocks and an additional special one where the windows are set as 1×1 pillars as mentioned in 3.4. The query window size of the normal MsSVT attention blocks is set to $(3, 3, 5)$, and the key windows are set to $(3, 3, 5)$ and $(7, 7, 7)$. We divide total 8 attention heads into 2 groups in the scale-aware head attention layer. The queries are obtained via alternating chessboard sampling with 0.25 sampling ratio, and the keys are balanced sampled with a maximum number of 32 in each window. We use center head [45] as the detection head to generate a single-stage detection bounding box. In addition, we also provide a two-stage detector with CT3D [26]. Please refer to OpenPCDet [34] for more details since we conduct our experiments with this toolbox.

4.2 Results on Waymo

Setup. We first evaluate our model on large-scale Waymo Open Dataset [32]. The input is a single-frame point cloud with $150\text{m}\times150\text{m}$ detection range. We set the detection range in the horizontal and the vertical direction as $[-75.2\text{m}, 75.2\text{m}]$ and $[-2.0\text{m}, 4.0\text{m}]$, respectively. The voxel size is $(0.4\text{m}, 0.4\text{m}, 0.6\text{m})$. We follow the same training strategy as in [20]. Specifically, we train the model on Tesla-V100 using the Adam [12] optimizer for 80 epochs on 20% Waymo data. We apply the cyclic decay scheme [39], by which the learning rate is increased from $1e{-}4$ to $1e{-}3$ during the first 40% epochs and further decreased to 1e-5 in the remaining epochs. We also provide the results of 30 epochs training on 100% data with the same optimizer and scheme. The evaluation metric is the 3D mean Average Precision (mAP) for difficulty levels of LEVEL 1 and LEVEL 2.

Table 1: Results on WOD validation set (train with **20**% Waymo data). SS: Single-stage model, TS: Two-stage model, SF: Single frame input. Note that some priors only publish results of single-class training, which is generally simpler than multi-class training.

Method	Reference	Vel_L1		Vel_L2		Ped_L1		Ped_L2		Cyc_L1		Cyc_L2	
		mAP	mAPH	mAP	mAPH	mAP	mAPH	mAP	mAPH	mAP	mAPH	mAP	mAPH
Single-Stage Methods													
SECOND [39]	Sensors 2018	70.96	70.34	62.58	62.02	65.23	54.24	57.22	47.49	57.13	55.62	54.97	53.53
PointPillar [14]	CVPR 2019	70.43	69.83	62.18	61.64	66.21	46.32	58.18	40.64	55.26	51.75	53.18	49.80
CenterPoint [45]	CVPR 2021	72.76	72.23	64.91	64.42	74.19	67.96	66.03	60.34	71.04	69.79	68.49	67.28
VOTR-SS [20]	ICCV 2021	68.99	68.39	60.22	59.69	–	–	–	–	–	–	–	–
RSN-SF [33]	CVPR 2021	75.10	74.60	66.00	65.50	–	–	–	–	–	–	–	–
MsSVT-SS (ours)	–	**77.18**	**76.67**	**68.75**	**68.28**	**80.25**	**73.05**	**72.88**	**66.14**	**73.75**	**72.53**	**70.96**	**69.79**
Two-Stage Methods													
Part-A2 [28]	TPAMI 2020	74.66	74.12	65.82	65.32	71.71	62.24	62.46	54.06	66.53	65.18	64.05	62.75
PV-RCNN [29]	CVPR 2020	75.95	75.43	68.02	67.54	75.94	69.40	67.66	61.62	70.18	68.98	67.73	66.57
Voxel-RCNN [3]	AAAI 2021	76.13	75.66	68.18	67.74	78.20	71.98	69.29	63.59	70.75	69.68	68.25	67.21
PV-RCNN++ [30]	ARXIV 2021	77.61	77.14	69.18	68.75	79.42	73.31	70.88	65.21	72.50	71.39	69.84	68.77
VOTR-TS [20]	ICCV 2021	74.95	74.25	65.91	65.29	–	–	–	–	–	–	–	–
CT3D [26]	ICCV 2021	76.30	–	69.04	–	–	–	–	–	–	–	–	–
MsSVT-TS (ours)	–	**78.41**	**77.91**	**69.74**	**69.17**	**82.34**	**76.77**	**74.71**	**69.36**	**75.74**	**74.65**	**73.72**	**72.64**

Table 2: Results on WOD validation set (train with **100**% Waymo data).

Method	Reference	Vel_L1		Vel_L2		Ped_L1		Ped_L2		Cyc_L1		Cyc_L2	
		mAP	mAPH	mAP	mAPH	mAP	mAPH	mAP	mAPH	mAP	mAPH	mAP	mAPH
Single-Stage Methods													
SECOND [39]	Sensors 2018	72.27	71.69	63.85	63.33	68.70	58.18	60.72	51.31	60.62	59.28	58.34	57.05
PointPillar [14]	CVPR 2019	71.57	70.99	63.06	62.54	70.61	56.70	62.85	50.24	64.36	62.27	61.95	59.93
SST-SS-SF [6]	CVPR 2022	73.57	–	64.80	–	80.01	–	71.66	–	70.72	–	68.01	–
MsSVT-SS (ours)	–	**77.83**	**77.32**	**69.53**	**69.06**	**80.39**	**73.61**	**73.00**	**66.65**	**75.17**	**73.99**	**72.37**	**71.24**
Two-Stage Methods													
Part-A2 [28]	TPAMI 2020	77.05	76.51	68.47	67.97	75.24	66.87	66.18	58.62	68.60	67.36	66.13	64.93
PV-RCNN [29]	CVPR 2020	78.00	77.50	69.43	68.98	79.21	73.03	70.42	64.72	71.46	70.27	68.95	67.79
PV-RCNN++ [30]	ARXIV 2021	79.25	78.78	70.61	70.18	81.83	76.28	73.17	68.00	73.72	72.66	71.21	70.19
MsSVT-TS (ours)	–	**79.35**	**78.86**	**70.65**	**70.23**	**82.41**	**77.04**	**74.74**	**69.57**	**77.12**	**76.01**	**74.98**	**74.07**

Main results. Our main results are shown in Table 1 and Table 2 . It is worth noting that our model detects three categories simultaneously, which is more challenging than detecting a single category. On 20% data, MsSVT-SS significantly outperforms other single-stage detectors, even though some methods are only trained on a single category. The single-stage results can almost match the state-of-the-art two-stage methods PV-RCNN++ [30] on the vehicle and even outperform 0.8-1.5 mAP on pedestrians and cyclists. MsSVT-TS achieves the best performance in all categories and has absolute superiority over pedestrians and cyclists.

On 100% data, our model also achieves state-of-the-art performance. Compared with the recent transformer work SST [6], which is adept at the small object detection, MsSVT not only takes into account all categories but also has better performance on small objects. This result proves MsSVT's superiority over general transformers in capturing multi-scale features.

Visualization. Fig. 4 shows some visualization of our results. As shown in Fig. 4 (a), in the range of $> 50m$ that the points are very sparse, the bounding boxes are still accurate, which means that MsSVT can rely on context information in the absence of fine-grained information. As shown in Fig. 4 (b), even in a complex scene with many different categories of objects at the same time, the model still perform well, which should be owed to the flexibility and robustness of MsSVT.

Analysis. We visualized some attention weights. As shown in Fig. 5, the smaller window model focuses on the local foreground information, while the larger window model tends to focus on the context information. The two windows can complement each other. This proves the ability of MsSVT to capture mixed-scale features.

Table 3: Results on KITTI validation set.

Method	Reference	3D Car (IoU=0.7)			3D Ped. (IoU=0.5)			3D Cyc. (IoU=0.5)		
		Easy	Mod	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
Single-Stage Methods										
SECOND [39]	Sensors 2018	86.46	77.28	74.65	61.63	56.27	52.60	80.10	62.69	59.71
PointPillar [14]	CVPR 2019	88.61	78.62	77.22	56.55	52.98	47.73	80.59	67.16	63.11
3DSSD [43]	CVPR 2020	88.55	78.45	77.30	58.18	54.32	49.56	86.25	70.49	65.32
VoTr-SS [20]	ICCV 2021	87.86	78.27	76.93	–	–	–	–	–	–
MsSVT-SS (ours)	–	**89.08**	**78.75**	**77.35**	**63.59**	**57.33**	**53.12**	**88.57**	**71.70**	**66.29**
Two-Stage Methods										
PointRCNN [27]	CVPR 2019	89.03	78.78	77.86	62.50	55.18	50.15	87.49	72.55	66.01
Part-A2 [28]	TPAMI 2020	88.48	78.96	78.36	**70.73**	**64.13**	**57.45**	88.18	73.35	**70.75**
PV-RCNN [29]	CVPR 2020	**89.35**	83.69	78.70	63.12	54.84	51.78	86.06	69.48	64.50
VoTr-TS [20]	ICCV 2021	89.04	84.04	78.68	–	–	–	–	–	–
MsSVT-TS (ours)	–	89.32	**84.66**	**78.94**	66.11	58.94	53.86	**92.49**	**73.60**	69.34

4.3 Results on KITTI

Setup. We further evaluate our model on KITTI [7]. Given an input point cloud, we reserve the points within the range of $[0\mathrm{m}, 70.4\mathrm{m}]$, $[-40.0\mathrm{m}, 40.0\mathrm{m}]$ and $[-3.0\mathrm{m}, 1.0\mathrm{m}]$ in the X, Y, and Z directions, respectively. The voxel size is set to $(0.32\mathrm{m}, 0.32\mathrm{m}, 0.4\mathrm{m})$ and other settings are same as 4.2. We train the model for 100 epochs with the Adam optimizer. The learning rate is 0.003, decayed by the cyclic scheme [39]. The evaluation metric is 3D mAP for three difficulty levels (easy, moderate, and hard).

Main results. Table 3 depicts our MsSVT achieves competitive performance on all the three categories. Specifically, on *Car*, MsSVT-SS beats the superior VoTr [20] by 0.5 mAP. Meanwhile, on *Pedestrian* and *Cyclist*, MsSVT-SS even outperforms some preeminent two-stage detectors. Moreover, our two-stage MsSVT-TS further increases the lead. The results on KITTI demonstrate that our MsSVT has a good generalization ability to adapt to different datasets.

4.4 Ablation Study

We perform ablation studies to validate our design choices. All models are trained for 12 epochs on 20% Waymo data. More implementation details and ablations on hyperparameters are available in supplementary materials.

Balanced multi-window sampling. We first validate our balanced multi-window sampling strategy in Table 4. The base model, listed in the first row, employs 3D version of standard window-based attention [18] (with window size $(3, 3, 5)$ and without shift window scheme), adopts dilated key sampling strategy [20], and gathers all non-empty voxels within the window as queries. We build a model variant by simply replacing the dilated key sampling in the base model with our balanced multi-window sampling strategy. The results in the second row shows sampling key voxels from windows of multiple sizes brings noticeable performance gains, $i.e.$, $71.66 \rightarrow 73.44$ on *Pedestrian* and $63.31 \rightarrow 66.88$ on *Cyclist*.

Table 4: Ablations on different components of MsSVT. BMS: Balanced Multi-window Sampling, SHA: Scale-aware Head Attention, SRPE: Relative Position Encoding.

BMS	SHA	SRPE	Veh / Ped / Cyc
			$69.51/71.66/63.31$
✓			$71.24/73.44/66.88$
✓	✓		$71.96/75.08/67.16$
✓	✓	✓	$72.37/75.99/67.90$

Scale-aware head attention. Table 4 also validates the effectiveness of our scale-aware head attention module. By comparing the second and the third rows, we find that making different attention head groups specialized in different scales helps to capture mixed-scale features and thus brings significant performance gains. We further visualize some attention maps by different head groups in

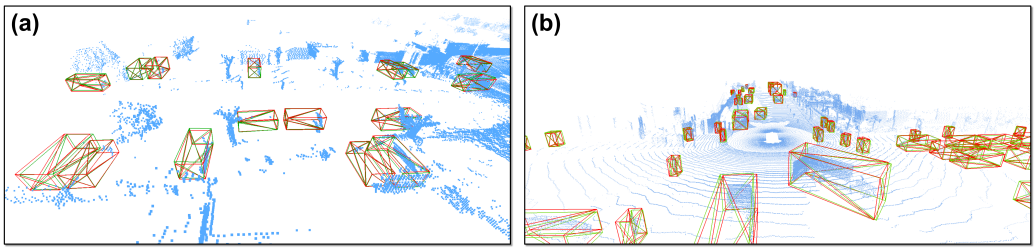

Figure 4: Qualitative results on Waymo. The red and green boxes denote ground-truths and predictions, respectively. Our MsSVT performs impressively in scenes (a) beyond the range of $50\mathrm{m}$, and (b) in the presence of dense objects with large scale variations.

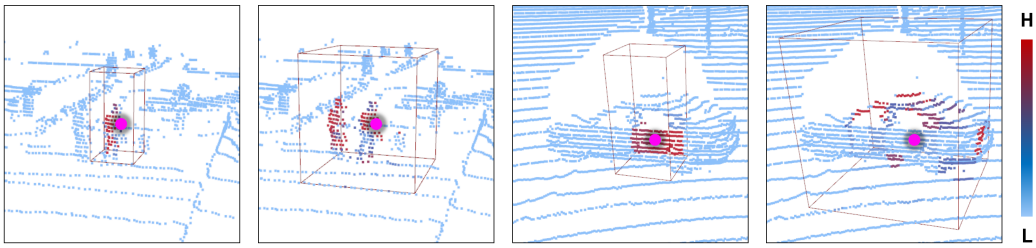

Figure 5: Visualization of attention maps. Pink dot denotes the query position. Positions with high and low attention weights are in red and blue, respectively.

Fig. 5. One can observe that the query point pays more attention to fine-grained details in a small window and is more interested in long-range contextual information in a large window.

Scale-aware relative position encoding. Table 4 shows compared with the model variant that adopts scale-agnostic position encoding (listed in the second row), our scale-aware relative position encoding gives better performance, which well supports our design motivation that the importance of positional information varies with different scales.

Chessboard sampling. Table 5 reports the performance of the model variants using chessboard sampling with different sampling ratios, and of the model without performing any sampling. One can see that our model is not sensitive to the variation of sampling ratio. The model variant with sampling ratio 1/4 performs even on par with the model without sampling, while bringing a significant reduction (up to 6.1G) in memory footprint.

Table 5: Ablations on sampling strategy.

Strategy	Veh / Ped / Cyc	Mem (G)	Lat (ms)
w/o	72.58/75.74/68.24	18.3	167
1/2	72.44/76.03/67.81	14.5	138
1/4	72.37/75.99/67.90	12.2	121
1/8	72.01/75.54/67.43	11.4	113

5 Conclusion

In this paper, we propose MsSVT, a novel 3D sparse voxel transformer backbone for 3D detection to capture mixed-scale information simultaneously. We first utilize the novel chessboard sampling and balanced multi-window sampling to obtain queries and a set of keys with different scales. Then MsSVT explicitly divides attention heads into multiple scale-aware groups and each in charge of capturing information at a specific scale. Equipped with the scale-aware relative position encoding, MsSVT finally gets the mixed-scale features which is crucial for accurate detection.

Limitations. MsSVT considers both fine-grained details and long-range contextual information in different windows and achieves promising performance, but still needs to set hand-crafted window sizes, we will explore adaptive window version of MsSVT in the future.

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

[2] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, pages 1907–1915, 2017.

[3] 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*, 1(2):4, 2020.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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

[6] Lue Fan, Ziqi Pang, Tianyuan Zhang, Yu-Xiong Wang, Hang Zhao, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Embracing single stride 3d object detector with sparse transformer. *arXiv preprint arXiv:2112.06375*, 2021.

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

[8] 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, 2021.

[9] Chenhang He, Ruihuang Li, Shuai Li, and Lei Zhang. Voxel set transformer: A set-to-set approach to 3d object detection from point clouds. *arXiv preprint arXiv:2203.10314*, 2022.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[11] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *IROS*, pages 1–8. IEEE, 2018.

[14] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019.

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

[16] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Improved multiscale vision transformers for classification and detection. *arXiv preprint arXiv:2112.01526*, 2021.

[17] Zhichao Li, Feng Wang, and Naiyan Wang. Lidar r-cnn: An efficient and universal 3d object detector. In *CVPR*, 2021.

[18] 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 *ICCV*, 2021.

[19] Anas Mahmoud, Jordan SK Hu, and Steven L Waslander. Dense voxel fusion for 3d object detection. *arXiv preprint arXiv:2203.00871*, 2022.

[20] Jiageng Mao, Yujing Xue, Minzhe Niu, Haoyue Bai, Jiashi Feng, Xiaodan Liang, Hang Xu, and Chunjing Xu. Voxel transformer for 3d object detection. In *ICCV*, 2021.

[21] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015.

[22] Chunghyun Park, Yoonwoo Jeong, Minsu Cho, and Jaesik Park. Fast point transformer. *arXiv preprint arXiv:2112.04702*, 2021.

[23] Namuk Park and Songkuk Kim. How do vision transformers work? *arXiv preprint arXiv:2202.06709*, 2022.

[24] Sucheng Ren, Daquan Zhou, Shengfeng He, Jiashi Feng, and Xinchao Wang. Shunted self-attention via multi-scale token aggregation. *arXiv preprint arXiv:2111.15193*, 2021.

[25] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

[26] 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 *ICCV*, 2021.

[27] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019.

[28] Shaoshuai Shi, Zhe Wang, Xiaogang Wang, and Hongsheng Li. Part-a^ 2 net: 3d part-aware and aggregation neural network for object detection from point cloud. *arXiv preprint arXiv:1907.03670*, 2(3), 2019.

[29] 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 *CVPR*, 2020.

[30] Shaoshuai Shi, Li Jiang, Jiajun Deng, Zhe Wang, Chaoxu Guo, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection. *arXiv preprint arXiv:2102.00463*, 2021.

[31] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *ICCV*, pages 945–953, 2015.

[32] 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 *CVPR*, 2020.

[33] Pei Sun, Weiyue Wang, Yuning Chai, Gamaleldin Elsayed, Alex Bewley, Xiao Zhang, Cristian Sminchisescu, and Dragomir Anguelov. Rsn: Range sparse net for efficient, accurate lidar 3d object detection. In *CVPR*, 2021.

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

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017.

[36] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*, pages 568–578, 2021.

[37] Yue Wang, Vitor Campagnolo Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, and Justin Solomon. Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In *Conference on Robot Learning*, pages 180–191. PMLR, 2022.

[38] Kan Wu, Houwen Peng, Minghao Chen, Jianlong Fu, and Hongyang Chao. Rethinking and improving relative position encoding for vision transformer. In *ICCV*, 2021.

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

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

[41] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *CVPR*, pages 7652–7660, 2018.

[42] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan, and Jianfeng Gao. Focal self-attention for local-global interactions in vision transformers. *arXiv preprint arXiv:2107.00641*, 2021.

[43] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *CVPR*, 2020.

[44] Zetong Yang, Li Jiang, Yanan Sun, Bernt Schiele, and Jiaya Jia. A unified query-based paradigm for point cloud understanding. *arXiv preprint arXiv:2203.01252*, 2022.

[45] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *CVPR*, 2021.

[46] Cheng Zhang, Haocheng Wan, Shengqiang Liu, Xinyi Shen, and Zizhao Wu. Pvt: Point-voxel transformer for 3d deep learning. *arXiv preprint arXiv:2108.06076*, 2021.

[47] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *ICCV*, 2021.

[48] Daquan Zhou, Zhiding Yu, Enze Xie, Chaowei Xiao, Anima Anandkumar, Jiashi Feng, and Jose M Alvarez. Understanding the robustness in vision transformers. *arXiv preprint arXiv:2204.12451*, 2022.

[49] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018.

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

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [No] Our work is only for academic research purpose.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Our implementation details are posted on Section 4.1 and the supplemental materials. The data are public datasets and our code will be available.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Section 4.1 and setup in 4.2 and 4.3

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We report average performance of 5 trials by default.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Section 4.1.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] Waymo [32] and KITTI [7]

(b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [No] We don't include any new assets.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] Waymo [32] and KITTI [7] are two standard benchmarks.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] Waymo [32] and KITTI [7] are two standard benchmarks.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] We didn't use crowdsourcing or conduct research with human subjects...

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] We didn't use crowdsourcing or conduct research with human subjects...

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] We didn't use crowdsourcing or conduct research with human subjects...