Embracing Single Stride 3D Object Detector with Sparse Transformer

Lue Fan
CASIA
fanlue2019@ia.ac.cn

Ziqi Pang
UIUC
ziqip2@illinois.edu

Tianyuan Zhang
CMU
tianyuaz@andrew.cmu.edu

Yu-Xiong Wang
UIUC
yxw@illinois.edu

Hang Zhao
THU
hangzhao@mail.tsinghua.edu.cn

Feng Wang
TuSimple
feng.wff@gmail.com

Naiyan Wang
TuSimple
winsty@gmail.com

Zhaoxiang Zhang
CASIA
zhaoxiang.zhang@ia.ac.cn

Abstract

*In LiDAR-based 3D object detection for autonomous driving, the ratio of the object size to input scene size is significantly smaller compared to 2D detection cases. Overlooking this difference, many 3D detectors directly follow the common practice of 2D detectors, which downsample the feature maps even after quantizing the point clouds. In this paper, we start by rethinking how such multi-stride stereotype affects the LiDAR-based 3D object detectors. Our experiments point out that the downsampling operations bring few advantages, and lead to inevitable information loss. To remedy this issue, we propose Single-stride Sparse Transformer (SST) to maintain the original resolution from the beginning to the end of the network. Armed with transformers, our method addresses the problem of insufficient receptive field in single-stride architectures. It also cooperates well with the sparsity of point clouds and naturally avoids expensive computation. Eventually, our SST achieves state-of-the-art results on the large-scale Waymo Open Dataset. It is worth mentioning that our method can achieve exciting performance (**83.8** LEVEL_1 AP on validation split) on small object (pedestrian) detection due to the characteristic of single stride. Codes will be released at https://github.com/TuSimple/SST.*

1. Introduction

LiDAR-based 3D object detection for autonomous driving has been benefiting from the progress of image-based object detection. The mainstream 3D detectors quantize the 3D space into a stack of pseudo-images from Bird

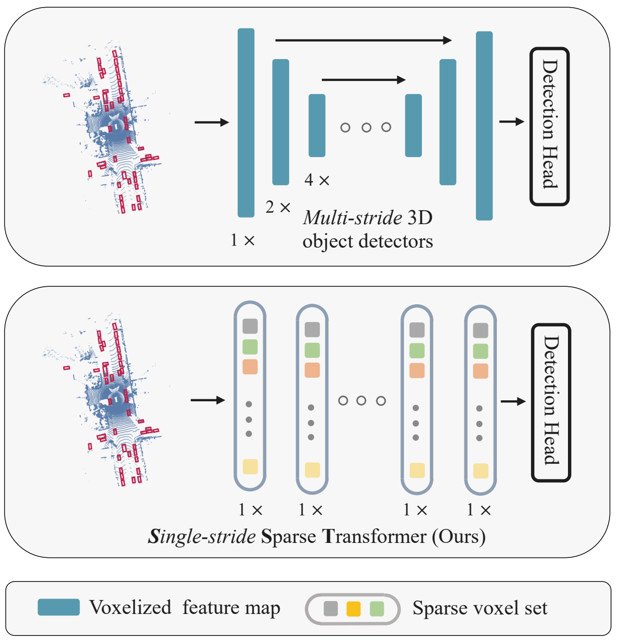


Figure 1. Compared with previous multi-stride 3D detectors, our model is single-stride and operates sparsely on the non-empty voxels. We paint the vehicle bounding boxes on the input point cloud to show the tiny object size compared to the input scene size.

Eye's View (BEV), which makes it convenient to borrow advanced techniques from the 2D counterparts. Many works [14, 20, 62, 65] are proposed under this paradigm and achieve competitive performance. However, 3D and 2D spaces have intrinsic distinction in their relative object scales, where the objects in 3D spaces have much smaller relative sizes (See Fig. 2). For example, in Waymo Open

Dataset [50], the perception range is usually $150m \times 150m$, while a vehicle is only about $4m$ long, even a pedestrian occupies as little as $1m$ in length. Such a tiny pedestrian equivalently translates to an object of size 8×8 pixels in a 1200×1200 image, suggesting that object detection on such a tiny scale is one of the challenges in 3D object detection.

Different from the above challenge of small scales in the 3D space, 2D detectors have to consider the handling of the objects with varied scales. It is observed in Fig. 2 that the scales of objects in 2D images exhibit a long-tail distribution, while in 3D space they are quite concentrated due to the non-projective transformation used in voxelization. To handle the varied scales, 2D detectors [24, 26, 48, 49] usually build multi-scale features with a series of downsampling and upsampling operations. Such multi-scale architecture is also widely inherited in 3D detectors (See Fig. 1) [14, 20, 62, 65, 70]. Since the object size in 3D object detectors is usually tiny while no large objects exist, a question naturally arises: *do we really need downsampling in 3D object detectors ?*

With this question in mind, we make an exploratory attempt on the single-stride architecture with **no downsampling operators**. The single-stride network maintains the original resolution throughout the network. However, it is challenging to make such a design feasible. The discard of downsampling operators leads to two issues: 1) the increase of computation cost; 2) the decrease of receptive field. The former constrains the applicability to the real-time system and the latter hinders the capability of object recognition. For the issue of computation, sparse convolution seems to be a solution, but the sparse connectivity between voxels[1] makes the decrease of receptive field even more severe (See Table 7). For the issue of receptive field, we experimentally show that some commonly adopted techniques do not meet our needs (See Table 1): the dilated convolution [5, 66] is not friendly to small objects, and the larger kernel leads to unaffordable computational overhead in the single stride architecture. Therefore, we are getting into a dilemma, where it is difficult to design a convolutional network simultaneously satisfying the three aspects: *single stride architecture*, *sufficient receptive field*, and *acceptable computation cost*.

These difficulties naturally lead us to think out of the paradigm of CNN, and the attention mechanism emerges as a better option because of the following two reasons: 1) The attention-based model is better at capturing large context and build sufficient receptive field. 2) Due to the capability of modeling dynamic data, the attention-based model fits well into the sparse voxelized representation of point clouds, where only a small portion of voxels are occupied. This property guarantees the efficiency of our single stride network. Although the attention mechanism is efficient on sparse data, computing attentions on a global scale is still unaffordable and undesirable. So we partition the voxelized 3D space into many local regions and apply self-attention inside each of them. Eventually, this local attention mechanism, named as **Sparse Regional Attention (SRA)**, enjoys the best of two worlds. By stacking SRA layers, we make the single-stride network feasible and obtain a transformer-style network, called **Single-stride Sparse Transformer (SST)**. Extensive experiments are conducted on the large-scale Waymo Open Dataset [50]. We summarize our contributions as follows:

[1]We provide a clear illustration for this in our supplementary materials.

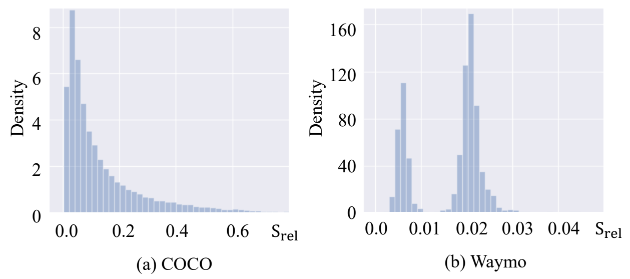


Figure 2. **Distribution of the relative object size S_{rel} in COCO dataset [28] and Waymo Open Dataset (WOD).** S_{rel} is defined as $\sqrt{A_o/A_s}$, where A_o denotes the area of 2D objects (COCO) and the BEV area of 3D objects (WOD). A_s is the image area in COCO, and $150m \times 150m$ in WOD. In COCO 73.03% objects in COCO have a S_{rel} larger than 0.04, while only 0.54% objects in WOD have a S_{rel} larger than 0.04.

- We rethink the architecture of current mainstream LiDAR-based 3D detectors. With pilot experiments, we point out that the network stride is an overlooked design factor for LiDAR-based 3D detectors.
- We propose the **S**ingle-stride **S**parse **T**ransformer (**SST**). With its local attention mechanism and capability of handling sparse data, we overcome receptive field shrinkage in the single-stride setting and avoid heavy computational overhead.
- Our method achieves state-of-the-art performance on the large-scale Waymo Open Dataset. Thanks to the characteristic of single stride, our method obtains exciting results on tiny objects like pedestrians (83.8 LEVEL_1 AP on the validation split).

2. Related Work

3D LiDAR-based Detection There are three major representations for point cloud learning in autonomous driving, Point-based, Voxel-based, and Range View. Point based representation backed by PointNet families [38, 39] are widely adopted for feature learning of small region of irregular points [7, 25, 37, 45]. Voxel-based representation [20, 62, 65, 70] combined with convolutions are the most popular treatment. As explored in several recent works

[1, 3, 14, 22, 34], range view enjoys computational advantages over voxels, especially for long-range LiDAR sensors. Some hybrid approaches investigate how to combine different types of representations [6, 25, 43, 44, 51, 58].

Transformers in Visual Recognition The success of transformer architectures in NLP [11, 54] and speech recognition [8] has inspired lots of work to investigate the power of attention in visual recognition [12, 30, 41, 53, 67]. The pioneering work ViT [12] splits an image into patches, and then feeds sequences of patches to multiple transformer blocks for image classification. DeiT [53] explores training strategies for data-efficient learning of vision transformers. Swin-Transformer [30] exploits the power of local attention to build high-performance transformer-based image backbones. Several works have investigated the use of transformers for point cloud perceptions. Some of them focus on the indoor scene such as [18, 35, 68]. For autonomous driving scenarios, Pointformer [36] proposes a point-based local and global attention module directly operating on point clouds. In addition, VoTr [32] uses the local self-attention module to replace the sparse convolution [17] for voxel processing, where each voxel serves as a query and attends with its neighbor voxels.

Small Object Detection Small object detection [28, 61, 71] is a challenging track in 2D object detection. The mainstream of current methods [15, 26, 47, 55, 63] focuses on increasing the resolution of the input and output features, while none of them gives up the multi-stride architectures. Some other methods adopt the scale-aware training [24, 26, 49] and strong data augmentations [19, 72]. To the best of our knowledge, there is no method specialized for small object detection in 3D space.

3. Discussion of Network Stride

The stride of a network is a simple but critical aspect in the architecture design. Some previous works [14, 16, 65] in 3D detection have found that the performance can benefit from the recovery of output resolution by upsampling. However, they do not delve into this phenomenon. Therefore, we conduct a simple pilot study to reveal the influence of network stride on 3D detectors and motivate the design of our network.

For generality, we adopt the widely used PointPillars [20] in MMDetection3D [9] as our base model. The experiments are conducted on Waymo Open Dataset [50]. We uniformly sample 20% training data (32K frames) [2] and adopt 1× schedule (12 epochs).

Based on the standard PointPillars model D_2, we extend it to three more variants: D_3, D_1, and D_0, and they only differ in the network stride. From D_3 to D_0, the set of strides of their four stages for each model are $\{1, 2, 4, 8\}$, $\{1, 2, 4, 4\}$, $\{1, 2, 2, 2\}$ and $\{1, 1, 1, 1\}$, respectively. Since the output feature maps of the four stages will be upsampled to the original resolution by an FPN-like module, our modification does not change the resolution of feature maps in the detection head. Except for the resolution of feature maps, all the four models have the same hyper-parameters. To reduce memory overhead, we change the filter number from 256 to 128 in convolution layers.

[2]Training with 20% data is a setting for efficient validation adopted in [9, 52].

The main results are shown in Table 1. Performances of all three classes improve from D_3 to D_1, and there is a significant boost from D_2 to D_1. The performance boost from D_3 to D_1 supports our motivation that *Smaller strides are better for 3D detection.*

However, from D_1 to D_0, the vehicle performance has a significant drop, while the performance drop in pedestrian is slight and performance of cyclist keeps going up. We conjecture that the limited receptive field of D_0 hinders the performance improvement from D_1 to D_0 since the pedestrian and cyclist have smaller sizes than vehicles.

To verify our conjecture, we add two more variants: $D_0^{dilation}$ and $D_0^{5\times5}$. $D_0^{dilation}$ adopts dilated convolutions with dilation as 2 in the last two stages. $D_0^{5\times5}$ increases the kernel size in last two stages to 5×5. Table 1 shows that, dilation increases the performance of vehicle class while decreases performances of pedestrian and cyclist, indicating that it indeed enlarges the receptive field, however misses fine-grained details. Meanwhile, larger kernel consistently improves the performance of all three classes but unfortunately has the highest latency. Above studies support our major motivation of single-stride 3D detectors, and it also reveals the another important aspect in our network design: *Sufficient receptive field is crucial.*

Models	Vehicle	Pedestrian	Cyclist	Latency
D_3	63.66	60.82	47.08	58ms
D_2	64.01 $\uparrow$ 0.35	60.85 $\uparrow$ 0.03	47.52 $\uparrow$ 0.44	60ms
D_1	66.03 $\uparrow$ 2.02	65.06 $\uparrow$ 4.21	52.97 $\uparrow$ 5.45	91ms
D_0	64.69 $\downarrow$ 1.34	64.32 $\downarrow$ 0.74	53.02 $\uparrow$ 0.05	185ms
$D_0^{dilation}$	66.26 $\uparrow$ 1.57	63.51 $\downarrow$ 0.81	50.95 $\downarrow$ 2.07	192ms
$D_0^{5\times5}$	66.42 $\uparrow$ 1.77	65.71 $\uparrow$ 1.41	53.70 $\uparrow$ 0.68	340ms

Table 1. Results of pilot study on Waymo Open Dataset validation split. Latency is evaluated in 2080Ti GPU with 2000 samples after a cold start of 500 samples. For D_n, the arrows indicate the performance changes based on D_{n+1}. For $D_0^{dilation}$ and $D_0^{5\times5}$, the arrows indicate performance changes based on D_0. Best viewed in color.

In summary, above experiments verify two motivations of 3D object detector designs:

- The single stride architecture has a great potential in LiDAR-based 3D detection for autonomous driving.
- The key to make single stride architecture feasible lies in appropriately addressing the shrinkage of receptive field and reducing computational overhead.

4. Methodology

4.1. Overall Architecture

So far, we know the keys to make single stride architecture feasible are sufficient receptive field and acceptable computational cost. However, as we discussed in Sec. 1, it is difficult to simultaneously satisfy the two factors with convolutional single stride architecture. So we turn to the attention mechanism in Transformer [54], and present our method as follows.

We build up our *Single-stride Sparse Transformer* (SST) as in Fig. 4. SST voxelizes the point clouds and extracts voxel features following prior work [20, 62, 70]. For each voxel and its features, SST treats them as "tokens." SST first partitions the voxelized 3D space to fixed-size non-overlapping regions (Sec. 4.2). Then SST applies *Sparse Regional Attention* (SRA) to voxel tokens in each region (Sec. 4.3). To handle the objects scattering multiple regions and capture useful local context, we adopt *Region Shift* (Sec. 4.4), which is inspired by the shifted window in Swin-Transformer [30]. The backbone preserves the number of voxels as well as their spatial locations, thus satisfying the single-stride property, and can be integrated with mainstream detection heads (Sec. 4.5).

4.2. Regional Grouping

Given the input voxel tokens, *Regional Grouping* divides the 3D space into non-overlapping regions, so that the self-attentions only interact with tokens coming from the same regions. The regional grouping not only maintains sufficient receptive field, but also avoids expensive computation overhead in global attentions. We illustrate it intuitively in Fig. 3. Each *regional grouping* divides the input tokens into groups according to their physical locations, where the tokens belonging to the same regions (green rectangles) are assigned to the same group.

4.3. Sparse Regional Attention

Sparse Regional Attention (SRA) operates on the regional sparse sets of voxel tokens coming from *regional grouping*. For a group of tokens $\mathcal{F}$ and their corresponding spatial (x, y, z) coordinates $\mathcal{I}$, SRA follows conventional transformers as follows

$$\begin{aligned} \mathcal{F}^{'} &= \mathbf{MSA}(\mathbf{LN}(\mathcal{F}), \mathbf{PE}(\mathcal{I})) + \mathcal{F} \\ \widetilde{\mathcal{F}} &= \mathbf{MLP}(\mathbf{LN}(\mathcal{F}^{'})) + \mathcal{F}^{'} \end{aligned} \tag{1}$$

where $\mathbf{PE}(\cdot)$ stands for the absolute positional encoding function used in [2], $\mathbf{MSA}(\cdot)$ denotes the Multi-head Self-Attention, and $\mathbf{LN}(\cdot)$ represents Layer Normalization. This manner of SRA well exploits the sparsity of point clouds, because it only computes the voxels with actual LiDAR points.

Region Batching for Efficient Implementation Due to the sparsity of point cloud, the number of valid tokens in each region varies. To utilize the parallel computation of modern devices, we batch regions with similar number of tokens together. In practice, if a region contains the tokens with number N_{token}, satisfying:

$$2^i \leqslant N_{token} < 2^{i+1}, \quad i \in \{0, 1, 2, 3, 4, 5, 6\}, \tag{2}$$

then we pad the number of tokens to 2^{i+1}. With padded tokens, we can divide all the regions into several batches, and then process all regions in the same batch in parallel. As the padded tokens are masked in the computation as in [2, 54], they have no effect on other valid tokens. In this way, it is easy to implement an efficient SRA module in current popular deep learning frameworks without engineering efforts as taken in the sparse convolution [17, 62].

4.4. Region Shift

Though SRA can cover a considerably large region, there are some objects inevitably truncated by the grouping. To tackle this issue and aggregate useful context, we further use *Region Shift* in our design, which is similar to the shifting mechanism in Swin Transformer for information communication. Supposing the size of regions in *regional grouping* is (l_x, l_y, l_z), the *Region Shift* moves the original regions by $(l_x/2, l_y/2, l_z/2)$ and groups the tokens according to this new set of regions, as illustrated in "Shifted regional grouping" of Fig. 3.

4.5. Integration with Detection

To work with the existing detector heads, SST places the sparse voxel tokens back to dense feature maps according to their spatial locations. Unoccupied locations are filled with zeros. As LiDAR only captures points on object surfaces, 3D object centers are likely to reside on the empty locations with zero features, which is unfriendly to the current designs of detection heads. [20, 65]. So we add two 3×3 convolutions to fill most of the holes on the object centers.

As for the detection head and loss function, we adopt the same settings as PointPillars [20] for simplicity. Specifically, we use the SSD [29] head, the smooth L1 bounding box localization loss $\mathcal{L}_{loc}$, the classification loss $\mathcal{L}_{cls}$ in the form of focal loss [27], and the direction loss $\mathcal{L}_{dir}$ penalizing wrong orientations. The final loss function is Eq 3, where N_p is the number of positive samples. We leave the detailed setting in supplementary materials.

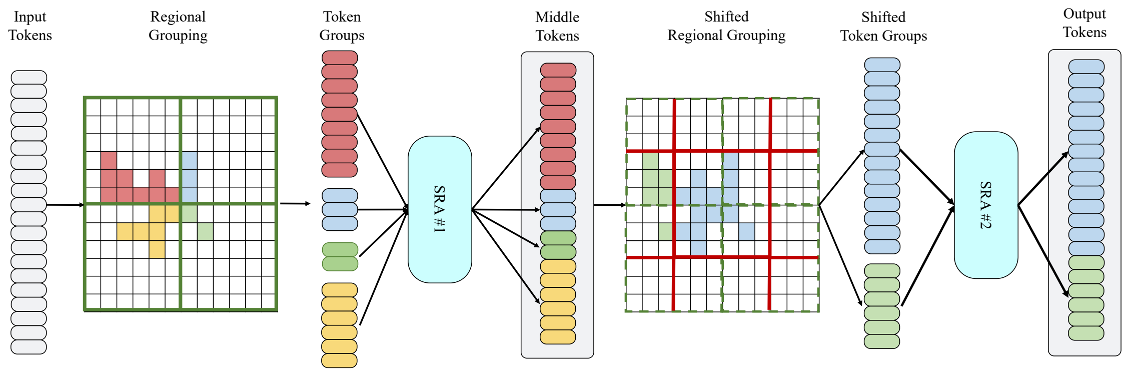


Figure 3. Computation of an example block in SST. For an incoming set of tokens, *Regional Grouping* first groups them according to the partitions of regions (in Sec. 4.2). Second, *Sparse Regional Attention* (SRA) deals with each group of tokens separately (Sec. 4.3). Third, the tokens are grouped another time according to *Region Shift*, and a second SRA processes the new groups of tokens (Sec. 4.4). These three steps complete the computation of a block.

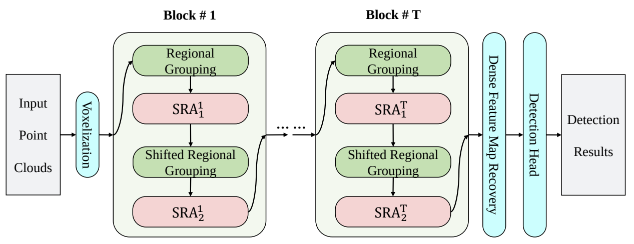


Figure 4. Architecture overview for *Single-stride Sparse Transformer* (SST). It begins from voxelizing an input point cloud, then processes the voxels with T blocks, and eventually recover a dense feature map. Inside each block, we consecutively append regional grouping on the voxel tokens and *Sparse Regional Attention* (SRA) to process them. Details in Sec. 4.1.

$$\mathcal{L} = \frac{1}{N_p}(\beta_{loc}\mathcal{L}_{loc} + \beta_{cls}\mathcal{L}_{cls} + \beta_{dir}\mathcal{L}_{dir}) \qquad (3)$$

4.6. Two Stage SST

Although our main contribution lies in the design of the single stride architecture in the first stage, there is a considerable gap between the single stage detector and the two stage detector. To match the performance with current two stage detectors, we apply LiDAR-RCNN [25] as our second stage. LiDAR-RCNN is a lightweight second stage network consists of a simple PointNet [38] for feature extraction, only taking the raw point cloud inside proposal as input.

4.7. Discussion

Because of the distinctions between point clouds and RGB images, there are several differences in the design choices and motivations between our design and Swin-Transformer [30] as highlighted here.

- Our SST network follows the **single-stride** guideline, while Swin-Transformer follows the hierarchical structure with **multi-stride**, which uses "token merge" to increase the receptive field.
- The tokens for our region-based attention scatter **sparsely** because of the sparsity of point clouds, while the tokens in vision transformers have **dense** layouts. This is one of the reasons for the efficiency of SST even in the single stride architecture.

5. Experiments

5.1. Dataset

We conduct our experiments on Waymo Open Dataset (WOD) [50]. The dataset contains 1150 sequences in total (more than 200K frames), 798 for training, 202 for validation and 150 for test. Each frame covers a scene with a size of $150m \times 150m$. It is a very challenging dataset and adopted as the benchmark in many recent state-of-the-art methods.

5.2. Implementation Details

We implement our model based on the popular 3D object detection codebase – MMDetection3D [9], which provides standard and solid baselines. Please refer to supplementary materials for more details.

Model Setup For generality, we build our *Single-stride Sparse Transformer* (SST) on the basis of popular PointPillars [20]. We replace its backbone with 6 consecutive *Sparse Regional Attentions* (SRA) blocks, and each block contains 2 attention modules as Fig. 4 shows. All the attention modules are equipped with 8 heads, 128 input channels, and 256 hidden channels. In *Regional Grouping*, each region covers a volume with size $3.84m \times 3.84m \times 6m$. As for other parts, SST follows the implementation of PointPillars in MMDetection3D. We use the BEV pillar size of

$0.32m \times 0.32m \times 6m$, which can be easily extended to the 3D voxels with smaller heights.

Model Variants We develop several variants of SST in our experiments. SST_1f: basic single-stage model using 1-frame point cloud. SST_3f: consecutive 3 frame point clouds are used as model input, and the point cloud in different frames are concatenated together after aligning the ego-pose. SST_TS_1f and SST_TS_3f: two stage model based on above models, using a standard LiDAR-RCNN [25] for refinement.

Training Scheme We train our model for 24 epochs (2×) on WOD with AdamW optimizer and cosine learning rate scheduler. The maximum learning rate is 0.001, and the weight decay is 0.05.

5.3. Comparison with State-of-the-art Detectors

We compare our SST with state-of-the-art methods in Table 2 (vehicle) and Table 3 (pedestrian). We divide current methods into the branches of one-stage and two-stage detectors for fair comparison.

Table 2 shows the results on vehicles, where our models achieve competitive performances. With a lightweight second stage for refinement, our two-stage detectors are comparable with state-of-the-art methods.

Table 3 shows the results on pedestrians. Due to the tiny size and non-rigid property, pedestrian detection is more challenging than vehicle detection. Networks are prone to confuse pedestrians with other slim objects, like poles and trees, leading to a high false positive rate. Under such cases, our best model **outperforms all other methods in the challenging pedestrian class**. SST_TS_3f is **4.4** AP ahead of the second best RSN with the same temporal information (3 frames). We owe such leading performance to the single-stride characteristic of SST.

5.4. Deep Investigation of Single Stride

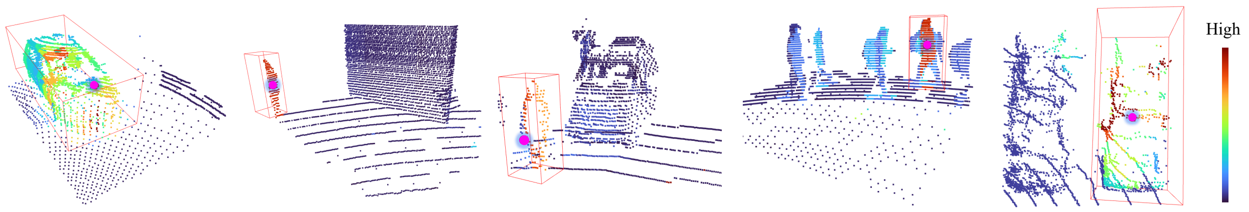
Single-stride models better use dense observations. **First**, SST has more advantages in short-range metrics (0m - 30m) than in long-range metrics (50m - inf): In Table 4, SST_1f outperforms the PointPillars counterpart in short-range metric by **12.8** AP for pedestrian class, but the margin is not that significant over PointPillars in the long-range metric. **Second**, SST benefits more from multi-frame data. In Table 4, RSN [51] got improved by **6.4** AP in long-range metric from RSN_1f to RSN_3f, while the performance of SST in long-range metric gets more significantly improved by **10.4** AP from SST_1f to SST_3f.

Does the single stride model fail on large vehicles? As smaller strides reduce the receptive fields, it would be a major concern whether our model has sufficient receptive fields for extreme cases, *e.g.*, extremely large vehicles. We therefore divide all the vehicles into three groups according to the lengths of their ground-truth boxes, and evaluate the recalls of SST on them. Please refer to supplementary materials for the evaluation details. In Table 5, our SST outperforms the PointPillars baseline for all vehicles, even those longer than $8m$. This supports that our attention mechanism provides proper receptive fields in the single stride architecture.

Localization quality test with stricter IoU thresholds. By preserving the original resolution, our SST is supposed to localize objects more precisely as in [21]. To verify this, we evaluate SST with higher 3D IoU thresholds (0.8 for vehicle, 0.6 for pedestrian). In Table 6, we compare our models with the PointPillars baseline and other models with available results from [40], then a couple of interesting findings emerge:

Methods	LEVEL_1 3D AP/APH	LEVEL_2 3D AP/APH
One-Stage Methods		
SECOND ‡ [62]	72.27/71.69	63.85/63.33
MVF [69]	62.93/-	-/-
LaserNet ¶ [34]	56.10/-	-/48.40
AFDet [16]	63.69/-	-/-
Pillar-OD [58]	69.80/-	-/-
PPC [3]	65.2/-	-/56.7
VoTr-SSD [33]	68.99/68.39	60.22/59.69
RangeDet [14]	72.85/72.33	64.03/63.57
CenterPoint-Voxel [65]	**74.78/74.22**	**66.70/66.19**
PointPillars* [20]	72.08/71.53	63.55/63.06
SST_1f (Ours)	74.22/73.77	65.47/65.07
SST_3f (Ours)	**77.04/76.56**	**68.50/68.08**
Two-Stage Methods		
Voxel RCNN [10]	75.59/-	66.59/-
RCD [1]	69.0/68.5	-/-
VoTr-TSD [33]	74.95/74.25	65.91/65.29
LiDAR-RCNN [25]	76.0/75.5	68.3/67.9
Pyramid RCNN [31]	76.30/75.68	67.23/66.68
Voxel-to-Point [23]	77.24/-	69.77/-
3D-MAN [64]	74.53/74.03	67.61/67.14
Part-A2-Net ‡ [46]	77.05/76.51	68.47/67.97
CenterPoint-Pillar [65]	76.10/75.50	68.00/67.50
CenterPoint-Voxel [65]	76.59/7605	68.85/68.35
PV-RCNN [43]	77.51/76.89	68.98/68.41
PV-RCNN++ [44]	**78.79/78.21**	**70.26/69.71**
RSN_1f † [51]	75.10/74.60	66.00/65.50
RSN_3f † [51]	78.40/78.10	69.50/69.10
SST_TS_1f (Ours)	76.22/75.79	68.04/67.64
SST_TS_3f (Ours)	**78.66/78.21**	**69.98/69.57**

Table 2. Performances of **vehicle** detection on the Waymo Open Dataset validation split. We mark the best result in **red**, and the second result in **blue**. †: RSN [51] is not a typical two stage detector, we put it here because it uses a segmentation network to remove background first. *: re-implemented by MMDetection3D. ¶: from [3]. ‡: from [44].



(a) Complete Vehicle (b) Person near a Wall (c) Person beside a Vehicle (d) Multiple Pedestrians (e) Multiple Vehicles

Figure 5. Visualization of the learned sparse regional attention. Each figure shows the attention weight distribution between the query token (pink dot) and all other tokens in the local region. Best viewed in color.

Methods	LEVEL_1 3D AP/APH	LEVEL_2 3D AP/APH
One-Stage Methods		
LaserNet¶ [34]	62.9/-	-/45.4
SECOND ‡ [62]	68.70/58.18	60.72/51.31
MVF [69]	65.33/-	-/-
Pillar-OD [58]	72.51/-	-/-
PPC [3]	73.90/-	-/59.60
RangeDet [14]	75.94/**71.94**	67.60/**63.89**
CenterPoint-Voxel [65]	75.82/69.65	68.34/62.62
PointPillars* [20]	70.59/56.70	62.84/50.25
SST_1f (Ours)	**78.71**/69.55	**70.02**/61.67
SST_3f (Ours)	**82.42**/**77.96**	**75.14**/**70.88**
Two-Stage Methods		
LiDAR-RCNN [25]	71.2/58.7	63.1/51.7
3D-MAN [64]	71.71/67.74	62.58/59.04
Part-A2-Net ‡ [46]	75.24/66.87	66.18/58.62
PV-RCNN [43]	75.01/65.65	66.04/57.61
PV-RCNN++ [44]	76.67/67.15	68.51/59.72
CenterPoint-Pillar [65]	76.10/65.10	68.10/57.90
CenterPoint-Voxel [65]	79.02/73.44	70.98/65.75
RSN_1f † [51]	77.80/72.70	68.30/63.70
RSN_3f † [51]	79.40/**76.20**	69.90/**67.00**
SST_TS_1f (Ours)	**81.39**/74.05	**72.82**/65.93
SST_TS_3f (Ours)	**83.81**/**80.14**	**75.94**/**72.37**

Table 3. Performance of **pedestrian** detection on the Waymo Open Dataset official validation split. Please refer to Table 2 for the meanings of the notions in this table.

1. Comparing MVF++ [40] with our SST_1f on vehicles, MVF++ is slightly better than SST_1f under the normal threshold, while SST_1f is better with the stricter threshold. This suggests the single stride structure enables more precise localization of vehicles.
2. The 3DAL [40] is an offboard method using all the past and future frames in a sequence (around 200 frames) and is equipped with tracking [59]. Nonetheless, our best model *SST_TS_3f surprisingly surpasses 3DAL on pedestrian* on both IoU thresholds with as few as 3 frames of point clouds.

These findings suggest that the single-stride architecture is capable of better localizing objects with full and fine-grained information.

Method	LEVEL_1 Pedestrian AP Overall	0-30m	30-50m	50m-inf
RSN_1f	77.8	83.9	74.1	62.1
RSN_3f	79.0	84.5	78.1	68.5
PointPillars	70.6	72.5	71.9	63.8
PointPillars_3f	73.7	72.9	75.9	70.6
SST_1f	78.7↑ 8.1	85.3↑ 12.8	77.0↑ 5.1	63.4↓ 0.4
SST_3f	82.4↑ 8.7	86.1↑ 13.2	81.2↑ 5.3	73.8↑ 3.2

Table 4. Distance-conditioned **pedestrian** detection performance. Our SST has larger advantages in short-range metrics and the multi-frame setting, where points are more dense. The increases and decreases are calculated based on PointPillars baseline.

Methods	Vehicle Recalls (IoU=0.7) $[0m, 4m]$	$[4m, 8m]$	$[8m, +\infty]$
PointPillars	40.60	73.11	10.59
SST_1f	41.31 ↑ 0.71	80.85 ↑ 7.74	13.41 ↑ 2.82

Table 5. Recalls for vehicles with different lengths. The vehicles with lengths in $[0m, 4m]$ and $[8m, +\infty]$ are rare (7.3% and 1.6% in WOD) and hardly get sufficient training, so their performances are relatively low.

Comparison with other alternatives. There are some potential alternatives to our SST in order to preserve the input resolution. Here we make a comprehensive comparison. We first introduce these alternative models as follows. **PointPillars-SS**: The single stride version of PointPillars introduced in Sec. 3. **SparsePillars-SS**: We replace all the standard 2D convolutions in backbone of PointPillars-SS with Submanifold Sparse Convolutions [17, 62]. Due to the sparsity, SparsePillars-SS also faces the issue of "empty hole" (details in Sec. 4.5) as in SST, so we add two more 2D convolutions before its detection head. **HRNetV2p-W18** [55]: HRNet maintains the high resolution while building multi-scale features. We adopt the standard HRNetV2p-W18 from MMDetection [4] for the experiment. To keep the output resolution in HRNet the same as

Method	Frames	Vehicle		Pedestrian	
		Normal	Strict	Normal	Strict
Single frame					
PointPillars [20]	1	72.08	36.83	70.59	44.86
PV-RCNN* [43]	1	70.47	39.16	65.34	45.12
MVF++* [40]	1	**74.64**	43.30	78.01	56.02
SST_1f (Ours)	1	74.22	**44.08**	**78.71**	**56.12**
Multiple frames					
MVF++ w. TTA* [40]	5	79.73	49.43	81.83	60.56
3DAL* [40]	all†	**84.50**	**57.82**	82.88	63.69
SST_TS_3f (Ours)	3	78.66	49.35	**83.81**	**65.06**

Table 6. Localization quality test with stricter IoU threshold. The normal and strict thresholds for vehicles are 0.7 and 0.8, and are 0.5 and 0.6 for pedestrians. $*$: the results are from [40]. TTA: test-time data augmentation. †: the offline setting using all the past and future frames in a point cloud sequence.

PointPillars, we reduce the stride of the first two convolutions in HRNet from 2 to 1. All the alternatives have the same setting with SST_1f except their backbones. Table 7 shows the comparison between different models.

Models	Vehicle 3D AP	Pedestrian 3D AP	#param.	Latency (ms)	Memory (GB)
PointPillars	64.01	60.85	6.4M	**60**	**5.4**
PointPillars-SS	64.69	64.32	6.4M	185	8.5
SparsePillars-SS	51.57	61.55	6.4M	67	5.8
SparsePillars-SS$_{5\times5}$†	55.40	61.28	17.1M	81	5.9
SparsePillars-SS$_{7\times7}$†	56.77	60.87	33.9M	97	5.9
HRNetV2p-W18 [55]	64.38	61.09	26.2M	130	7.6
SST_1f	**67.86**	**70.94**	**1.6M**	97	6.8

Table 7. Comparison with alternatives to SST. Using 20% data for training. The latency is evaluated with standard benchmarking script in MMDetection3D on 2080Ti GPUs. †: The size of all kernels in sparse convolution increases to 5×5 or 7×7.

In Table 7, our method outperforms all other alternatives with relatively low latency. Besides, two things need to be noticed: (1) **SparsePillars-SS** is much worse than other models in vehicle class. Due to the properties of submanifold sparse convolution, this model suffers from more severe receptive field shrinkage than PointPillars-SS. For example, if a vehicle part is isolated with all the surrounding voxels being in empty, it can not perceive information from other parts in the whole forward process. On the contrary, the attention mechanism in SST well addresses this issue while maintaining sparsity. (2) **HRNetV2p-W18** allocates too much computation on the high-stride (low resolution) branches which is not needed in 3D object detection. So the capacity of its high-resolution branch is limited, leading to its inferior performance.

5.5. Qualitative Analysis of Sparse Attention

We visualize the attention weights in Fig. 5 and list our observations as follows.

Sufficient Coverage In Fig. 5 (a) *Complete Vehicle*, the query token (pink dot) in the car has strong relation with all other parts of the car. In other words, this single token can effectively cover the whole car. This demonstrates that the attention mechanism is indeed effective to enlarge the receptive field.

Semantic Discrimination In Fig. 5 (b) *Person near a Wall*, the query token on the person builds strong dependency with other body parts, but has little relations with background points, *e.g.*, wall. In Fig. 5 (c) *Person beside a Vehicle*, the pedestrian standing next to the vehicle attends only with itself. These two cases reveal that the learned sparse attention weight is discriminative between different semantic classes. This property helps distinguish pedestrians from other slim objects and reduces false positives.

Instance Discrimination In the crowded cases, such as Fig. 5 (d) *Multiple Pedestrians*, the query token in a person mainly focuses on the same person. Due to the high semantic similarity, it also slightly attends to other people. In Fig. 5 (e) *Multiple Vehicles*, the query token in the vehicle almost has no dependency on the nearby vehicles. These two cases suggest that the learned sparse attention weights are discriminative for different instances.

5.6. Hyper-parameter Ablation

Region Size We show the performance under different region sizes for *Regional Grouping* in Table 8. SST is in general robust to the region size and slightly better with larger regions. Especially, SST has the best performance in pedestrian detection with the largest local region size. It suggests that the local context is helpful to recognize pedestrians. For example, pedestrians are more likely to appear on the sidewalks than on vehicle lanes.

Region Size	Max number of voxels	LEVEL_1 AP/APH	
		Vehicle	Pedestrian
$3.20m$	100	66.9/66.4	70.4/56.9
$3.84m$	144	**67.9/67.3**	70.9/**57.3**
$4.48m$	196	67.8/67.3	70.6/56.5
$5.12m$	256	66.9/66.3	**71.1**/57.1

Table 8. Ablation of the region size. Using 20% data for training.

Network Depth SST is relatively shallow by design thanks to the large receptive fields from the attention mechanism. In Table 9 we show the impact of model depths on SST. In general, SST is robust to different depths, and the performance of pedestrian class is even slightly better with fewer layers. This demonstrates our method does not rely on a very large or deep model, thus it is easier to build efficient single-stride models.

Number of SRA blocks	LEVEL_1 AP/APH Vehicle	LEVEL_1 AP/APH Pedestrian
5	67.7/67.1	**71.1/57.9**
6	**67.9/67.3**	70.9/57.3
7	67.8/67.3	70.6/56.1

Table 9. Ablation of the network depth. Using 20% training data.

6. Conclusion and Limitations

In this paper, we analyze the impact of the network stride on 3D object detectors for autonomous driving, and empirically show that 3D object detectors do not really need downsampling. To build a single-stride network, we adopt the sparse regional attention to address the problem of insufficient receptive fields and avoid expensive computation. By stacking the sparse attention modules, we propose the Single-stride Sparse Transformer, achieving state-of-the-art performance on the Waymo Open Dataset. Due to the single stride structure, our models obtain remarkable performance on the challenging pedestrian class. Without elaborated optimization, our model uses slightly more memory than baseline models, and we will pursue a more memory-friendly model in the future. We wish our work could break the stereotype in the backbone design of point cloud data, and inspire more thoughts on the specialized architectures.

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. 3, 6

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end Object Detection with Transformers. In *ECCV*, 2020. 4

[3] Yuning Chai, Pei Sun, Jiquan Ngiam, Weiyue Wang, Benjamin Caine, Vijay Vasudevan, Xiao Zhang, and Dragomir Anguelov. To the Point: Efficient 3D Object Detection in the Range Image With Graph Convolution Kernels. In *CVPR*, 2021. 3, 6, 7

[4] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. MMDetection: Open MMLab Detection Toolbox and Benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 7

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017. 2

[6] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-View 3D Object Detection Network for Autonomous Driving. In *CVPR*, 2017. 3

[7] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast Point R-CNN. In *ICCV*, 2019. 2

[8] Jan Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based Models for Speech Recognition. *NeurIPS*, 2015, 2015. 3

[9] MMDetection3D Contributors. MMDetection3D: OpenMMLab Next-generation Platform for General 3D Object Detection. https://github.com/open-mmlab/mmdetection3d, 2020. 3, 5

[10] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection. In *AAAI*, 2021. 6

[11] 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. 3

[12] 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. In *ICLR*, 2020. 3

[13] Alaaeldin El-Nouby, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. XCiT: Cross-Covariance Image Transformers. *arXiv preprint arXiv:2106.09681*, 2021. 13

[14] Lue Fan, Xuan Xiong, Feng Wang, Naiyan Wang, and ZhaoXiang Zhang. RangeDet: In Defense of Range View for LiDAR-Based 3D Object Detection. In *ICCV*, 2021. 1, 2, 3, 6, 7, 13

[15] Cheng-Yang Fu, Wei Liu, Ananth Ranga, Ambrish Tyagi, and Alexander C Berg. DSSD: Deconvolutional Single Shot Detector. *arXiv preprint arXiv:1701.06659*, 2017. 3

[16] 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. 3, 6

[17] Benjamin Graham and Laurens van der Maaten. Submanifold Sparse Convolutional Networks. *arXiv preprint arXiv:1706.01307*, 2017. 3, 4, 7, 12

[18] 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. 3

[19] Mate Kisantal, Zbigniew Wojna, Jakub Murawski, Jacek Naruniec, and Kyunghyun Cho. Augmentation for Small Object Detection. *arXiv preprint arXiv:1902.07296*, 2019. 3

[20] 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. 1, 2, 3, 4, 5, 6, 7, 8, 13

[21] Hei Law and Jia Deng. CornerNet: Detecting Objects as Paired Keypoints. In *ECCV*, 2018. 6

[22] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle Detection from 3D Lidar Using Fully Convolutional Network. 2016. 3

[23] Jiale Li, Hang Dai, Ling Shao, and Yong Ding. From Voxel to Point: IoU-guided 3D Object Detection for Point Cloud with Voxel-to-Point Decoder. In *ACM-MM*, 2021. 6

[24] Yanghao Li, Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. Scale-Aware Trident Networks for Object Detection. In *ICCV*, 2019. 2, 3

[25] Zhichao Li, Feng Wang, and Naiyan Wang. LiDAR R-CNN: An Efficient and Universal 3D Object Detector. In *CVPR*, 2021. 2, 3, 5, 6, 7, 13

[26] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. In *CVPR*, 2017. 2, 3

[27] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. In *ICCV*, 2017. 4

[28] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *European conference on computer vision*. Springer, 2014. 2, 3

[29] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single Shot Multibox Detector. In *ECCV*, 2016. 4

[30] 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. 3, 4, 5, 12, 13

[31] Jiageng Mao, Minzhe Niu, Haoyue Bai, Xiaodan Liang, Hang Xu, and Chunjing Xu. Pyramid R-CNN: Towards Better Performance and Adaptability for 3D Object Detection. In *ICCV*, 2021. 6, 13

[32] 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. 3

[33] 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. 6

[34] 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 *CVPR*, 2019. 3, 6, 7

[35] Ishan Misra, Rohit Girdhar, and Armand Joulin. An End-to-End Transformer Model for 3D Object Detection. In *ICCV*, 2021. 3

[36] Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3d Object Detection with Pointformer. In *CVPR*, 2021. 3

[37] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum PointNets for 3D Object Detection from RGB-D Data. In *CVPR*, 2018. 2

[38] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *CVPR*, 2017. 2, 5

[39] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *NeurIPS*, 2017. 2

[40] Charles R Qi, Yin Zhou, Mahyar Najibi, Pei Sun, Khoa Vo, Boyang Deng, and Dragomir Anguelov. Offboard 3D Object Detection from Point Cloud Sequences. In *CVPR*, 2021. 6, 7, 8

[41] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-Alone Self-Attention in Vision Models. *NeurIPS*, 32, 2019. 3

[42] Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. DynamicViT: Efficient Vision Transformers with Dynamic Token Sparsification. *arXiv preprint arXiv:2106.02034*, 2021. 13

[43] 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. 3, 6, 7, 8, 12, 13

[44] 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. 3, 6, 7

[45] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud. In *CVPR*, 2019. 2

[46] 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. 6, 7, 13

[47] Abhinav Shrivastava, Rahul Sukthankar, Jitendra Malik, and Abhinav Gupta. Beyond Skip Connections: Top-down Modulation for Object Detection. *arXiv preprint arXiv:1612.06851*, 2016. 3

[48] Bharat Singh and Larry S Davis. An Analysis of Scale Invariance in Object Detection - SNIP. In *CVPR*, 2018. 2

[49] Bharat Singh, Mahyar Najibi, and Larry S Davis. SNIPER: Efficient Multi-Scale Training. In *NeurIPS*, 2018. 2, 3

[50] 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. 2, 3, 5

[51] 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. 3, 6, 7, 12

[52] OpenPCDet Development Team. OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds. https://github.com/open-mmlab/OpenPCDet, 2020. 3

[53] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training Data-efficient Image Transformers & Distillation through Attention. In *ICML*, 2021. 3

[54] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *NeurIPS*, 2017. 3, 4

[55] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep High-resolution Representation Learning for Visual Recognition. *IEEE transactions on pattern analysis and machine intelligence*, 2020. 3, 7, 8

[56] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with Linear Complexity. *arXiv preprint arXiv:2006.04768*, 2020. 13

[57] Tao Wang, Li Yuan, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. PnP-DETR: Towards Efficient Visual Analysis with Transformers. In *ICCV*, 2021. 13

[58] Yue Wang, Alireza Fathi, Abhijit Kundu, David Ross, Caroline Pantofaru, Tom Funkhouser, and Justin Solomon. Pillar-based Object Detection for Autonomous Driving. In *ECCV*, 2020. 3, 6, 7

[59] Xinshuo Weng and Kris Kitani. A Baseline for 3D Multi-object Tracking. *arXiv preprint arXiv:1907.03961*, 2019. 7

[60] Kan Wu, Houwen Peng, Minghao Chen, Jianlong Fu, and Hongyang Chao. Rethinking and Improving Relative Position Encoding for Vision Transformer. In *ICCV*, 2021. 13

[61] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. DOTA: A Large-scale Dataset for Object Detection in Aerial Images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018. 3

[62] Yan Yan, Yuxing Mao, and Bo Li. SECOND: Sparsely Embedded Convolutional Detection. *Sensors*, 18(10), 2018. 1, 2, 4, 6, 7, 12

[63] Chenhongyi Yang, Zehao Huang, and Naiyan Wang. QueryDet: Cascaded Sparse Query for Accelerating High-Resolution Small Object Detection. *arXiv preprint arXiv:2103.09136*, 2021. 3

[64] Zetong Yang, Yin Zhou, Zhifeng Chen, and Jiquan Ngiam. 3D-MAN: 3D Multi-Frame Attention Network for Object Detection. In *CVPR*, 2021. 6, 7

[65] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3D Object Detection and Tracking. *arXiv preprint arXiv:2006.11275*, 2020. 1, 2, 3, 4, 6, 7, 12, 13

[66] Fisher Yu and Vladlen Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. In *ICLR*, 2016. 2

[67] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring Self-attention for Image Recognition. In *CVPR*, 2020. 3

[68] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point Transformer. In *ICCV*, 2021. 3

[69] 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 *CoRL*, 2020. 6, 7

[70] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In *CVPR*, 2018. 2, 4

[71] Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Haibin Ling, Qinghua Hu, Qinqin Nie, Hao Cheng, Chenfeng Liu, Xiaoyu Liu, et al. Visdrone-det2018: The Vision Meets Drone Object Detection in Image Challenge Results. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018. 3

[72] Barret Zoph, Ekin D Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V Le. Learning Data Augmentation Strategies for Object Detection. In *ECCV*. Springer, 2020. 3

Appendices

A. Submission on Test Server

Due to the submission frequency limit of the Waymo test server, we only report the results of our best model. We compare SST with the three most competitive methods and report their performances in the multi-frame setting from the official leaderboard. The results are shown in Table A and Table B. The performance of SST on vehicle class is comparable with these methods, and the performance of SST on pedestrian class significantly outperforms other methods.

Methods	LEVEL_1 3D AP/APH	LEVEL_2 3D AP/APH
PVRCNN_2f [43]	81.06/80.57	73.69/73.23
CenterPoint_2f [65]	81.05/80.59	73.42/72.99
RSN_3f [51]	80.70/80.30	71.90/71.60
SST_TS_3f (Ours)	80.99/80.62	73.08/72.74

Table A. Performance of **vehicle** detection on test split of Waymo Open Dataset.

Methods	LEVEL_1 3D AP/APH	LEVEL_2 3D AP/APH
PVRCNN_2f [43]	80.31/76.28	73.98/70.16
CenterPoint_2f [65]	80.47/77.28	74.56/71.52
RSN_3f [51]	78.90/75.60	70.70/67.80
SST_TS_3f (Ours)	83.05/79.38	76.65/73.14

Table B. Performance of **pedestrian** detection on test split of Waymo Open Dataset.

B. Discussion of Sparse Operations

Due to the space limit of the main paper, we leave the discussion on sparse operations in the supplementary materials. In this section, we discuss two problems for sparse operations: (1) *insufficient receptive field of submanifold sparse convolution (SSC)* [17], and (2) *the difficulties of downsampling/upsampling in sparse data*.

B.1. Insufficient Receptive Field of Submanifold Sparse Convolution (SSC)

In Sec. 1 and Table 7 in our main paper, we briefly point out that the SSC-based single-stride architecture faces a severe problem of the insufficient receptive field. We demonstrate this issue here in Fig. A by comparing the behaviors of SSC and standard 2D convolution in sparse data. Both the SSC and standard convolutions have two layers with a kernel size of three. However, the SSC could not reach the voxel on the top-left corner from the voxel marked with a star, while the standard convolution is capable of doing this. This example intuitively illustrates the insufficiency of receptive fields for SSC, and we explain the reasons in detail as follows.

The SSC do not "fill" empty voxels for the sake of efficiency, which largely constrains the information communication between voxels. Under such conditions, in Fig. A (a), only one voxel (the pink one) in has information communication with the one marked by a red star if the kernel size is 3×3. On the contrary, Fig. A (b) shows that the 2D convolution can gradually enlarge the receptive field by involving the empty voxels in the convolution process, which is more effective for aggregating information compared to the SSC.

To give an experimental illustration, we conduct experiments on the class of vehicles, which require sufficient receptive field for detection. In the Table 7 of the main paper, replacing the 3×3 standard convolutions with SSC will cause a significant drop of AP from 64.69 to 51.57. We further increase the receptive field by expanding the kernel size of SSC to 5×5 and 7×7. These improve the performance from the 3D AP 51.57 to 55.40 and 56.77, but there is still a large gap to the variant using standard convolutions. Therefore, these numbers support our analyses on the insufficient receptive fields of SSC.

B.2. Downsampling/Upsampling in Sparse Data

Although downsampling and upsampling are common in dense data, *e.g.*, pooling in CNN, token merge in Swin-Transformer, it is non-trivial to transfer these techniques to sparse data like point clouds. A variant of SSC named Sparse Convolution (SC) follows the standard convolution to implement the downsampling and upsampling in sparse data. With such implementation, data loses sparsity rapidly [62, 65] and this leads to high computational overhead.

In our sparse Transformer, downsampling/upsampling by token merge [30] also needs careful consideration. First, the downsampling operation is still an open problem for point clouds: what is the best way to merge the varied number of tokens scattered in different spatial locations? Second, the upsampling operation is also non-trivial and requires future research: how to recover a couple of tokens in different locations from a single token effectively and efficiently? In developing the SST, we encounter these challenges and find it difficult to offer satisfying solutions. Although we have bypassed these difficulties by adopting the single-stride architecture, we hope future research may work on this downsampling/upsampling question and better utilizes sparse data.

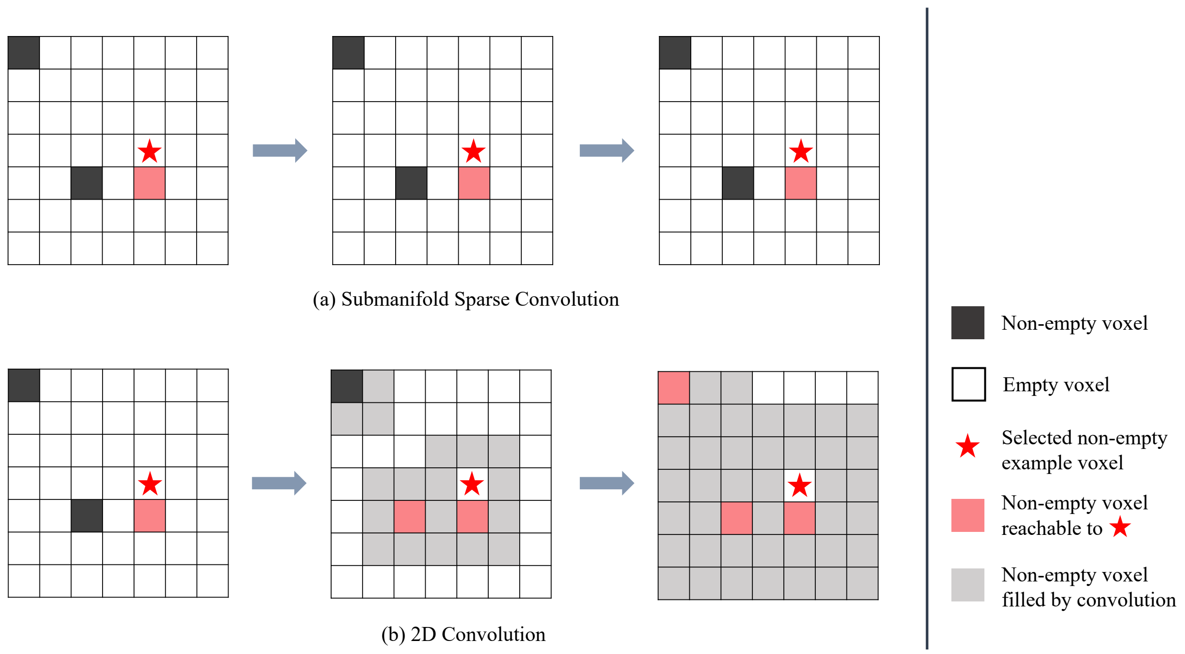

(a) Submanifold Sparse Convolution
(b) 2D Convolution
Non-empty voxel
Empty voxel
Selected non-empty example voxel
Non-empty voxel reachable to ★
Non-empty voxel filled by convolution

Figure A. Illustration of receptive enlarging in the 3×3 submanifold sparse convolution (SSC) and the standard 3×3 convolution. In SSC, only the information of the voxel (the pink one) covered by the kernel can reach to the red star. In 2D convolution, all non-empty voxels can reach to the red star after 2 convolution layers, because the empty locations are "filled" by the convolution.

C. Potential Improvements

In order to rule out unimportant factors and present a clean architecture, we only inherit the basic framework of PointPillars [20]. So there is a large room for further performance improvements, and we list some of them as follows. We will adopt these techniques in our future work.

IoU Prediction. In detection, the classification score of a bounding box are not always consistent with the real regression quality. So many recent methods [14,43,46,65] use another branch to predict the IoU between output bounding boxes and the corresponding ground-truth boxes, and use the predicted IoU to correct the classification scores.

More Powerful Second Stage. We use LiDAR-RCNN [25] as our second stage, which is a lightweight PointNet-like module only takes the raw point cloud as input. So it has no effect on our first stage and is convenient for our analysis of single-stride architecture. However, its performance is inferior to some other elaborately designed RCNNs, *e.g.*, CenterPoint [65], PartA2 [46], PVR-CNN [43], PyramidRCNN [31], which reuse the features from the single stage to achieve better refinement. With the point-level features interpolated from feature maps in the first stage, SST can be equipped with most of these methods and aim for better abilities.

Incorporating Advanced Techniques in Vision Transformer. We have witnessed the fast progress of vision transformers. Many advanced techniques can be borrowed to enhance the performance of SST. **(1) Better efficiency**: There are a lot of techniques can be adopted to improve our efficiency, for example, token selection [42, 57], attention simplication [56]. **(2) Better efficacy**: Some techniques can be used to make SST more effective, *e.g.*, relative positional encoding [60], different attention mechanism [13].

D. Computational Complexity Compared with Convolutions

We investigate the computational complexity of the SST architecture and convolutional architectures. Our analyses demonstrate that SST has a unique advantage in efficiency by utilizing the sparsity of point clouds and the regional grouping.

Following the calculation in Swin-Transformer [30], we inspect the computational complexities of convolutional architectures and SST. For an input scene size of $h \times w$, a convolution layer with kernel size $k \times k$ and channel number C has the complexity as Equation A. On the same scene, an SRA operation has the complexity as Equation B, where it has H-heads, region size of $R \times R$, and the average sparsity as S, which is the ratio for non-empty voxels[3].

$$\Omega(\text{Conv}) = hwk^2C^2, \tag{A}$$
$$\Omega(\text{SRA}) = 4ShwC^2 + 2HS^2R^2hwC, \tag{B}$$

As shown in the equations, the computational complexities for convolutions and SRA operations are all $O(hw)$, thus are both linear to the scale of input. However, the SRA operations have the linear factor of S, which is generally small due to the sparsity of point clouds. According to our statistics, S roughly equals to 0.09 on Waymo Open Dataset with our voxelization. Such an analysis indicates that our SRA operations is efficient by properly exploiting the sparsity of LiDAR data.

[3]Our calculation is approximate because we assume non-empty voxels uniformly scatter in the space.