Embracing Single Stride 3D Object Detector with Sparse Transformer

Lue Fan\textsuperscript{1,4,6,7} Ziqi Pang\textsuperscript{2} Tianyuan Zhang\textsuperscript{3} Yu-Xiong Wang\textsuperscript{2}
Hang Zhao\textsuperscript{5} Feng Wang\textsuperscript{8} Naiyan Wang\textsuperscript{8} Zhaoxiang Zhang\textsuperscript{1,4,6,9}

\textsuperscript{1}Institute of Automation, Chinese Academy of Sciences \textsuperscript{2}University of Illinois Urbana-Champaign \textsuperscript{3}Carnegie Mellon University \textsuperscript{4}University of Chinese Academy of Sciences \textsuperscript{5}Tsinghua University \textsuperscript{6}National Laboratory of Pattern Recognition, CASIA \textsuperscript{7}School of Future Technology, UCAS \textsuperscript{8}TuSimple \textsuperscript{9}Center for Artificial Intelligence and Robotics, HKISI CAS

\{fanlue2019, zhaoxiang.zhang\}@ia.ac.cn \{ziqip2, yxw\}@illinois.edu tianyuaz@andrew.cmu.edu
hangzhao@mail.tsinghua.edu.cn \{feng.wff, winsty\}@gmail.com

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 \textsc{level1} AP on validation split) on small object (pedestrian) detection due to the characteristic of single stride. Our codes will be public soon.

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 Eye’s View (BEV), which makes it convenient to borrow advanced techniques from the 2D counterparts. Many works \cite{13, 19, 57, 60} 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 \cite{48}, the perception range is usually 150m × 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 [23, 25, 46, 47] 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) [13, 19, 57, 60, 65]. 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 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, 61] 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.

---

**Figure 2. Distribution of the relative object size **$S_{rel}$** in COCO dataset [27] 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 × 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.**

---

2We provide a clear illustration for this in our supplementary materials.

---

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 [37, 38] are widely adopted for feature learning of small region of irregular points [7, 24, 36, 43]. Voxel-based representation [19, 57, 60, 65] combined with convolutions are the most popular treatment. As explored in several recent works [1, 3, 13, 21, 33], 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, 24, 41, 42, 49, 54].

**Transformers in Visual Recognition** The success of transformer architectures in NLP [11, 52] and speech recognition [8] has inspired lots of work to investigate the power of attention in visual recognition [12, 29, 40, 51, 62]. The pioneering work ViT [12] splits an image into patches, and then feeds sequences of patches to multiple transformer blocks for image classification. DeiT [51] explores training strategies for data-efficient learning of vision transformers. Swin-Transformer [29] 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 [17, 34, 63]. For autonomous driving scenarios, Pointformer [35] proposes a point-based local and global attention module directly operating on point clouds. In addition, VoTr [31] uses the local self-attention module to replace the sparse convolution [16] for voxel processing, where each voxel serves as a query and attends with its neighbor voxels.

**Small Object Detection** Small object detection [27, 56, 66] is a challenging track in 2D object detection. The mainstream of current methods [14, 25, 45, 53, 58] 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 [23, 25, 47] and strong data augmentations [18, 67]. 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 [13, 15, 60] 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 [19] in MMDetection3D [9] as our base model. The experiments are conducted on Waymo Open Dataset [48]. We uniformly sample 20% training data (32K frames) 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.

To reduce memory overhead, we change the filter number in convolution layers. 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.

<table>
<thead>
<tr>
<th>Models</th>
<th>Vehicle</th>
<th>Pedestrian</th>
<th>Cyclist</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_3$</td>
<td>63.66</td>
<td>60.82</td>
<td>47.08</td>
<td>58ms</td>
</tr>
<tr>
<td>$D_2$</td>
<td>64.01 ↑ 0.35 60.85 ↑ 0.03 47.52 ↑ 0.44 60ms</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_1$</td>
<td>66.03 ↑ 2.02 65.06 ↑ 4.21 52.97 ↑ 5.45 91ms</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_0$</td>
<td>64.69 ↓ 1.34 64.32 ↓ 0.74 53.02 ↓ 0.05 185ms</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_0^{\text{dilation}}$</td>
<td>66.26 ↑ 1.57 63.51 ↓ 0.81 50.95 ↓ 2.07 192ms</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D_0^{\times 5}$</td>
<td>64.42 ↑ 1.77 65.71 ↑ 1.41 53.70 ↓ 0.68 340ms</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 $D_0^{\text{dilation}}$ and $D_0^{\times 5}$, the arrows indicate performance changes based on $D_0$.

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.
- 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 [52], and present our convolutional single stride architecture. It is difficult to simultaneously satisfy the two factors with computational cost. However, as we discussed in Sec. 1, 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 [19, 57, 65]. 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 [29]. 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 $\mathcal{I}$, SST forms conventional transformers as follows

$$
\mathcal{F'} = \text{MSA}(\text{LN}(\mathcal{F}), \text{PE}(\mathcal{I})) + \mathcal{F} \\
\tilde{\mathcal{F}} = \text{MLP}(\text{LN}(\mathcal{F'})) + \mathcal{F'} 
$$

(1)

where PE(·) stands for the absolute positional encoding function used in [2]. MSA(·) denotes the Multi-head Self-Attention, and LN(·) 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:

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

(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, 52], 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 [16, 57].

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. [19, 60]. So we add two $3 \times 3$ 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 [19] for simplicity. Specifically, we use the SSD [28] head, the smooth L1 bounding box localization loss $\mathcal{L}_{loc}$, the classification loss $\mathcal{L}_{cls}$ in the form of focal loss [26], 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.

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

(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 [24] as our second stage. LiDAR-RCNN is a lightweight second stage network consists of a simple PointNet [37] 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 [29] 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) [48]. 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 150 m × 150 m. 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 [19]. 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 3.84 m × 3.84 m × 6 m. As for other parts, SST follows the implementation of PointPillars in MMDetection3D. We use the BEV pillar size of 0.32 m × 0.32 m × 6 m, 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 egopose. SST_TS_1f and SST_TS_3f: two stage model based on above models, using a standard LiDAR-RCNN [24] 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 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 [49] 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 [20]. To verify this, we evaluate SST with higher 3D IoU thresholds (0.8 for 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 [49] 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 [42].

vehicle, 0.6 for pedestrian). In Table 6, we compare our models with the PointPillars baseline and other models with available results from [39], then a couple of interesting findings emerge:

1. Comparing MVF++ [39] 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 [39] is an offboard method using all the past and future frames in a sequence (around 200 frames) and is equipped with tracking [55]. 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

<table>
<thead>
<tr>
<th>Methods</th>
<th>LEVEL_1</th>
<th>LEVEL_2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3D AP/APH</td>
<td>3D AP/APH</td>
</tr>
<tr>
<td>One-Stage Methods</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SECOND † [57]</td>
<td>72.27/71.69</td>
<td>63.85/63.33</td>
</tr>
<tr>
<td>MVF [64]</td>
<td>62.93/-</td>
<td>-/-</td>
</tr>
<tr>
<td>LaserNet ‡ [33]</td>
<td>56.10/-</td>
<td>-/-</td>
</tr>
<tr>
<td>AFDet [15]</td>
<td>63.69/-</td>
<td>-/-</td>
</tr>
<tr>
<td>Pillar-OD [54]</td>
<td>69.80/-</td>
<td>-/-</td>
</tr>
<tr>
<td>PPC [3]</td>
<td>65.2/-</td>
<td>-/-</td>
</tr>
<tr>
<td>VoTr-SSD [32]</td>
<td>68.99/68.39</td>
<td>60.22/59.69</td>
</tr>
<tr>
<td>RangeDet [13]</td>
<td>72.85/72.33</td>
<td>64.03/63.57</td>
</tr>
<tr>
<td>CenterPoint-Voxel [60]</td>
<td>74.78/74.22</td>
<td>66.70/66.19</td>
</tr>
<tr>
<td>PointPillars* [19]</td>
<td>72.08/71.53</td>
<td>63.55/63.06</td>
</tr>
<tr>
<td>SST_{1f} (Ours)</td>
<td>74.22/73.77</td>
<td>65.47/65.07</td>
</tr>
<tr>
<td>SST_{3f} (Ours)</td>
<td>77.04/76.56</td>
<td>68.50/68.08</td>
</tr>
<tr>
<td>Two-Stage Methods</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Voxel RCNN [10]</td>
<td>75.59/-</td>
<td>66.59/-</td>
</tr>
<tr>
<td>RCD [1]</td>
<td>69.06/68.5</td>
<td>-/-</td>
</tr>
<tr>
<td>VoTr-TSD [32]</td>
<td>74.95/74.25</td>
<td>65.91/65.29</td>
</tr>
<tr>
<td>LiDAR-RCNN [24]</td>
<td>76.07/55.5</td>
<td>68.3/67.9</td>
</tr>
<tr>
<td>Pyramid RCNN [30]</td>
<td>76.30/75.68</td>
<td>67.23/66.68</td>
</tr>
<tr>
<td>Voxel-to-point [22]</td>
<td>77.24/-</td>
<td>69.77/-</td>
</tr>
<tr>
<td>3D-MAN [59]</td>
<td>74.53/74.03</td>
<td>67.61/67.14</td>
</tr>
<tr>
<td>Part-A2-Net † [44]</td>
<td>77.05/76.51</td>
<td>68.47/67.97</td>
</tr>
<tr>
<td>CenterPoint-Pillar [60]</td>
<td>76.10/75.50</td>
<td>68.00/67.50</td>
</tr>
<tr>
<td>CenterPoint-Voxel [60]</td>
<td>76.59/7605</td>
<td>68.85/68.35</td>
</tr>
<tr>
<td>PV-RCNN [41]</td>
<td>77.51/76.89</td>
<td>68.98/68.41</td>
</tr>
<tr>
<td>PV-RCNN++ [42]</td>
<td>78.79/78.21</td>
<td>70.26/69.71</td>
</tr>
<tr>
<td>RSN_{1f} † [49]</td>
<td>75.10/74.60</td>
<td>66.00/65.50</td>
</tr>
<tr>
<td>RSN_{3f} † [49]</td>
<td>78.40/78.10</td>
<td>69.50/69.10</td>
</tr>
<tr>
<td>SST_{TS, 1f} (Ours)</td>
<td>76.22/75.79</td>
<td>68.04/67.64</td>
</tr>
<tr>
<td>SST_{TS, 3f} (Ours)</td>
<td>78.66/78.21</td>
<td>69.98/69.57</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Method</th>
<th>LEVEL_1 Pedestrian AP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Overall</td>
</tr>
<tr>
<td></td>
<td>0-30m</td>
</tr>
<tr>
<td></td>
<td>30-50m</td>
</tr>
<tr>
<td></td>
<td>50m-inf</td>
</tr>
<tr>
<td>RSN_1f</td>
<td>77.8</td>
</tr>
<tr>
<td>RSN_3f</td>
<td>79.0</td>
</tr>
<tr>
<td>PointPillars</td>
<td>70.6</td>
</tr>
<tr>
<td>PointPillars_3f</td>
<td>73.7</td>
</tr>
<tr>
<td>SST_1f</td>
<td>78.7*</td>
</tr>
<tr>
<td>SST_3f</td>
<td>82.4*</td>
</tr>
</tbody>
</table>

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.

is capable of better localizing objects with full and fine-grained information.

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.

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 [39]. TTA: test-time data augmentation. †: the offline setting using all the past and future frames in a point cloud sequence.

**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 [16, 57]. 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** [53]: HRNet maintains the high resolution while building multi-scale features. We adopt the standard HRNetV2p-W18 from MMDetection [4] for the exper-
to keep the output resolution in HRNet the same as 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 except their backbones. Table 7 shows the comparison between different models.

<table>
<thead>
<tr>
<th>Models</th>
<th>Vehicle 3D AP</th>
<th>Pedestrian 3D AP</th>
<th>#param.</th>
<th>Latency (ms)</th>
<th>Memory (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PointPillars</td>
<td>64.01</td>
<td>60.85</td>
<td>6.4M</td>
<td>60</td>
<td>5.4</td>
</tr>
<tr>
<td>PointPillars-SS</td>
<td>64.69</td>
<td>64.32</td>
<td>6.4M</td>
<td>185</td>
<td>8.5</td>
</tr>
<tr>
<td>SparsePillars-SS</td>
<td>51.57</td>
<td>61.55</td>
<td>6.4M</td>
<td>67</td>
<td>5.8</td>
</tr>
<tr>
<td>SparsePillars-SS4×5†</td>
<td>55.40</td>
<td>61.28</td>
<td>17.1M</td>
<td>81</td>
<td>5.9</td>
</tr>
<tr>
<td>SparsePillars-SS7×7†</td>
<td>56.77</td>
<td>60.87</td>
<td>33.9M</td>
<td>97</td>
<td>5.9</td>
</tr>
<tr>
<td>HRNetV2p-W18 [53]</td>
<td>64.38</td>
<td>61.09</td>
<td>26.2M</td>
<td>130</td>
<td>7.6</td>
</tr>
<tr>
<td>SST_1f</td>
<td>67.86</td>
<td>70.94</td>
<td>1.6M</td>
<td>97</td>
<td>6.8</td>
</tr>
</tbody>
</table>

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 $5 \times 5$ or $7 \times 7$.

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.

<table>
<thead>
<tr>
<th>Region Size</th>
<th>Max number of voxels</th>
<th>LEVEL_1 AP/APH</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.20m</td>
<td>100</td>
<td>66.9/66.4</td>
</tr>
<tr>
<td>3.84m</td>
<td>144</td>
<td>67.9/67.3</td>
</tr>
<tr>
<td>4.48m</td>
<td>196</td>
<td>67.8/67.3</td>
</tr>
<tr>
<td>5.12m</td>
<td>256</td>
<td>66.9/66.3</td>
</tr>
</tbody>
</table>

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

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.

7. Acknowledgements

This work was supported in part by the Major Project for New Generation of AI (No.2018AAA0100400), the National Natural Science Foundation of China (No. 61836014, No. U21B2042, No. 62072457, No. 62006231), and in part by the TuSimple Collaborative Research Project.
References


[19] 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


[22] 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


[39] 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


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


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


