Sparse2Dense: Learning to Densify 3D Features for 3D Object Detection

Tianyu Wang\textsuperscript{1,2,3}, Xiaowei Hu\textsuperscript{3}*, Zhengzhe Liu\textsuperscript{1}, Chi-Wing Fu\textsuperscript{1,2}

\textsuperscript{1} The Chinese University of Hong Kong
\textsuperscript{2} The Shun Hing Institute of Advanced Engineering
\textsuperscript{3} Shanghai AI Laboratory
\{wangty,zzliu,cwf\}@cse.cuhk.edu.hk, huxiaowei@pjlab.org.cn

Abstract

LiDAR-produced point clouds are the major source for most state-of-the-art 3D object detectors. Yet, small, distant, and incomplete objects with sparse or few points are often hard to detect. We present Sparse2Dense, a new framework to efficiently boost 3D detection performance by learning to densify point clouds in latent space. Specifically, we first train a dense point 3D detector (DDet) with a dense point cloud as input and design a sparse point 3D detector (SDet) with a regular point cloud as input. Importantly, we formulate the lightweight plug-in S2D module and the point cloud reconstruction module in SDet to densify 3D features and train SDet to produce 3D features, following the dense 3D features in DDet. So, in inference, SDet can simulate dense 3D features from regular (sparse) point cloud inputs without requiring dense inputs. We evaluate our method on the large-scale Waymo Open Dataset and the Waymo Domain Adaptation Dataset, showing its high performance and efficiency over the state of the arts. The code is available at \url{https://github.com/stevewongv/Sparse2Dense}.

1 Introduction

3D object detection is an important task for supporting autonomous vehicles to sense their surroundings. Previous works \cite{1,4} design various neural network structures to improve the detection performance. Yet, it remains extremely challenging to detect small, distant, and incomplete objects, due to the point sparsity, object occlusion, and inaccurate laser reflection. These issues hinder further improvements in the precision and robustness of 3D object detection.

To improve the detection performance, some works attempt to leverage additional information, e.g., images \cite{5,9}, image segmentations \cite{10,11}, and multi-frame information \cite{12,13}. By fusing the information with the input point cloud in

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure1.png}
\caption{Our approach is able to produce dense and good-quality 3D features (d) from regular (raw) point clouds, enabling better detection of small, distant, and incomplete objects (b) vs. (a,c). Red boxes are detection results and green boxes are the ground truths.}
\end{figure}

*Corresponding author (huxiaowei@pjlab.org.cn)
physical or latent space, the 3D detector can obtain enhanced features for small and distant objects to improve the 3D detection performance. However, the above works require additional data pre-processing and fusion in inference, thereby unavoidably increasing the computational burden and slowing down the overall detection efficiency in practice.

More recently, [14] associates incomplete perceptual features of objects with more complete features of the corresponding class-wise conceptual models via an incomplete-aware re-weighting map and a weighted MSE loss. However, this network still struggles to deal with sparse regions with limited points, due to the difficulty of generating good-quality features in these regions. Recently, [15] proposes to generate semantic points on the predicted object regions and then train modern detectors, leveraging both the generated and original points. However, as the generated points in sparse regions could be incomplete, the generation quality in these regions is still far from satisfactory. Also, it takes a long time to generate the semantic points in large scenes.

In this work, we present a new approach to address the point sparsity issue in 3D object detection. Specifically, we design the Sparse2Dense framework with two detectors: (i) the Dense point 3D Detector (DDet), which is pre-trained with dense point clouds for 3D detection, and (ii) the Sparse point 3D Detector (SDet), which is trained with regular (raw) point clouds as input. Very importantly, when we train SDet, we use DDet to teach SDet to simulate densified 3D features, so that it can learn to produce good-quality 3D features from regular point clouds to improve the detection performance. Unlike previous approaches, we design two effective modules to further help densify the sparse features from regular point clouds in latent space. Also, unlike previous multi-modal approaches [9, 10], which require extra information, like image segmentation and images, in both training and inference, our approach needs dense point data only in training but not in inference.

Our framework is trained in two stages. First, we prepare dense point clouds by fusing multi-frame point clouds for training DDet. Then, we transfer the densified 3D features derived from DDet for embedding into the SDet features when training SDet, such that it can learn to generate densified 3D features, even from regular normal point clouds. To facilitate SDet to simulate dense 3D features, we design the lightweight and effective (S2D) module to densify sparse 3D features in latent space. Also, to further enhance the feature learning, we formulate the point cloud reconstruction (PCR) module to learn to reconstruct voxel-level point cloud as an auxiliary task. Like DDet, we need this PCR module only in training but not in inference.

Furthermore, our framework is generic and compatible with various 3D detectors for boosting their performance. We adopted our framework to work with three different recent works [16, 17, 1] on the large-scale Waymo [18] Open and Waymo Domain Adaptation Datasets, showing that the detection performance of all three methods are improved by our approach. Particularly, the experimental results show that our approach outperforms the state-of-the-art 3D detectors on both datasets, demonstrating the effectiveness and versatility of our approach.

Below, we summarize the major contributions of this work.

(i) We design a new approach to address the point sparsity issue in 3D detection and formulate the Sparse2Dense framework to transfer dense point knowledge from the Dense point 3D Detector (DDet) to the Sparse point 3D Detector (SDet).

(ii) We design the lightweight plug-in S2D module to learn dense 3D features in the latent space and the point cloud reconstruction (PCR) module to regularize the feature learning.

(iii) We evaluate our approach on the large-scale benchmark datasets, Waymo [18] open and Waymo domain adaptation, demonstrating its superior performance over the state of the arts.

2 Related Work

3D Object Detection. Recent years have witnessed the rapid progress of 3D object detection, existing works [19–31] have gained remarkable achievements on 3D object detection. SECOND [16] employs sparse convolution [32, 33] and PointPillars [17] introduces a pillar representation to achieve a good trade-off between speed and performance. Recently, CenterPoint [1] proposes a center-based anchor-free method to localize the object and VoTr [30] joins self-attention with sparse convolution to build a transformer-based 3D backbone. Besides, PV-RCNN [4] fuses deep features by RoI-grid pooling from both point and voxel features, and LiDAR R-CNN [26] presents a PointNet-based
second-stage refinement to address size ambiguity. In this work, we adopt our approach to three recent methods as representatives, i.e., \[16, 17, 1\], showing that our approach can effectively enhance the performance of all three methods, demonstrating the compatibility of our framework.

Sparse/Dense Domain Transformation for 3D Point Cloud. Raw point clouds are typically sparse and incomplete in practice, thereby limiting the performance of many downstream tasks. To address this issue, several approaches, including point cloud upsampling \[34–36\] and completion \[37–40\], have been proposed to densify point cloud and complete the objects to improve the 3D segmentation \[41, 42\] and detection \[43, 14, 15, 31, 44\] performance.

Here, we review some works on sparse/dense domain transformation on 3D object detection. \[45\] first propose a multi-frame to single-frame distillation framework, which only uses five adjacent frames to generate the dense features as the guidance, thus limiting the performance for distant objects. \[43, 14\] present a self-contained method to first extract complete cars at similar viewpoints and high-density regions across the dataset, then merge these augmented cars at the associated ground-truth locations for conceptual feature extraction. Later, \[15\] introduces semantic point generation to address the missing point issue. Recently, \[44\] presents a two-stage framework: first predict 3D proposals then complete the points by another module, which employs an attention-based GNN to refine the detection results with completed objects. The above works \[43, 14, 15, 44\] conduct various operations in the point cloud explicitly, e.g., object extraction and matching \[43, 14\], and point generation \[15, 44\]; however, the explicit point cloud operations lead to two issues. First, it is challenging to conduct the above operations in distant and occluded regions, due to the high sparsity of points, thus severely limiting their performance in these regions. Second, it typically takes a very long time to conduct these operations explicitly, especially for large scenes.

Beyond the prior works, we present an efficient sparse-to-dense approach to learn to densify 3D features in the latent space, instead of explicitly generating points in the point cloud. Notably, our approach needs dense point clouds only in training. In inference, it takes only a regular (sparse) point cloud as input for 3D detection. Quantitative experiments demonstrate that our approach outperforms existing ones in terms of both accuracy and efficiency.

3 Methodology

3.1 Overall Framework Design

Figure 2 gives an overview of our Sparse2Dense framework, which consists of the Dense point 3D Detector (DDet) on top and the Sparse point 3D Detector (SDet) on bottom. Overall, DDet takes a dense point cloud as input, whereas SDet takes a regular (sparse) point cloud as input. Our idea is to transfer dense point knowledge from DDet to SDet and encourage SDet to learn to generate dense 3D features, even with sparse point clouds as inputs, to boost its 3D detection performance. The workflow of our framework design is summarized as follows:

(i) Dense object generation: we prepare dense point clouds for training DDet using raw multi-frame point cloud sequences. Particularly, we design a dense object generation procedure by building voxel grids and filling the voxels with object points (Section 3.2). Then, we replace the object regions in the sparse point cloud \(P^S\) with the corresponding dense object points to obtain the dense point cloud \(P^D\).

(ii) From dense detector to sparse detector: The training has two stages. First, as Figure 2 (a) shows, we train DDet with dense point cloud \(P^D\) with a region proposal network (RPN) to extract region proposals and multiple heads to perform object classification and regression, following VoxelNet-based methods \[1, 16\] or Pillar-based method \[17\]. Second, as Figure 2 (b) shows, we initialize SDet with the weight of the pre-trained DDet, and train SDet with regular sparse point cloud \(P^S\) as input. Meanwhile, we freeze the weights of DDet and adopt an MSE loss to reduce the feature difference \(F^D_a & F^S_a\) between DDet and SDet.

(iii) Dense feature generation by S2D module: MSE loss itself is not sufficient to supervise SDet to effectively simulate dense 3D features like DDet. To complement the MSE loss, we further design the S2D module to learn dense 3D features of objects in latent space. In detail, we feed dense object point cloud \(P^D_O\), which is the object region of the dense point cloud \(P^D\), to DDet and then extract the dense object features \(F^D_o\). After that, we further encourage feature \(F^S_o\) enhanced by S2D to simulate to the dense object feature \(F^D_o\).
Figure 2: The overall framework of our proposed Sparse2Dense. Our framework contains two training stages: (a) in the first stage, we train the Dense point 3D Detector (DDet) by taking the dense point cloud as the input (dark arrows); and (b) in the second stage, we train the Sparse point 3D Detector (SDet) by using the dense features from DDet as the supervision signals (gray and pink arrows). In testing, we only need SDet for 3D object detection on the raw point cloud input (pink arrows), without the DDet and the point cloud reconstruction module.

(iv) **Feature enhancement by point cloud reconstruction (PCR) module:** further, we adopt the PCR module to promote the S2D module to simulate better dense 3D features; as an auxiliary task, PCR takes the feature from S2D module and predicts the voxel mask and point offset for reconstructing the voxel-level dense object point cloud $P^D_D$.

### 3.2 Dense Object Generation

To prepare dense point clouds to train the DDet network, we design an offline pre-processing pipeline to process raw point cloud sequences, each with around 198 frames (see Figure 3 (a)):

(i) First, for each annotated object, we fuse the points inside its bounding box from multiple frames and then filter out the outlier points by using the radius outlier removal algorithm from Open3D [46], as shown in Figure 3 (b).

(ii) To keep the LiDAR-scanned-line patterns and reduce the point number, we voxelize the fused object points and obtain a voxel grid (see Figure 3 (c)) of granularity $(0.1m, 0.1m, 0.15m)$, which is the same size for point cloud voxelization. Specifically, we sort the frames in descending order of the object point number, as shown in Figure 3 (a). Then, we fill the voxel grid with object points starting from the beginning of the sorted frames until more than 95% voxels have been filled (see Figure 3 (d)). Note that we stop filling a single voxel when the number of points in the voxel has reached five or this voxel has been filled by the previous frames to obtain enough points for training.

(iii) For the vehicle category, whose shape is often symmetric, we flip and copy the denser side of each object about its axial plane to further improve its density (see Figure 3 (e)).

### 3.3 Dense Feature Distillation with the S2D Module

To transfer dense point knowledge from DDet to SDet, a straightforward solution is to pair up associated features in DDet and SDet and minimize the distance between each pair of features with an MSE loss. However, MSE loss itself is struggling to achieve satisfying feature transfer, as the inputs of DDet and SDet differ a lot from each other, especially for objects far from the LiDAR sensor. Also, the backbone structure [16][1] built up with VoxelNet consists of SPConv, which processes only non-empty voxels. Hence, it cannot generate features on empty voxels.
Yet, it is extremely challenging to directly reconstruct large-scale dense object points \[15\]. Thus, we propose a voxel-level reconstruction scheme to predict only the average of the input points in each non-empty voxel. Specifically, we decouple this task into two sub-tasks: first predict a soft voxel-occupancy mask indicating the probability that the voxel is non-empty; further predict point offset \(P_{\text{offset}}\) for each non-empty voxel, \(i.e.,\) an offset from voxel center \(V_c\) to the averaged input points of this voxel.

### 3.4 Point Cloud Reconstruction Module

To encourage the S2D module to produce good-quality dense 3D features, we further design the point cloud reconstruction (PCR) module with an auxiliary task. In short, the PCR module reconstructs a voxel-level dense object point cloud \(P^D\) from feature \(F^S\), \(i.e.,\) the output of the S2D module. Yet, it is extremely challenging to directly reconstruct large-scale dense object points \[15\]. Thus, we propose a voxel-level reconstruction scheme to predict only the average of the input points in each non-empty voxel.
Figure 5 shows the architecture of the PCR module. The densified feature $F^S_b$ is first projected back to 3D view and fed to two 3D convolution layers and one transposed 3D convolution to upscale to 1/4 scale of the original input. Then, we predict the voxel-occupancy mask $V_{mask}$ by using one $1 \times 1$ 3D convolution with sigmoid function and predict point offset $P_{offset}$ by using one $1 \times 1$ 3D convolution. Next, we repeat the same steps to further predict voxel mask $V_{mask}$ and point offset $P_{offset}$ at 1/2 scale. Then, the reconstructed point $P_c$ is predicted as

$$P_c = (P_{offset} + V_c) \times V_{mask},$$

(1)

and $P_c$ is optimized to reconstruct the voxelized dense object point cloud $P^D_O$.

### 3.5 Training Loss

Our Sparse2Dense framework is trained in two stages. First, we train DDet with the following loss:

$$L_{DDet} = L_{reg} + L_{hm/cls},$$

(2)

where we adopt an $L_1$ Loss as the regression loss $L_{reg}$ followed [16][17]; heatmap loss $L_{hm}$ is a variant of the focal loss [33] for center-based methods [1][17]; and classification loss $L_{cls}$ is the focal loss for anchor-based method [16]. Here, $L_{hm/cls}$ means we use $L_{hm}$ or $L_{cls}$ depending which method that we adopt our framework to. Second, we train SDet with the following overall loss function:

$$L_{SDet} = L_{reg} + L_{hm/cls} + L_{S2D} + L_{mask} + L_{offset} + L_{hm/dis},$$

(3)

where $L_{S2D}$ is for optimizing S2D; $L_{mask}$ and $L_{offset}$ are for training PCR; and $L_{hm/dis}$ is the distillation loss like $L_{hm}$, but its input is the predicted heat maps of SDet and DDet. In detail, $L_{S2D}$ helps to optimize SDet to learn to densify 3D features based on the associated features in DDet and it is an MSE Loss with the masks to indicate the empty and non-empty elements in the feature maps:

$$L_{S2D} = \beta \frac{1}{|N|} \sum_i (F^S_{a_i} - F^D_{a_i})^2 + \gamma \frac{1}{|N|} \sum_i (F^S_{a_i} - F^D_{a_i})^2 + \beta \frac{1}{|M|} \sum_i (F^S_{b_i} - F^D_{b_i})^2 + \gamma \frac{1}{|M|} \sum_i (F^S_{b_i} - F^D_{b_i})^2,$$

(4)

where $N$ and $\tilde{N}$ are numbers of non-zero and zero values, respectively, on $F^D_a$, while $M$ and $\tilde{M}$ are numbers of non-zero and zero values, respectively, on $F^D_b$. Also, we empirically set $\beta=10$ and $\gamma=20$ to balance the loss weight on non-empty and empty features. $L_{mask}$ and $L_{offset}$ are for training PCR:

$$L_{mask} = \sum_j \left( -\frac{N_b}{N_f} y_j \log(p_j) - (1 - y_j) \log(1 - p_j) \right),$$

(5)

and

$$L_{offset} = \frac{1}{|N_f|} \sum_i (P_{offset,i} + V_{ci}) - P_{gt,i},$$

(6)

where $N_b$ and $N_f$ are numbers of background and foreground voxels, respectively; $p_j$ and $y_j$ are the prediction and ground-truth values of the voxel mask; and $j$ indexes the voxels in $V_{mask}$. Note also that Eq. (3) includes $L_{hm/dis}$, only when adopting our method to the center-based methods [1][17].

### 4 Experiments

#### 4.1 Datasets and Evaluation Metrics

We employ the Waymo Open Dataset and the Waymo Domain Adaptation Dataset [18], which are under the Waymo Dataset License Agreement, to evaluate our framework. Waymo Open Dataset is
We compare our methods with multiple state-of-the-art 3D object detectors [28, 4, 30, 29, 15–17, 1] on three categories, i.e., pedestrian, vehicle, and cyclist, and on two difficulty levels, i.e., level 1 (L1) and level 2 (L2). Note that our method is a plug-and-play module and can be easily adopted to work with various deep-learning-based 3D object detectors. We obtain the results of the existing works by copying from their papers and GitHub [28, 4, 14, 30, 29, 15] or by re-producing their methods using the public code with recommended parameters [1, 16]. We cannot compare with [44], as the authors...
Figure 6: Visual comparison of 3D object detection results and 3D features produced by (a,b,c) baselines [16,17,1] and (d,e,f) our methods (our approach + corresponding baseline), where our approach successfully densifies the object features and help the baseline methods produce more accurate detection results than the three baselines. Note that red boxes show the detection results and green boxes show the ground truths. Orange boxes highlight the improvement brought by our approach.

did not report the performance of their method on the Waymo Open Dataset and did not release code. Among the existing works, [15,14] are the state-of-the-art methods that adopt densified operations explicitly in point clouds and can be adopted to work with other methods.

Table 1 reports the comparison results, where our method clearly improves all three baseline methods (SECOND, PointPillar-center, and CenterPoint) for all categories on all evaluation metrics. Notably, our approach achieves more performance gain in comparison than [14] when working with [16], and outperforms [15] when working with [17,1], even though [15] is built upon a stronger model [4]. Also, level 2 (L2) contains more challenging samples than level 1 (L1), since the point cloud in L2 are much sparser. Despite that, our method consistently shows greater improvement on L2, demonstrating its effectiveness to deal with the sparse point clouds. Furthermore, Table 2 shows the performance gain for three different distant ranges on the Waymo validation set. Our method also achieves significant improvements over all three baseline methods for all distance ranges, including the long-range, showing again that our method can effectively learn to densify 3D features in challenging cases. See more detailed comparison results and the performance of our model trained on the full Waymo Open Dataset in Appendix A. We further provide the visual comparisons in Figure 6, where we can see that: (i) our method successfully detects more objects that contain a few points and compensates sparse features of these objects; (ii) our method generates more accurate 3D bounding boxes, which are consistent with the ground truth boxes; see the orange boxes in the first row; (iii) our method generates more dense and robust features in sparse or distant regions. Please see more visual comparisons in Appendix B.

4.4 Quantitative Comparison on the Waymo Domain Adaptation Dataset

We further evaluate our methods on the Waymo Domain Adaptation Dataset and compare it with the state-of-the-art methods. Table 3 shows that our method is able to consistently improve the performance of all three methods [16,17,1] for all categories and all difficulty levels. Note also
Table 2: Performance gain over baseline approaches on Waymo validation set (level-2) in different ranges. Evaluated on three categories.

<table>
<thead>
<tr>
<th>Methods</th>
<th>All Range</th>
<th>Range [0, 30)</th>
<th>Range [30, 50)</th>
<th>Range [50, +inf)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mAP</td>
<td>mAPH</td>
<td>mAP</td>
<td>mAPH</td>
</tr>
<tr>
<td>SECOND</td>
<td>54.20</td>
<td>49.65</td>
<td>70.58</td>
<td>66.17</td>
</tr>
<tr>
<td>SECOND (ours)</td>
<td>57.21 +3.01</td>
<td>52.91 +3.26</td>
<td>72.94 +2.36</td>
<td>68.91 +2.74</td>
</tr>
<tr>
<td></td>
<td>55.30 +2.97</td>
<td>50.28 +3.33</td>
<td>55.30 +2.97</td>
<td>50.28 +3.33</td>
</tr>
<tr>
<td>CenterPoint-Pillar</td>
<td>61.67</td>
<td>57.94</td>
<td>74.18</td>
<td>70.55</td>
</tr>
<tr>
<td>CenterPoint-Pillar (ours)</td>
<td>66.60 +4.93</td>
<td>63.13 +5.19</td>
<td>77.90 +3.72</td>
<td>74.60 +4.05</td>
</tr>
<tr>
<td>CenterPoint-Voxel</td>
<td>66.04</td>
<td>63.78</td>
<td>80.80</td>
<td>78.86</td>
</tr>
<tr>
<td>CenterPoint-Voxel (ours)</td>
<td>69.19 +3.15</td>
<td>66.88 +3.10</td>
<td>82.72 +1.92</td>
<td>80.77 +1.91</td>
</tr>
</tbody>
</table>

Table 3: Comparisons on the Waymo Domain Adaptation Dataset on 20 validation sequences with existing works. † means re-produced by [15], ‡ means re-produced by us. Still, our re-produced models and most of other re-produced models were trained on 20% data of Waymo Open Dataset following the strategy of [28]; see more details in Sec. 4.2. [15] was trained on two categories (vehicle and pedestrian) while ours were trained on all the three categories.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mAP</td>
<td>mAPH</td>
<td>mAP</td>
<td>mAPH</td>
</tr>
<tr>
<td>SPG † [15]</td>
<td>58.31</td>
<td>49.55</td>
<td>13.96</td>
<td>12.14</td>
</tr>
<tr>
<td>SECOND † [16]</td>
<td>51.56</td>
<td>49.55</td>
<td>12.14</td>
<td>12.14</td>
</tr>
<tr>
<td>SECOND (ours)</td>
<td>55.49</td>
<td>53.96</td>
<td>17.45</td>
<td>15.25</td>
</tr>
<tr>
<td>CenterPoint-Pillar † [17]</td>
<td>54.15</td>
<td>53.26</td>
<td>12.50</td>
<td>10.36</td>
</tr>
<tr>
<td>CenterPoint-Pillar (ours)</td>
<td>59.18</td>
<td>58.52</td>
<td>18.95</td>
<td>16.26</td>
</tr>
<tr>
<td>CenterPoint-Voxel † [1]</td>
<td>57.54</td>
<td>56.99</td>
<td>30.21</td>
<td>28.30</td>
</tr>
<tr>
<td>CenterPoint-Voxel (ours)</td>
<td>60.54</td>
<td>59.87</td>
<td>37.15</td>
<td>35.21</td>
</tr>
</tbody>
</table>

that the most recent state-of-the-art method SPG [15] is trained only on the vehicle and pedestrian categories, yet our method can achieve better performance even when trained on three categories.

4.5 Ablation Study

We conduct experiments to evaluate the key components in our Sparse2Dense framework. Here, we adopt our approach to work with CenterPoint (one stage version) [1]. Then, we conduct the feature distillation (“+ Distillation”) to distill the 3D features from $F^D$ in DDet to $F^S_a$ in SDet, and adopt heat map distillation $L_{hm\_dis}$ between the heat maps of SDet and DDet, as discussed in Sec. 3.5. Next, we further add the S2D module (“+ S2D”) in SDet to enable the feature distillation between the $F^D_b$ in DDet and $F^S_b$ in SDet. In addition, we construct our full pipeline by further adding the point cloud reconstruction module (“+ PCR”). Also, we conduct an additional experiment (“- Distillation”) by ablating the feature loss $L_{S2D}$ and heat map distillation loss $L_{hm\_dis}$ from our full pipeline.

The results are shown in Table 4. First, feature distillation (“+ Distillation”) helps to moderately improve the performance of both categories. By adopting our S2D module (“+ S2D”), we can largely improve the quality of the densified features, boosting the performance of 3D object detection by around 2% on the vehicle and 1% on the cyclist, compared with “+ Distillation”. Next, our point cloud reconstruction module (“+ PCR”) further enhances the performance on both categories and metrics consistently by providing additional supervision to regularize the feature learning. Finally, even without distillation (“- Distillation”), our approach can still improve the baseline performance by more than 2.5% on the vehicle, demonstrating the effectiveness of our S2D and PCR modules.

4.6 Latency Analysis for S2D module

In our framework, both DDet and PCR are used only in the training stages. In inference, we only add the lightweight S2D module into the basic 3D object detection framework. To evaluate the efficiency of the S2D module, we employ the Waymo Open Dataset validation set and report the average processing time with and without the S2D module in inference. Table 5 reports the results, showing that S2D only brings around extra 10 ms latency to detectors, thus demonstrating our approach’s high efficiency. Also, we show the latency of SPG [15] reported in their paper, i.e., 16.9 ms, which is evaluated on the KITTI dataset with a much smaller number of points and objects than the dataset we employed, i.e., Waymo. The latency analysis results manifest the superior efficiency of our S2D in comparison to the state-of-the-art approach [15].
Table 4: Ablation studies on the Waymo Open Dataset validation set.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Vehicle-L2 mAP</th>
<th>Pedestrian-L2 mAP</th>
<th>Cyclist-L2 mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>63.03</td>
<td>62.53</td>
<td>63.72</td>
</tr>
<tr>
<td>+ Distillation</td>
<td>63.84</td>
<td>63.32</td>
<td>67.04</td>
</tr>
<tr>
<td>+ S2D</td>
<td>65.75</td>
<td>65.22</td>
<td>67.62</td>
</tr>
<tr>
<td>+ PCR</td>
<td>66.12</td>
<td>65.58</td>
<td>68.50</td>
</tr>
</tbody>
</table>

Table 5: Latency analysis on our S2D module. We evaluate each model with the batch size of 1. The latency is averaged over the Waymo validation set. As a reference, we include SPG [15] (evaluated on KITTI). Our method needs only \(~10\) ms vs. 16.9 ms by SPG.

<table>
<thead>
<tr>
<th>Detectors</th>
<th>Inference time (ms)</th>
<th>Inference time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CenterPoint-Pillar</td>
<td>42.7</td>
<td>140.0</td>
</tr>
<tr>
<td>CenterPoint-Pillar+S2D</td>
<td>53.1 (+10.4)</td>
<td>156.9 (+16.9)</td>
</tr>
<tr>
<td>CenterPoint-Voxel</td>
<td>53.0</td>
<td></td>
</tr>
<tr>
<td>CenterPoint-Voxel+S2D</td>
<td>62.8 (+9.8)</td>
<td></td>
</tr>
<tr>
<td>PV-RCNN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PV-RCNN+SPG</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5 Discussion and Conclusion

This paper presents the novel Sparse2Dense framework that learns to densify 3D features to boost 3D object detection performance. Our key idea is to learn to transfer dense point knowledge from the trained dense point 3D detector (DDet) to the sparse point 3D detector (SDet), such that SDet can learn to densify 3D features in the latent space. With the trained SDet, we only need the core component of SDet to detect 3D objects in regular point clouds, so we can enhance the detection accuracy without degrading the speed. Further, to enhance the transfer of dense point knowledge, we design the S2D module and the point cloud reconstruction module in SDet to enhance the sparse features. Last, we adopt our framework to various 3D detectors, showing that their performance can all be improved consistently on multiple benchmark datasets. In the future, we will apply our framework to more point cloud applications that require dense features, such as 3D segmentation and object tracking, to boost their performance while maintaining high computational efficiency.

Limitations. First, objects far from the LiDAR sensor in the point cloud sequence contain only a few points. It is still difficult to generate dense features for these objects with our DDet. Second, the training time of our framework is longer than traditional training, as we need multi-stage training to pre-train DDet and then transfer knowledge from the pre-trained DDet to SDet. Third, our models trained on Waymo Open Dataset need inputs containing specific point features like intensity and elongation, which limits our models evaluating on different datasets, like KITTI. We will explore removing the specific point features to make the model more general in future works.

Societal Impacts. Our proposed framework can provide better 3D object detection performance for autonomous vehicles. However, like most existing 3D detectors, it may produce errors in some edge cases, due to the limited data, so further research is still needed to improve its robustness.

Acknowledgements. Thank all the co-authors, reviewers, and ACs for their remarkable efforts. This work was supported by the project #MMT-p2-21 of the Shun Hing Institute of Advanced Engineering, The Chinese University of Hong Kong, and the Shanghai Committee of Science and Technology (Grant No.21DZ1100100).

References


[41] Xu Yan, Jiatao Gao, Jie Li, Ruimao Zhang, Zhen Li, Rui Huang, and Shuguan Cui. Sparse single sweep lidar point cloud segmentation via learning contextual shape priors from scene completion. In AAAI, 2021.


