SparseDet: Towards End-to-End 3D Object Detection

Jianhong Han¹, Zhaoyi Wan², Zhe Liu³, Jie Feng¹ and Bingfeng Zhou¹

¹Wangxuan Institute of Computer Technology, Peking University, Beijing, China
²University of Rochester, Rochester, U.S.A.
³Huazhong University of Science and Technology, Wuhan, China

{hanjh, fengjie, cczbfr}@pku.edu.cn, i@wanzy.me, zheliu1994@hust.edu.cn

Keywords: 3D Deep Learning, Object Detection, Point Clouds, Scene and Object Understanding

Abstract: In this paper, we propose SparseDet for end-to-end 3D object detection from point cloud. Existing works on 3D object detection rely on dense object candidates over all locations in a 3D or 2D grid following the mainstream methods for object detection in 2D images. However, this dense paradigm requires expertise in data to fulfill the gap between label and detection. As a new detection paradigm, SparseDet maintains a fixed set of learnable proposals to represent latent candidates and directly perform classification and localization for 3D objects through stacked transformers. It demonstrates that effective 3D object detection can be achieved with none of post-processing such as redundant removal and non-maximum suppression. With a properly designed network, SparseDet achieves highly competitive detection accuracy while running with a more efficient speed of 34.5 FPS. We believe this end-to-end paradigm of SparseDet will inspire new thinking on the sparsity of 3D object detection.

1 INTRODUCTION

3D Object detection is a technology used to identify the category and location of an object of interest in a scene. As one of the fundamental topics in the field of computer vision, it plays a pivotal role in applications for a wide range of scenarios, e.g., autonomous driving and augmented reality. With the advancement of 3D sensors, considerable research attention has been paid to 3D object detection from point cloud in recent years. Due to its representation power from a low-resolution resampling of the real 3D world geometry, point cloud scanned from LiDAR depth sensors achieves remarkable success in this field. Meanwhile, the sparsity and irregularity of point cloud data pose great challenges to accurate and robust 3D object detection.

For the task of 3D object detection from point cloud, we aim at representing an object with an oriented 3D bounding box around it given the point cloud of a scene as input. To this end, researchers in 3D object detection have proposed various effective methods. Mainstream 3D object detectors perform dense regression from a large set of coarse proposals or anchors in a rigid spatial grid. Despite the success that this paradigm has achieved, it is limited in several aspects. (1) The dense prediction requires and is sensitive to post-processing procedures, e.g., non-maximum suppression (NMS), to form the final detection. (2) Hyper-parameters such as sizes and aspect ratios of anchors must be carefully tuned for effective detection. (3) The gap between intermediate representation and final detection makes empirical label assignment strategies critical to detectors.

Drawing inspiration from object detection in 2D images, we explore end-to-end 3D object detection in a sparse manner in this paper. To fit into the nature of sparsity in point cloud, we propose to apply sparse prediction for detection, thus making an end-to-end paradigm for 3D object detection. Although end-to-end detection is proven feasible in 2D images [Carion et al., 2020], an effective design of sparse detector for 3D object in point cloud is non-trivial due to the challenges and differences in tasks. As shown in Tab. 1:

Table 1: Directly applying sparse prediction in a 3D object detection algorithm VoxelNet (Zhou and Tuzel, 2018) suffers from significant degradation of performance. SparseDet makes the first sparse 3D object detector and demonstrates effectiveness and efficiency.

| Method          | Easy | Mod. | Hard | mAP  |
|-----------------|------|------|------|------|
| VoxelNet        | 81.98| 65.46| 62.85| 70.10|
| sparse VoxelNet | 61.17| 56.73| 56.61| 58.17|
| SparseDet       | 88.36| 78.45| 77.03| 81.28|
**SparseDet**

- **Voxelization**
- **3D Feature Extraction**
- **Map to BEV**
- **Feature Alignment**
- **Dense Head**
- **Post Processing**

Figure 1: Comparison with mainstream 3D object detection paradigms. Distinguishable from conventional dense and dense-to-sparse approaches, SparseDet performs end-to-end detection without post-processing.

**Dense-to-Sparse**

- **Voxelization**
- **3D Feature Extraction**
- **Map to BEV**
- **Feature Alignment**
- **Sparse Head**
- **Post Processing**

According to the manner of generating boxes, the existing methods for 3D object detection are mainly divided into two categories, including the dense and dense-to-sparse methods. As shown in Fig. [1](#).

**Dense 3D Object Detection.** The early dense 3D detectors mainly adopted the anchor-based design, which requires manually setting anchors for each category object. A typical representative method is VoxelNet (Zhou and Tuzel, 2018), which first extracts the voxel features by a Voxel Feature Encoding layer. Then, a region proposal network (RPN) takes the voxel feature as input and generates the dense 3D boxes on the detection head. SECOND (Yan et al., 2018) utilizes a more efficient sparse convolution operation to learn voxel feature representation for improving VoxelNet. Based on SECOND, Pointpillars (Lang et al., 2019) divides the point cloud as a special voxel called pillars to further improve the running speed. However, the pillar-based manner inevitably loses some important context and spatial information. Towards this goal, TANet (Liu et al., 2020) proposes a TA module to obtain more robust and distinguishable features. Recently, anchor-free 3D detector CenterPoint (Yin et al., 2021) regards objects as points and discards the hand-crafted anchor boxes. Moreover, all these methods need NMS post-processing operation through setting a suitable score threshold to filter out the redundant boxes in inference. Which is a common disadvantage of the dense 3D detection paradigm.

**Dense-to-Sparse 3D Object Detection.** F-PointNet (Qi et al., 2018), Frustum-ConvNet (Wang and Jia, 2019) and SIFRNet (Zhao et al., 2019) first generate a set of 2D region proposals on a front view image with the help of offline strong 2D detectors (Liu et al., 2016; Redmon and Farhadi, 2018).
Then, each 2D proposal is converted into a 3D viewing frustum in point cloud. Finally, PointNets (Qi et al., 2017a, 2017b) is applied to estimate the final 3D boxes for each 3D viewing frustum. However, the manner is limited by the performance of 2D detectors. Instead of F-PointNet (Qi et al., 2018), Shi et al., 2019; Yang et al., 2019b, Shi et al., 2020; Chen et al., 2019; Huang et al., 2020; Chen et al., 2017; Ku et al., 2018; Deng et al., 2020) directly achieve the online dense-to-sparse paradigm. Specifically, these methods first produce a large number of boxes via a RPN, then to provide the high-quality boxes for the RCNN stage, the NMS is also a necessary operation during the process from the first stage to the second stage. Finally, the RCNN further estimates the final 3D boxes via refining the sparse boxes from the RPN stage. These approaches can usually obtain high performance but a slower running speed compared with the dense 3D object detectors. In contrast, our SparseDet not only abandons some tedious manual operation (e.g., anchors and NMS), but also can achieve comparable results with a high running speed.

**Sparse 2D Object Detection.** Recently, with the widespread application of transformers in computer vision, sparse object detection methods have attracted more and more attention. DETR (Carion et al., 2020) builds the first fully end-to-end 2D object detector, which not only eliminates the anchor design and NMS operation but also achieves comparable results with the existing detectors. Based on DETR, D-DETR (Zhu et al., 2020) mitigates the slow convergence and high complexity issues of DETR through the proposed deformable attention module. Sparse R-CNN (Sun et al., 2020) completely remove to object candidates design and achieve better performance and faster convergence through the learnable proposal boxes and their corresponding features than DETR and D-DETR. Motivated by these sparse methods, we propose SparseDet in this paper, which aims to provide a strong and simple baseline serving for the 3D object detection community.

## 3 SPARSEDET FOR 3D OBJECT DETECTION

In this section, we describe the novel SparseDet to explore the feasibility of a sparse detector from 3D point cloud. The overall architecture is illustrated in Fig. 1. The paradigm of SparseDet can be roughly divided into three main components: a voxel-based feature extractor, a feature alignment module that refines the extracted features, and a detection head performing detection in a sparse manner. In the following, we introduce each of the components of SparseDet and the loss functions for the objective of training in detail.

### 3.1 Voxel Feature Extraction

Compared with two-stage point-based methods (Shi et al., 2019; Huang et al., 2020), voxel-based approaches (Lang et al., 2019; Yan et al., 2018) occupy a higher efficiency. Therefore, our method focus on voxel-based methods. Specifically, We first encode the point cloud into a regular volume grid, namely voxels, thus effectively compressing the representation with aggregated features. However, point cloud is of uneven distribution since it is collected by LiDAR sensors distant from targets. Therefore, the number of points to be represented in each voxel varies dramatically. To avoid being overwhelmed by voxels with overly dense points, we limit the maximum number of points as \( T \) in each voxel. The neutralization is implemented by random sampling inside these voxels.

More formally, given a point cloud in the range of \( L \times W \times H \) meters and voxel size of \( D_x, D_y, D_z \), the discrete voxel feature grid with the shape of \( S_x \times S_y \times S_z \) is obtained, where

\[
S_x = \lceil \frac{L}{D_x} \rceil, S_y = \lceil \frac{W}{D_y} \rceil, S_z = \lceil \frac{H}{D_z} \rceil.  \tag{1}
\]

After voxelization, non-empty voxels are maintained and encoded as the mean of point-wise features of all inside points. Then the voxel features are fed into a stack of 3D sparse convolution layers (Graham, 2014; Graham, 2015; Graham and van der Maaten, 2017; Graham et al., 2018; Yan et al., 2018) to extract rich features from the voxel representation.

### 3.2 Feature Alignment

We avoid directly exploiting the 3D voxel features for prediction, which bring high computational cost. Instead, the features in 3D are collapsed into a fixed view, e.g., Birds’ Eye View (BEV), to align features along the Z-axis. The conventional BEV feature alignment involves a downsample-and-upsample schema: a sub-network that reduces the resolution with stacked 2D convolutions and a multi-scale feature fusion sub-network.

Therefore, we introduce the Feature Pyramid Network (FPN) (Lin et al., 2017a) and Pyramid Sampling Aggregation (PSA) (Liu et al., 2020) modules as the basis for our feature alignment. FPN is designed to solve the problem of scale variance of which objects emerge in images. Although the scale of akin
objects is consistent in point cloud due to the differences in sensors, point sparsity and reflection strength decrease for objects far from the sensor. Therefore, sparse 3D object detection benefits from a pyramid model structure to capture information from different scales. Given the BEV features where the view is fixed, we obtain features with different solutions. Specifically, 2x, 4x, and 8x downsampled features in the top-down order are produced by three 4-layers convolution modules, whose number of filters is 64, 128, and 256, respectively. The outputs of these blocks are denoted as $B_1$, $B_2$ and $B_3$.

Then, three feature pyramids of each $B$-feature are achieved through down-sampling and up-sampling. Specifically, $B_1$ obtains its feature pyramids with itself and 2x and 4x down-sampled features. $B_2$ obtains its feature pyramids with its 2x up-sampled feature, itself, and its 2x down-sampled feature. Similarly, $B_3$ obtains its feature pyramids with two up-sampling operations based on itself. Now, each pyramid will have the same shape features. Laterally concatenate the same size features of each pyramid to get cross-layer features. Followed by a fusion block convolution layer, the new feature pyramid with multi-level information is fed into a FPN-like structure inversely. Namely, the down-top interpolated feature is added to the last hierarchy lateral feature. Finally, we obtain an informative feature list, which will be delivered to the sparse detection head.

### 3.3 Sparse 3D Detection Head

Recently, the attention mechanism has achieved good results in object detection in 2D images. DETR (Carion et al., 2020) uses a sparse set of object queries, to interact with the global image feature. Features of each position in the image can obtain information of other positions, predictions without any hand-crafted components are realized. Subsequently, DETR (Carion et al., 2020) is improved by Deformable-DETR (Zhu et al., 2020) through reducing the range of feature search to facilitate the convergence process. Sparse R-CNN (Sun et al., 2020) uses a small number of boxes with features as the learning region proposal. Via the self-attention of these proposal features and their interaction with RoI features, it achieves pleasing object feature learning.

The model design of our sparse detection head is illustrated in Fig. 2. Inspired by Sparse R-CNN (Sun et al., 2020), we use a small set of learnable bounding boxes with learnable features as object candidates. These sparse candidates are used to extract the feature of Region of Interest (RoI), which avoids hun-

![Figure 2: Model design of the sparse detection head, where $K$ represents the index number of stages. The learnable proposals, identified by 3D boxes and feature vectors, interact with the stacked dynamic heads to be refined. Note the proposals produced by the last stage of detection head directly perform as the final detection. Dynamic heads and prediction layers are not shared by proposals.](image-url)
When calculating loss, there are two stages. One is the matching stage between the fixed-size set of $N$ predictions and $M$ ground truth objects, where $N$ is set to be significantly larger than the typical number of objects in a point cloud. And the second stage is the optimization of the matched $M$ predictions and ground truth pairs. Unlike the predecessors, our sparse method does not have the concept of anchor, so all losses are directly compared in the prediction results and ground truth.

The $N-M$ matching aims to filter out $M$ competitive candidates from all the $N$ predictions, we follow the bipartite matching loss approach (Carion et al., 2020; Zhu et al., 2020; Sun et al., 2020; Yang et al., 2019b; Stewart et al., 2016) based on the Hungarian algorithm (Kuhn, 1955) to compute the pairwise matching cost. Let us denote by $y = \{y_i\}_{i=1}^M$ the ground truth set of objects, and $\hat{y} = \{\hat{y}_i\}_{i=1}^N$ the set of $N$ predictions. Where $N>M$. The matching cost is defined as follows:

$$ C = \arg \min_{i \in [M]} L_{\text{match}}(\hat{y}_i, y_j)$$(3)

In which the pair-wise loss is computed as:

$$L_{\text{match}} = \lambda_{\text{cls}} \cdot L_{\text{cls}} + \lambda_{\text{d1}} \cdot L_{\text{d1}} + \lambda_{\text{IoU}} \cdot L_{\text{BEV, IoU}}$$

We find that the angle has a great influence on matching during the research. A proper angle matching method plays a big role. Here Sine-Error loss is adopted following (Yan et al., 2018) to solve the adversarial example problem between orientations of 0 and $\pi$.

In terms of $\text{IoU}$ loss, it is used to constrain the overlapping degree between boxes. For 3D oriented box matching, it is natural to think of 3D rotatedIoU between boxes. It is difficult to optimize all
dimensions together, which makes the detector confused. So we design a two-stage IoU matching style to make it easier to learn the relationship between boxes.

At the matching stage, Axis-Aligned (AA) BEV 2D bounding box is used to compute the IoU loss of prediction and ground truth box, denoted as $L_{\text{AA,BEV,IoU}}$. Axis-Aligned means turning the box to X-axis or Y-axis according to the origin orientation angle $\theta$. BEV means the position and scale of the bounding box on the Z-axis (height axis) are not considered. The purpose of these operations is to reduce the difficulty of matching orientation and height with other parameters at the same time, making it easier to catch the rough shape. More accurate box matching will be executed at the optimization stage of matched pairs.

The training loss is only performed on matched pairs. Which is almost the same as matching cost, except the IoU loss adopts Rotated 3D DIoU loss denoted as $L_{\text{R,3D,DIoU}}$. The Rotated 3D DIoU loss (Zheng et al., 2020) encode the orientation and height information to make the results more precise:

$$L = \lambda_{\text{cls}} \cdot L_{\text{cls}} + \lambda_{\text{L1}} \cdot L_{\text{L1}} + \lambda_{\text{IoU}} \cdot L_{\text{R,3D,DIoU}}$$

The final loss is the sum of all matched pairs normalized by the number of ground truth objects inside the batch.

4 EXPERIMENTS

In this section, we conduct extensive experiments to demonstrate the effectiveness of SparseDet and reveal its design concepts. All the experiments are conducted upon the popular KITTI (Geiger et al., 2012) dataset. KITTI is a 3D object detection dataset for autonomous driving in the wild with 7481 training samples. Following F-PointNet (Qi et al., 2018), we divide the training samples into a training set consisting of 3717 samples and a validation set that contains 3769 samples. We evaluate models on the Car category with the official evaluation protocol where the IoU threshold is set as 0.7. Mean average precision (mAP) with 11 recall positions under three difficulty levels (Easy, Moderate and Hard) is reported for a fair comparison with previous methods.

4.1 Implementation Details

Network Architecture. The raw point cloud range is clipped into [0,70.4] m, [-40,40] m, and [-3,1] m for the XYZ axes according to the FOV annotation of KITTI Dataset. Then we adopt a voxel size of (0.05m, 0.05m, 0.1m) to discrete the whole point cloud into regular grids. Standard data-augmentation for point clouds is performed following the mainstream strategies in (Yan et al., 2018; Lang et al., 2018; Shi et al., 2019; Shi et al., 2020), etc. Specifically, it includes randomly sampling ground truth objects from the database to join the current point cloud, randomly
Table 2: Evaluation comparison with state-of-the-art on KITTI validation set. As a detector in the family of voxel-based modality, SparseDet achieves clear and consistent improvements over akin methods. Theoretically, point-based methods can be augmented with the sparse detection paradigm of SparseDet, although we choose voxel-based methods as our baseline in consideration of inference efficiency. Data of other methods come from their corresponding paper.

| Method       | Modality        | AP$_3D$ (IoU = 0.7) | AP$_{BEV}$ (IoU = 0.7) | FPS | Param. | FLOPs |
|--------------|-----------------|---------------------|------------------------|-----|--------|-------|
| MV3D         | RGB & LiDAR     | 62.68               | 78.10                  | 2.78| -      | -     |
| ContFuse     | RGB & LiDAR     | 73.25               | 87.34                  | 16.67| -      | -     |
| AVOD-FPN     | RGB & LiDAR     | 74.44               | 84.02                  | 5.88| -      | -     |
| F-PointNet   | RGB & LiDAR     | 70.92               | 84.02                  | 5.88| -      | -     |
| VoxelNet     | Voxel-based     | 65.46               | 84.81                  | -   | -      | -     |
| SECOND       | Voxel-based     | 76.48               | 87.07                  | 20.00| 5.33M  | 76.70G|
| TANet        | Voxel-based     | 77.85               | 87.07                  | 20.00| 5.33M  | 76.70G|
| PointRCNN    | Point-based     | 78.63               | 87.89                  | 8.33| 4.04M  | 27.38G|
| Part-A$^2$-f| Point-based     | 79.96               | 88.05                  | 12.50| 59.23M | -     |
| Part-A$^2$-a| Point-based     | 79.47               | 88.61                  | 12.50| 59.23M | -     |
| SparseDet    | Voxel-based     | 78.45               | 88.02                  | 34.48| 25.10M | 37.98G|

Table 3: Evaluation results on KITTI Car detection under different IoU thresholds. In combination with sparse prediction, SparseDet demonstrates significant advantages in accurate localization. Data of other methods come from (Zhou et al., 2019).

| Metric        | Method               | IoU = 0.7 | IoU = 0.75 | IoU = 0.8 |
|---------------|----------------------|-----------|------------|-----------|
|               |                      | Easy Mod. | Easy Mod.  | Easy Mod. |
|               |                      | Hard      | Hard       | Hard      |
| AP$_3D$       | PointPillars(Lang et al., 2019) | 87.29     | 72.39      | 47.23     |
|               | +3D IoU Loss(Zhou et al., 2019) | 76.69     | 70.47      | 36.31     |
|               | +3D GIoU Loss(Zhou et al., 2019) | 88.43     | 77.58      | 52.56     |
|               | Ours                 | 78.45     | 77.03      | 47.22     |
| AP$_{BEV}$    | PointRCNN(Shi et al., 2019) | 88.14     | 73.27      | 44.21     |
|               | +3D IoU Loss(Zhou et al., 2019) | 77.80     | 63.54      | 38.88     |
|               | +3D GIoU Loss(Zhou et al., 2019) | 88.84     | 77.47      | 45.38     |
|               | Ours                 | 88.62     | 84.02      | 46.50     |

The 3D sparse convolution backbone does one convolution block of its own resolution and three consecutive 2x down-sampling convolutions to obtain an 8x down-sampled feature map with filter numbers of (16, 32, 48, 64) respectively. After the feature alignment process, three convolution block with 4 layers in each is utilized to obtain a feature list consisting of its origin, half and quarter resolution of the input feature. The dimensions of these blocks are (64, 128, 256). Then the feature list is fed into the PSA-FPN feature extractor, transformed as a new same resolution feature list with the dimension of (128, 128, 128). As for the detection head, the proposal boxes number $N$ is set as 100, and the dimension of the proposal features is 128. A single head is stacked 6 times. Layers of the heads are initialized with Xavier (Glorot and Bengio, 2010). The pooling resolution S is 7. We set $\lambda_{cls}$, $\lambda_{cls}$, and $\lambda_{IoU}$ to 2.0, 5.0, 2.0 respectively following (Carion et al., 2020; Sun et al., 2020).

Training details. The whole framework of SparseDet is optimized with the Adam optimizer in an end-to-end manner. All models are trained with 8 GPUs for 320 epochs with batch size 48, which takes around 6 hours. The learning rate is initialized as 0.003 and updated by a cosine annealing strategy for the learning rate decay.

Inference details. At the inference process, the proposed boxes processed by the stacked heads will be directly used as the prediction result without any post-processing. The score of classification results shows the confidence of the prediction.

4.2 Comparison with SOTA

Evaluation of 3D object detection on KITTI.
Robustness. This method is more robust to noisy input data because its accuracy has less relevance to the input data variation due to its sparse property. Following TANet (Liu et al., 2020), we introduce noise points to each object to verify robustness. As shown in Tab. 6, we add three different numbers of noise points to the objects. The data of PointRCNN (Shi et al., 2019), PointPillars (Lang et al., 2018), and TANet (Liu et al., 2020) comes from TANet (Liu et al., 2020) paper. While the data of SECOND (Yan et al., 2018) is reproduced by us using OpenPCDet (Team, 2020).

After the noise points are added, the accuracy of every method will decrease inevitably. Leaving aside the specific accuracy number, we only observe the declining trend of accuracy. In terms of hard category and overall mAP, SECOND (Yan et al., 2018) is better than SparseDet at the beginning, but with the increase of noise points, its accuracy is gradually surpassed by SparseDet. The results show that the average decline rates of mAP of the methods under discussion are 1.36%, 1.08%, 0.51%, 0.71%, and 0.48% respectively. The decline speed of our SparseDet is significantly slower when the number of noise points grows.

Sparsity. The performance of a detector will be greatly affected by the hyper-parameters under a single matching IoU threshold (Tychsen-Smith and Persson, 2018). So we utilize three different matching thresholds 0.70, 0.75, 0.80 for evaluation following (Zhou et al., 2019). We compare the evaluation results with PointPillars (Lang et al., 2018) family on the 3D benchmark, and with PointRCNN (Shi et al., 2019) family on the BEV benchmark. It is proved that our method without any hand-crafted prior is solid. At the same time, BEV results are better than 3D results, which shows that the Z-axis dimension in 3D detection is difficult to match.

The performance of a detector will be greatly affected by the hyper-parameters under a single matching IoU threshold (Tychsen-Smith and Persson, 2018). So we utilize three different matching thresholds 0.70, 0.75, 0.80 for evaluation following (Zhou et al., 2019). We compare the evaluation results with PointPillars (Lang et al., 2018) family on the 3D benchmark, and with PointRCNN (Shi et al., 2019) family on the BEV benchmark. It is proved that our method without any hand-crafted prior is solid. At the same time, BEV results are better than 3D results, which shows that the Z-axis dimension in 3D detection is difficult to match.

The performance of a detector will be greatly affected by the hyper-parameters under a single matching IoU threshold (Tychsen-Smith and Persson, 2018). So we utilize three different matching thresholds 0.70, 0.75, 0.80 for evaluation following (Zhou et al., 2019). We compare the evaluation results with PointPillars (Lang et al., 2018) family on the 3D benchmark, and with PointRCNN (Shi et al., 2019) family on the BEV benchmark. It is proved that our method without any hand-crafted prior is solid. At the same time, BEV results are better than 3D results, which shows that the Z-axis dimension in 3D detection is difficult to match.

The performance of a detector will be greatly affected by the hyper-parameters under a single matching IoU threshold (Tychsen-Smith and Persson, 2018). So we utilize three different matching thresholds 0.70, 0.75, 0.80 for evaluation following (Zhou et al., 2019). We compare the evaluation results with PointPillars (Lang et al., 2018) family on the 3D benchmark, and with PointRCNN (Shi et al., 2019) family on the BEV benchmark. It is proved that our method without any hand-crafted prior is solid. At the same time, BEV results are better than 3D results, which shows that the Z-axis dimension in 3D detection is difficult to match.

The performance of a detector will be greatly affected by the hyper-parameters under a single matching IoU threshold (Tychsen-Smith and Persson, 2018). So we utilize three different matching thresholds 0.70, 0.75, 0.80 for evaluation following (Zhou et al., 2019). We compare the evaluation results with PointPillars (Lang et al., 2018) family on the 3D benchmark, and with PointRCNN (Shi et al., 2019) family on the BEV benchmark. It is proved that our method without any hand-crafted prior is solid. At the same time, BEV results are better than 3D results, which shows that the Z-axis dimension in 3D detection is difficult to match.

The performance of a detector will be greatly affected by the hyper-parameters under a single matching IoU threshold (Tychsen-Smith and Persson, 2018). So we utilize three different matching thresholds 0.70, 0.75, 0.80 for evaluation following (Zhou et al., 2019). We compare the evaluation results with PointPillars (Lang et al., 2018) family on the 3D benchmark, and with PointRCNN (Shi et al., 2019) family on the BEV benchmark. It is proved that our method without any hand-crafted prior is solid. At the same time, BEV results are better than 3D results, which shows that the Z-axis dimension in 3D detection is difficult to match.

The performance of a detector will be greatly affected by the hyper-parameters under a single matching IoU threshold (Tychsen-Smith and Persson, 2018). So we utilize three different matching thresholds 0.70, 0.75, 0.80 for evaluation following (Zhou et al., 2019). We compare the evaluation results with PointPillars (Lang et al., 2018) family on the 3D benchmark, and with PointRCNN (Shi et al., 2019) family on the BEV benchmark. It is proved that our method without any hand-crafted prior is solid. At the same time, BEV results are better than 3D results, which shows that the Z-axis dimension in 3D detection is difficult to match.

The performance of a detector will be greatly affected by the hyper-parameters under a single matching IoU threshold (Tychsen-Smith and Persson, 2018). So we utilize three different matching thresholds 0.70, 0.75, 0.80 for evaluation following (Zhou et al., 2019). We compare the evaluation results with PointPillars (Lang et al., 2018) family on the 3D benchmark, and with PointRCNN (Shi et al., 2019) family on the BEV benchmark. It is proved that our method without any hand-crafted prior is solid. At the same time, BEV results are better than 3D results, which shows that the Z-axis dimension in 3D detection is difficult to match.

The performance of a detector will be greatly affected by the hyper-parameters under a single matching IoU threshold (Tychsen-Smith and Persson, 2018). So we utilize three different matching thresholds 0.70, 0.75, 0.80 for evaluation following (Zhou et al., 2019). We compare the evaluation results with PointPillars (Lang et al., 2018) family on the 3D benchmark, and with PointRCNN (Shi et al., 2019) family on the BEV benchmark. It is proved that our method without any hand-crafted prior is solid. At the same time, BEV results are better than 3D results, which shows that the Z-axis dimension in 3D detection is difficult to match.
Figure 4: The proposal boxes behavior visualization. At first, the learned proposals are randomly distributed in the point cloud space. Then they match to GTs gradually with the stacked heads structure. The effectiveness of sparse detection is proved.

Table 7: Ablations on angle regression loss. Since the sparse detection is performed without guidance from anchors nor voxel correspondence, a proper loss is demanded to effectively supervise the angle regression.

| $L_\theta$         | Easy | Mod. | Hard | mAP | \(\Delta\) |
|--------------------|------|------|------|-----|----------|
| $L_1$              | 81.69| 72.86| 72.92| 75.82| -5.46    |
| $L_1(\sin, \cos)$ | 83.41| 74.49| 71.57| 76.49| -4.79    |

Table 8: Ablations on 3D box matching constraints for the training of sparse prediction. Cost stands for the matching cost, and Loss stands for the training loss of matched pairs. ~\(\theta\) means discarding the angle constraint and ~\(h\) means discarding the height constraint. See Sec. 3.4 and 4.3 for details.

| Cost     | Loss     | Mod. | Hard | mAP   | \(\Delta\) |
|----------|----------|------|------|-------|----------|
| 3D       | 3D       | 0    | 0    | 0     | -81.29   |
| (-h)     | (-h)     | 59.15| 58.03| 60.55 | -20.74   |
| (-\(\theta\)) | (-\(\theta\)) | 76.91| 75.55| 79.24 | -2.05    |
| (-\(\theta\),-h) | (-\(\theta\),-h) | 77.24| 75.55| 79.64 | -1.65    |
| (-\(\theta\),-h) | (-\(\theta\)) | 75.12| 74.77| 76.77 | -4.51    |
| (-\(\theta\),-h) | 3D       | 78.48| 77.03| 81.29 | -        |

Significantly lower than other methods. TANet (Liu et al., 2020), which is specially designed to solve the problem of robustness, has a close descending speed as ours because they add the point-wise, channel-wise, and voxel-wise attention mechanism in the voxel feature extraction. So that the information of the input point cloud can be more fully extracted. However, we do not perform such complex interactions, and the result is still better than TANet (Liu et al., 2020), which proves the pretty robustness of our method.

4.3 Ablation Studies

Feature alignment fashion and loss design for the sparse detection head are the keys of SparseDet. In this section, we explore how the key component influence the final performance with extensive ablation experiments. All the ablation experiments are performed on the car class of the KITTI 3D benchmark.

Importance of PSA-FPN module. The widely used feature aligner in predecessors is a simple module in a conv-deconv way (Yan et al., 2018; Lang et al., 2018), which is reproduced as a baseline. We remove the two parts of our novel feature aligner one by one as a comparative experiment to verify the effectiveness of the overall design. The baseline achieves a 3D mAP of 77.23%. With only PSA and only FPN structure, the performance is boosted to 78.01% and 78.88%, respectively. While the joint mechanism yields a 3D mAP of 81.28%, outperforming the baseline model by 5.24% and improving the accuracy of each category steadily. It is obvious that our feature aligner is more effective and robust. The detailed results of each difficulty category are shown in Tab. 5.

Loss ablations. There are three components of the overall loss: classification loss, $\ell_1$ bounding box distance loss, and IoU loss. We have executed extensive ablation experiments to study the importance of each component. First, we turn each component on and off to evaluate the overall effect of each part. Then we look into the specific part to study the effectiveness by conducting different alternatives. Specifically, the choice of $\theta$ loss and IoU loss of the matching cost and the loss are deliberated respectively.

On the overall effect study of each loss component, a model without the IoU loss and a model without the $L_1$ distance loss are trained respectively to compare with the complete form. The corresponding results are presented in Tab. 6. The model without $L_1$ loss can not converge at all, leading to a complete failure on this task. While the model without IoU loss is a little better with 13.13% mAP, but is still a failure. The surprisingly poor results of the two incomplete models show that each component is indispensable for 3D detection. And the $L_1$ loss is at a more dominant position. These results are quite different from the situation of 2D images. Whose $L_1$ loss and IoU loss are a complement of each other, which would not lead to the failure of the task, but only a slight de-
cline (Carion et al., 2020). This phenomenon reveals that the 3D detection task has more challenges than 2D. Moreover, previous 3D detectors almost have no IoU loss also show satisfying detection results. This phenomenon reveals that sparse 3D detection is more difficult than dense fashion.

Then, we study the details of $L_1$ loss and IoU loss respectively. In terms of $L_1$ loss, there are two parts including $(x, y, z, w, l, h)$ and $\theta$ of the regression results as Eq(5). We calculate $L_1$ distance directly with the first part because it is trivial without particularity. But as for the second part $\theta$, it has a great influence on the detection effect for its periodicity. As shown in Tab. 4, the most direct way is to subtract the angle directly, it shows degradation of 6.72% of our final Sin-Error choice. Another alternative to solve the periodicity is to convert the angle into a vector encoded by $(\sin, \cos)$ pair (Ku et al., 2018). But in this way, two angles corresponding to the same box may generate a large loss for that each radian has a unique code, e.g. the 0 and $\pi$ radians. A 5.89% degradation appeared using this loss. While the Sin-Error only uses the sin value of the angle difference, fixing the adversarial problem, which is more reasonable to constrain the angle regression.

In terms of IoU loss, there are two stages of the matching cost and the matched loss. As stated in Set.4, the direct use of rotated 3D IoU cost and loss would not work at all. While our coarse-to-fine way consists of 2D IoU loss without angle and 3D IoU loss with angle exploiting a pretty good solution. To illustrate the effectiveness of this design, we show the comparison results by using different constraints in Tab. 5. By alternatively discarding $\theta$, height, or the combination of them, we validate the success of our design. The results inspire us that we should relax the restrictions on some dimensions to achieve better results sometimes.

### 4.4 Visualization

**Qualitative Results.** Some qualitative results of our proposed SparseDet are shown in Fig[3] Our SparseDet only takes the point cloud as input and the image is just for better visualization. The score threshold is set as 0.2 when drawing the prediction boxes. We can find that the detector is fairly smart to discriminate the similar classes of Truck and Van from the predicted target Car class.

**The Proposal Boxes Behavior.** Fig[4] shows the behavior of learned proposal boxes. We project the 3D boxes onto the image for convenient visualization. In the beginning, the proposals are rather big boxes, including almost the whole point cloud in different orientations. After continuous optimization of the stacked heads, precise results are obtained. It demonstrates that acceptable results can be produced even at the first head, which shows the effectiveness of the sparse detection. Subsequent heads are used to refine the predictions gradually to generate the final results.

## 5 CONCLUSIONS

In this paper, we present an end-to-end paradigm for 3D object detection in point cloud based on sparse prediction, namely SparseDet. It maintains and iteratively refines a sparse set of proposals to directly produce final object detection without redundant removal and non-maximum suppression. To collaborate with the sparse prediction, we closely investigate key components for effective 3D object detection and properly design the network. The proposed method achieves competitive results in the KITTI dataset, with significant and consistent improvement over the baseline. Moreover, the presented SparseDet is more robust to noisy data in point cloud.

SparseDet demonstrates the potential of end-to-end 3D object detection with sparse prediction. We hope it will inspire more exploration into the sparsity of 3D object detection and open up a new opportunity in this field. As one of the exploration directions, we will next try to combine the sparse detection paradigm with the point-based backbone to further improve the accuracy of the prediction results.

## ACKNOWLEDGEMENTS

This work was supported by National Key Research and Development Program of China [grant number 2018YFB1403901] and National Natural Science Foundation of China (NSFC) [grant number 61872014].

## REFERENCES

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In ECCV, pages 213–229. Springer.

Chen, X., Ma, H., Wan, J., Li, B., and Xia, T. (2017). Multi-view 3d object detection network for autonomous driving. In CVPR, pages 1907–1915.

Chen, Y., Liu, S., Shen, X., and Jia, J. (2019). Fast point r-cnn. In ICCV, pages 9775–9784.

Deng, J., Shi, S., Li, P., Zhou, W., Zhang, Y., and
