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

Jianhong Han[1], Zhaoyi Wan[2], Zhe Liu[3], Jie Feng[1] and Bingfeng Zhou[1]
[1]*Wangxuan Institute of Computer Technology, Peking University, Beijing, China*
[2]*University of Rochester, Rochester, U.S.A.*
[3]*Huazhong University of Science and Technology, Wuhan, China*

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.

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

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 chal-

Han, J., Wan, Z., Liu, Z., Feng, J. and Zhou, B.
SparseDet: Towards End-to-End 3D Object Detection.
DOI: 10.5220/0010918000003124
In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2022) - Volume 4: VISAPP*, pages 781-792
ISBN: 978-989-758-555-5; ISSN: 2184-4321

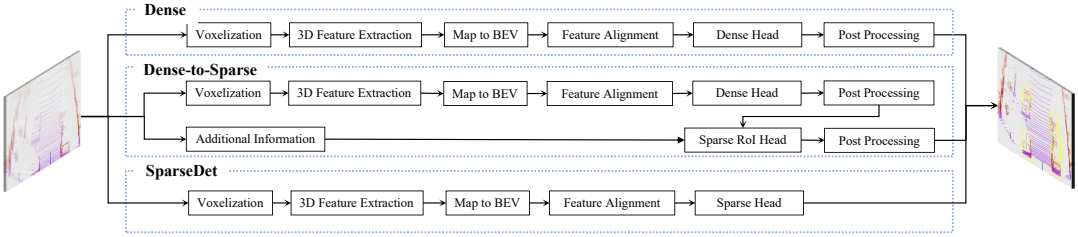

Figure 1: Comparison with mainstream 3D object detection paradigms. Distinguishable from conventional dense and dense-to-sparse approaches, SparseDet performs end-to-end detection without post-processing.

lenges and differences in tasks. As shown in Tab. 1, intuitively applying sparse prediction in 3D object detection thus suffers from significant degradation of detection accuracy. For the purpose of an effective end-to-end 3D object detector, we closely investigate the characteristics of detection in point cloud and accordingly design an algorithm that detects 3D objects through sparse prediction, namely *SparseDet*.

SparseDet bears several major differences from previous practices in 3D object detection. Firstly, it discards the enumeration through the voxel grid to generate dense proposals. Instead, it keeps the sparsity by maintaining a set of learnable proposals that are concentrated on the most possible regions in the point cloud. Secondly, it produces object detection without intermediate representation based on label assignment. Thus, requirements of any post-processing such as NMS are eliminated. Thirdly, it relies on none of pre-defined anchors but adaptively learns proposal initialization from a statistics perspective. We conduct extensive experiments to demonstrate the effectiveness of the proposed SparseDet. Specifically, state-of-the-art detection accuracy performance of 81.3% 3D mAP and efficiency performance of 34.5 FPS is achieved in KITTI. Moreover, exhaustive ablation studies are provided to reveal the insights of our model design. What we prefer more about the proposed sparse paradigm is it demonstrates promising robustness, which is crucial to the applicable value for a 3D object detector (see Sec. 4.2 for details).

Our contribution in this paper can be summarized as follows:

- We are the first to conduct an end-to-end 3D object detector in point cloud by applying a sparse prediction paradigm.
- In the context of sparse prediction, we closely investigate the concepts for model design, including a proper feature extractor and targeted loss functions, which provide guidance for effective practice of end-to-end 3D object detection.
- Extensive experiments validate the effectiveness of the proposed method: it is a trade-off between accuracy and efficiency, and demonstrates superiority in robustness.

2 RELATED WORK

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; Qi et al., 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

$$S_x = \lceil \frac{L}{D_x} \rceil, S_y = \lceil \frac{W}{D_y} \rceil, S_z = \lceil \frac{H}{D_z} \rceil. \quad (1)$$

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 hundreds of thousands of candidates from prior anchors or RPN. Specifically, we use N (e.g., 100) learnable bounding boxes. Each box is represented as the 7-D parameters $(c_x, c_y, c_z, h, w, l, \theta)$, including its center (c_x, c_y, c_z), size (w, l, h), and orientation θ that indicates the heading angle around the up-axis. However, the 3D bounding box is merely a rough representation of the object, lacking abundant semantic and geometric information. Thus the corresponding learnable proposal features of each box are proposed. Each such feature is represented as a slightly higher dimension latent vector to capture more characteristics of the object. The proposal boxes are initialized as the whole point cloud size with no rotation for a larger search space. The proposal features are randomly initialized.

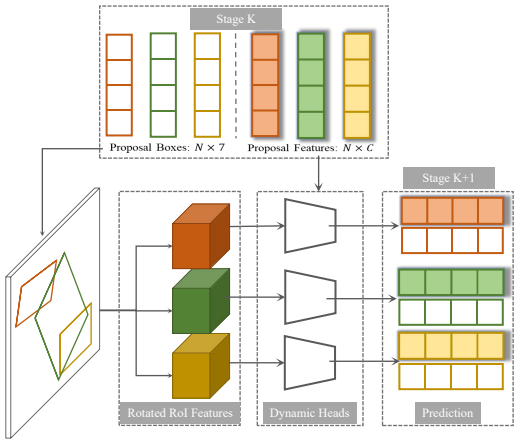

Figure 2: Model design of the sparse detection head, where K represents the index number of stages. The learnable proposals, identified by 3D boxes and feature vectors, interact with the stacked dynamic heads to be refined. Note the proposals produced by the last stage of detection head directly perform as the final detection. Dynamic heads and prediction layers are **not** shared by proposals.

The subsequent learning process will gradually narrow the scope of these boxes until they match the corresponding target. This initialization is effective and reasonable. The parameters of proposal boxes and proposal features will be optimized together with other parameters in the whole network. In fact, what these features eventually learn is the statistical characteristics of the objects that may appear throughout the data.

Then, given the proposal boxes, we can obtain the orientated BEV 2D bounding boxes by discarding the location and scale in the vertical direction. Each BEV box can extract its corresponding feature from the feature map by utilizing the Rotated RoIAlign (He et al., 2017) operation. So far, we have two types of features: the proposal features and the RoI features. For

N proposal boxes, there are N proposal features, and $N \times S \times S$ RoI features, in which S is the pooling resolution. First, a multi-head attention module followed by a LayerNorm layer is applied to the proposal features to reason about the relations between objects. Then, following Sparse R-CNN, each RoI feature will interact with the corresponding proposal feature to filter out ineffective bins through an exclusive Dynamic Instance Interactive Module. The Dynamic Instance Interactive Module is mainly used for making the RoI features and the proposal features communicate with each other to get more discriminate features. Specifically, it is achieved by consecutive 1×1 convolution with LayerNorm and ReLU activation function.

The output of the Dynamic Instance Interactive Module will serve as the final object features, which are used to compute the classification predictions and 7-D 3D bounding box regression predictions through two Multi Layer Perception (MLP) with ReLU activation branches. The regression branch outputs a vector $\Delta = (\Delta_x, \Delta_y, \Delta_z, \Delta_w, \Delta_l, \Delta_h, \Delta_\theta) \in \mathbb{R}^7$ that represents the residue from 3D proposal boxes to the ground truth boxes following predecessors (Zhou and Tuzel, 2018; Yan et al., 2018; Lang et al., 2018; Shi et al., 2019; Liu et al., 2020) :

$$\begin{aligned} &\Delta_x = \frac{x_g - x_p}{d_p}, \Delta_y = \frac{y_g - y_p}{d_p}, \Delta_z = \frac{z_g - z_p}{h_p} \\ &\Delta_w = \log\left(\frac{w_g}{w_p}\right), \Delta_l = \log\left(\frac{l_g}{l_p}\right), \Delta_h = \log\left(\frac{h_g}{h_p}\right) \\ &\Delta_\theta = \theta_g - \theta_p \end{aligned} \tag{2}$$

, where $d_p = \sqrt{(w_p)^2 + (l_p)^2}$. Then the residue vector will be decoded to the last prediction box to compose the new prediction.

The proposed Sparse head is stacked several times to perform an iterative structure. The object boxes prediction and object features of the previous stage are fed into the next stage to serve as the proposal boxes and proposal features. Each stacked head is supervised to optimize the proposal boxes and features at different stages.

3.4 Loss Function

Current 3D object detectors usually produce thousands of candidates, which can lead to a large number of near duplications. So the NMS post-processing is applied to screen out the fairly good results which exceed a certain score threshold or IoU threshold. There is no method to predict the ultimate objects directly in 3D object detection. While SparseDet is post-processing-free for it aims to predict the final result from the beginning.

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., 2019a; Stewart et al., 2016) based on the Hungarian algorithm (Kuhn, 1955) to compute the pair-wise 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:

$$\mathcal{C} = \underset{i \in M, j \in N}{\arg\min} \mathcal{L}_{\text{match}}(\hat{y}_i, y_j) \tag{3}$$

, in which the pair-wise loss is computed as:

$$\mathcal{L}_{\text{match}} = \lambda_{cls} \cdot \mathcal{L}_{cls} + \lambda_{L1} \cdot \mathcal{L}_{L1} + \lambda_{IoU} \cdot \mathcal{L}_{AA\_BEV\_IoU} \tag{4}$$

λ_{cls}, λ_{L1} and λ_{IoU} are coefficients of each component. $\mathcal{L}_{cls}$ is focal loss(Lin et al., 2017b) of predicted classifications and ground truth category labels.

As for regression loss, we find it is crucial in 3D sparse detection. It almost determines the performance of the detector. For details, please refer to our ablation in Sec.4.3. While general 3D detectors only use $L1$ loss to constrain the regression, we find it does not work in 3D sparse detection. The reason may be that sparse proposal is difficult to find the corresponding target merely relying on $L1$ loss. Thus we adopt the union of $L1$ loss and IoU loss following(Carion et al., 2020; Zhu et al., 2020; Sun et al., 2020). $\mathcal{L}_{L1}$ is $L1$ loss between the normalized predicted box and ground truth box, which has two parts as following:

$$\mathcal{L}_{L1} = \mathcal{L}_{reg\_\theta} + \mathcal{L}_{reg\_other} \tag{5}$$

, $\mathcal{L}_{reg_other}$is the regression loss for 3D center location and box size. $\mathcal{L}_{reg\_\theta}$ is the regression loss of θ. 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 π.

In terms of IoU loss, it is used to constrain the overlapping degree between boxes. For 3D oriented box matching, it is natural to think of 3D rotated IoU between boxes. But it is difficult to optimize all

Figure 3: Qualitative results on the KITTI dataset. For each sample, the upper part is the image for just visualization and the lower part is the point cloud corresponding to the image perspective. The detected objects are shown with yellow 3D bounding boxes.

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 $\mathcal{L}_{AA\_BEV\_IoU}$. Axis-Aligned means turning the box to X-axis or Y-axis according to the origin orientation angle θ. 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 $\mathcal{L}_{R\_3D\_DIoU}$. The Rotated 3D *DIoU* loss (Zheng et al., 2020) encode the orientation and height information to make the results more precise:

$$\mathcal{L} = \lambda_{cls} \cdot \mathcal{L}_{cls} + \lambda_{L1} \cdot \mathcal{L}_{L1} + \lambda_{IoU} \cdot \mathcal{L}_{R\_3D\_DIoU} \quad (6)$$

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

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 comes from their corresponding paper.

Method	Modality	AP_{3D} $(IoU = 0.7)$			AP_{BEV} $(IoU = 0.7)$			FPS	Param.	FLOPs
		Mod.	Hard	mAP	Mod.	Hard	mAP			
MV3D	RGB & LiDAR	62.68	56.56	63.51	78.10	76.67	80.44	2.78	-	-
ContFuse	RGB & LiDAR	73.25	67.81	75.79	87.34	82.43	88.40	16.67	-	-
AVOD-FPN	RGB & LiDAR	74.44	68.65	75.83	-	-	-	10.00	-	-
F-PointNet	RGB & LiDAR	70.92	63.65	72.78	84.02	76.44	82.87	5.88	-	-
VoxelNet	Voxel-based	65.46	62.85	70.07	84.81	78.57	84.33	4.35	-	-
SECOND	Voxel-based	76.48	69.10	77.67	87.07	79.66	85.56	20.00	5.33M	76.70G
TANet	Voxel-based	77.85	75.62	80.56	-	-	-	28.78	-	-
PointRCNN	Point-based	78.63	77.38	81.63	87.89	85.51	87.87	8.33	4.04M	27.38G
Part-A^2-f	Point-based	78.96	78.36	81.93	88.05	85.85	88.04	12.50	59.23M	-
Part-A^2-a	Point-based	79.47	78.54	82.49	88.61	87.31	88.78	12.50	63.81M	-
SparseDet	Voxel-based	78.45	77.03	81.28	88.02	86.49	88.11	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.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
AP_{3D}	PointPillars(Lang et al., 2019)	87.29	76.99	70.84	72.39	62.73	56.40	47.23	40.89	36.31
	+*3D IoU Loss*(Zhou et al., 2019)	87.88	77.92	75.70	76.18	65.83	62.12	57.82	45.03	42.95
	+*3D GIoU Loss*(Zhou et al., 2019)	**88.43**	78.15	76.34	76.93	66.36	63.68	56.36	44.43	42.72
	Ours	88.36	**78.45**	**77.03**	**77.58**	**68.02**	**65.61**	**63.38**	**52.56**	**47.22**
AP_{BEV}	PointRCNN(Shi et al., 2019)	88.14	77.58	75.36	73.27	63.54	61.08	44.21	38.88	34.62
	+*3D IoU Loss*(Zhou et al., 2019)	88.83	78.80	78.18	77.42	67.83	66.85	58.22	49.09	45.38
	+*3D GIoU Loss*(Zhou et al., 2019)	88.84	78.85	78.15	77.47	67.98	67.18	59.80	51.25	46.50
	Ours	**89.62**	**88.02**	**86.49**	**88.62**	**84.09**	**78.83**	**78.41**	**75.03**	**70.83**

the database to join the current point cloud, randomly flipping points in a 3D bounding box along the X-axis, global rotation and scaling, etc. The maximum number of points T in each voxel is set as 5.

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 λ_{cls}, λ_{L1}, and λ_{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.

Table 4: Robustness evaluation using metrics from (Liu et al., 2020). SparseDet degrades slower with the number of noise points growing.

Method	+noise	AP_{3D} $(IoU = 0.7)$		
		Mod.	Hard	mAP
PointRCNN	0	77.73	76.67	80.89
PointPillars	0	77.01	74.77	79.76
TANet	0	77.85	75.62	80.56
SECOND	0	78.43	**77.13**	**81.43**
SparseDet	0	**78.45**	77.03	81.28
PointRCNN	20	76.95	74.73	79.97
PointPillars	20	76.74	74.54	79.50
TANet	20	77.68	75.31	80.39
SECOND	20	78.19	76.81	81.17
SparseDet	20	**78.52**	**76.93**	**81.33**
PointRCNN	100	75.98	69.34	77.63
PointPillars	100	76.06	68.91	77.20
TANet	100	76.64	73.86	79.34
SECOND	100	77.27	73.35	79.7
SparseDet	100	**77.69**	**74.44**	**80.12**

4.2 Comparison with SOTA

Evaluation of 3D Object Detection on KITTI. We compare our SparseDet with several state-of-the-art methods on the 3D detection benchmark and the bird's eye view detection benchmark in Tab. 2. We achieved 81.28% and 88.11% average precision in the two tasks respectively with the real-time processing frame rate (34.5 FPS).

In terms of accuracy, SparseDet outperforms previous multi-sensor methods with large margins by only using LiDAR point clouds for the 3D detection benchmark. For both two benchmarks, SparseDet outperforms previous voxel-based methods, whose backbone is more similar to ours. At the same time, it is almost on par with the point-based methods. Which directly take raw point clouds as their operation inputs instead of converting a point cloud into a regular discrete representation. So the point-based methods are generally more accurate than voxel-based methods due to their fine-grained processing unit.

In terms of efficiency, our method is almost the fastest due to its sparse and end-to-end nature. It has an acceptable number of parameters and significantly fewer FLOPs. PointRCNN (Shi et al., 2019) seems better in parameters and FLOPs, but the running time is relatively long due to their hard work of point-wise operations. By comparison, our SparseDet makes the best balance between effectiveness and efficiency among all the methods.

Table 5: Ablation studies on feature alignment. To cooperate with the sparse detection, the feature aligner is redevised.

Backbone	Easy	Mod.	Hard	mAP
Baseline	83.38	74.82	73.48	77.23
+PSA	84.29	75.80	73.93	78.01
+FPN	84.65	76.62	75.38	78.88
Ours	88.36	78.45	77.03	81.28

Table 6: Impact of terms on our loss function. The integration of both $L1$ and IoU loss into our formulation is crucial for the effectiveness of SparseDet. The details are given in Sec. 4.3

Class	IoU	ℓ_1	Easy	Mod.	Hard	mAP
✓	✓	×	0.00	0.00	0.00	0.00
✓	×	✓	11.63	13.50	14.25	13.13
✓	✓	✓	88.36	78.45	77.03	81.28

Solidness. The performance of a detector will be greatly affected by the hyper-parameters under a single matching IoU threshold (Tychsen-Smith and Petersson, 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 in Tab. 3. The results show that our method outperforms predecessors when the IoU threshold is high, especially in 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.

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. 4, 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 (OpenPCDet, 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

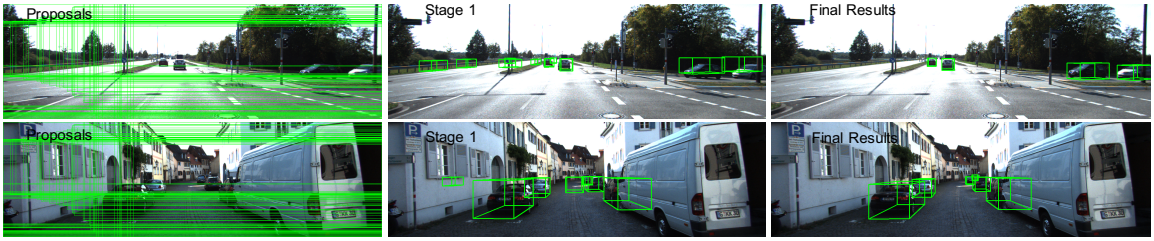

Figure 4: The proposal boxes behavior visualization. At first, the learned proposals are randomly distributed in the point cloud space. Then they match to GTs gradually with the stacked heads structure. The effectiveness of sparse detection is proved.

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.

$\mathcal{L}_\theta$	Easy	Mod.	Hard	mAP	Δ
$L1$	81.69	72.86	72.92	75.82	-5.46
$L1(sin, cos)$	83.41	74.49	71.57	76.49	-4.79
$Sin-Error$	88.36	78.46	77.03	81.28	-

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	Δ
3D	3D	0	0	0	-81.29
(-h)	(-h)	59.15	58.03	60.55	-20.74
(-θ)	(-θ)	76.91	75.55	79.24	-2.05
(-θ,-h)	(-θ,-h)	77.24	75.55	79.64	-1.65
(-θ,-h)	(-θ)	75.12	74.77	76.77	-4.51
(-θ,-h)	3D	78.48	77.03	81.29	-

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 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, ℓ_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 θ 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 $L1$ distance loss are trained respectively to compare with the complete form. The corresponding results are presented in Tab. 6. The model without $L1$ 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 $L1$ loss is at a more dominant position. These results are quite different from the situation of 2D images. Whose $L1$ loss and IoU loss are a complement of each other, which would not lead to the failure of the task, but only a slight decline (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 $L1$ loss and IoU loss respectively. In terms of $L1$ loss, there are two parts including (x, y, z, w, l, h) and θ of the regression results as Equ.5. We calculate $L1$ distance directly with the first part because it is trivial without particularity. But as for the second part θ, it has a great influence on the detection effect for its periodicity. As shown in Tab. 7, 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 π 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.3.4, 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. 8. By alternatively discarding θ, 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 Li, H. (2020). Voxel r-cnn: Towards high performance voxel-based 3d object detection. *arXiv preprint arXiv:2012.15712*.

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, pages 3354–3361. IEEE.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pages 249–256. JMLR Workshop and Conference Proceedings.

Graham, B. (2014). Spatially-sparse convolutional neural networks. *arXiv preprint arXiv:1409.6070*.

Graham, B. (2015). Sparse 3d convolutional neural networks. *arXiv preprint arXiv:1505.02890*.

Graham, B., Engelcke, M., and Van Der Maaten, L. (2018). 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, pages 9224–9232.

Graham, B. and van der Maaten, L. (2017). Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *ICCV*, pages 2961–2969.

Huang, T., Liu, Z., Chen, X., and Bai, X. (2020). Epnet: Enhancing point features with image semantics for 3d object detection. In *ECCV*, pages 35–52. Springer.

Ku, J., Mozifian, M., Lee, J., Harakeh, A., and Waslander, S. L. (2018). Joint 3d proposal generation and object detection from view aggregation. In *IROS*, pages 1–8. IEEE.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., and Beijbom, O. (2018). Pointpillars: Fast encoders for object detection from point clouds. *arXiv preprint arXiv:1812.05784*.

Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., and Beijbom, O. (2019). Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, pages 12697–12705.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017a). Feature pyramid networks for object detection. In *CVPR*, pages 2117–2125.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2017b). Focal loss for dense object detection. In *ICCV*, pages 2980–2988.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *ECCV*, pages 21–37. Springer.

Liu, Z., Zhao, X., Huang, T., Hu, R., Zhou, Y., and Bai, X. (2020). Tanet: Robust 3d object detection from point clouds with triple attention. In *AAAI*, pages 11677–11684.

OpenPCDet (2020). Openpcdet: An open-source toolbox for 3d object detection from point clouds. https://github.com/open-mmlab/OpenPCDet.

Qi, C. R., Liu, W., Wu, C., Su, H., and Guibas, L. J. (2018). Frustum pointnets for 3d object detection from rgb-d data. In *CVPR*, pages 918–927.

Qi, C. R., Su, H., Mo, K., and Guibas, L. J. (2017a). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 652–660.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, pages 5099–5108.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Shi, S., Wang, X., and Li, H. (2019). Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVRR*, pages 770–779.

Shi, S., Wang, Z., Shi, J., Wang, X., and Li, H. (2020). From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *TPAMI*.

Stewart, R., Andriluka, M., and Ng, A. Y. (2016). End-to-end people detection in crowded scenes. In *CVPR*, pages 2325–2333.

Sun, P., Zhang, R., Jiang, Y., Kong, T., Xu, C., Zhan, W., Tomizuka, M., Li, L., Yuan, Z., Wang, C., and Luo, P. (2020). SparseR-CNN: End-to-end object detection with learnable proposals. *arXiv preprint arXiv:2011.12450*.

Tychsen-Smith, L. and Petersson, L. (2018). Improving object localization with fitness nms and bounded iou loss. In *CVPR*.

Wang, Z. and Jia, K. (2019). Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. *arXiv preprint arXiv:1903.01864*.

Yan, Y., Mao, Y., and Li, B. (2018). Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337.

Yang, B., Wang, J., Clark, R., Hu, Q., Wang, S., Markham, A., and Trigoni, N. (2019a). Learning object bounding boxes for 3d instance segmentation on point clouds. *arXiv preprint arXiv:1906.01140*.

Yang, Z., Sun, Y., Liu, S., Shen, X., and Jia, J. (2019b). Std: Sparse-to-dense 3d object detector for point cloud. In *ICCV*, pages 1951–1960.

Yin, T., Zhou, X., and Krähenbühl, P. (2021). Center-based 3d object detection and tracking.

Zhao, X., Liu, Z., Hu, R., and Huang, K. (2019). 3d object detection using scale invariant and feature reweighting networks. In *AAAI*, volume 33, pages 9267–9274.

Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., and Ren, D. (2020). Distance-iou loss: Faster and better learning for bounding box regression. In *AAAI*, volume 34, pages 12993–13000.

Zhou, D., Fang, J., Song, X., Guan, C., and Yang, R. (2019). Iou loss for 2d/3d object detection. In *3DV*.

Zhou, Y. and Tuzel, O. (2018). Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, pages 4490–4499.

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., and Dai, J. (2020). Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*.