This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

FSHNet: Fully Sparse Hybrid Network for 3D Object Detection

Shuai Liu[1], Mingyue Cui[1], Boyang Li[1,2,*], Quanmin Liang[1], Tinghe Hong[1], Kai Huang[1]
Yunxiao Shan[3], Kai Huang[1]
[1]School of Computer Science and Engineering, Sun Yat-sen University
[2]Key Laboratory of Machine Intelligence and Advanced Computing, Sun Yat-sen University
[3]School of Artificial Intelligence, Sun Yat-sen University
{liush376@mail2, liby83@mail}.sysu.edu.cn

Abstract

Fully sparse 3D detectors have recently gained significant attention due to their efficiency in long-range detection. However, sparse 3D detectors extract features only from non-empty voxels, which impairs long-range interactions and causes the center feature missing. The former weakens the feature extraction capability, while the latter hinders network optimization. To address these challenges, we introduce the Fully Sparse Hybrid Network (FSHNet). FSHNet incorporates a proposed SlotFormer block to enhance the long-range feature extraction capability of existing sparse encoders. The SlotFormer divides sparse voxels using a slot partition approach, which, compared to traditional window partition, provides a larger receptive field. Additionally, we propose a dynamic sparse label assignment strategy to deeply optimize the network by providing more high-quality positive samples. To further enhance performance, we introduce a sparse upsampling module to refine downsampled voxels, preserving fine-grained details crucial for detecting small objects. Extensive experiments on the Waymo, nuScenes, and Argoverse2 benchmarks demonstrate the effectiveness of FSHNet. The code is available at `https://github.com/Say2L/FSHNet`.

1. Introduction

3D object detection has attracted increasing research interest with its growing range of applications in areas such as autonomous vehicle perception and precision robotic operations. Currently, 3D LiDAR-based object detectors [21, 40, 42] typically transform sparse features into 2D dense feature maps to leverage established 2D object detection techniques. However, these detectors often waste significant computational and memory resources processing unoccupied voxels. This issue becomes more pronounced when performing long-range detection [10]. Therefore, developing fully sparse 3D detectors (abbreviated as sparse detectors) that extract sparse features directly from sparse inputs without dense operations is urgently needed.

Sparse detectors are still emerging compared to their dense counterparts. Some methods [31, 43, 48] utilize the PointNet series [27, 28] to extract sparse features from raw point clouds. Point R-CNN [31] generates proposals for foreground points, 3DSSD [43] improves speed by eliminating the upsampling layer in PointNet, and IA-SSD [48] trims background points to further enhance efficiency. Other methods [4, 10, 46] convert points into voxels for feature extraction and prediction. FSD [10] employs instance-level optimization to preserve essential features, VoxelNeXt [4] predicts on the nearest voxels to object centers, and SAFDNet [46] mitigates the center feature missing through adaptive feature diffusion. Currently, there is no standardized solution to build sparse detectors with robust feature extraction capability and superior optimization.

Constructing a robust and highly optimized sparse detector presents two key challenges. First, establishing long-range interactions is difficult. Sparse detectors [4, 10, 46] typically operate only on non-empty voxels for efficiency, which often results in the inability to maintain connections between distant elements. As shown in Figure 1, two voxels become isolated because they lie outside the range of the convolution kernel. This issue does not affect dense detectors, where dense convolutions can establish connections between distant voxels through intermediary ones. Second, network optimization is challenging. Since non-empty voxels generally occupy a small proportion of each object, the object centers are often empty, particularly in the case of large objects. Many popular detectors [12, 40, 44] have shown that the center feature serves as an effective proxy for an object. As a result, the absence of center features hampers the network optimization [10].

To address the challenges outlined above, we introduce a sparse detector called the Fully Sparse Hybrid Net-

*Corresponding author.

work (FSHNet). Specifically, to tackle the first challenge, FSHNet incorporates a SlotFormer block to enable long-range interactions between sparse voxels. Unlike traditional square windows, SlotFormer divides sparse voxels into several 'slots' where each slot represents a window with infinite edge length on one side. The block then uses a linear attention mechanism [5, 7, 17, 22] to establish connections between voxels within each slot. In different layers of SlotFormer, we alternate the direction of the slots to facilitate global-range interactions. By combining SlotFormer with an existing sparse convolution encoder [4, 46], we overcome the limitations of current encoders in handling long-range interactions. To address the second challenge, i.e., network optimization, we propose a dynamic sparse label assignment method that dynamically selects positive samples from several nearest voxels to each object center, providing high-quality positive samples for training. Additionally, we introduce a sparse upsampling module to refine the sparse voxels, thus retaining fine-grained features which is useful for detecting small objects.

We compare our FSHNet with current state-of-the-art sparse detectors [4, 10, 46] and advanced dense detectors [14, 21, 40]. Experiments are conducted on the Waymo Open [34], nuScenes [2], and Argoverse2 [41] datasets. The results show that FSHNet significantly outperforms existing methods. The contributions of this work can be summarized as follows:

- We propose FSHNet, a 3D sparse detector that combines the efficiency of sparse convolution with the long-range interaction capability of the attention mechanism in a single model.
- We introduce the SlotFormer block to establish global-range interactions between sparse voxels.
- We design a dynamic sparse label assignment strategy to improve the optimization of the sparse detector.
- Extensive experiments on the Waymo Open [34], nuScenes [2], and Argoverse2 [41] datasets demonstrate the effectiveness and generalization of our method.

2. Related Work

2.1. LiDAR-based 3D Object Detection

LiDAR-based 3D object detectors can be categorized into dense detectors [6, 18, 42, 45, 50] and sparse detectors [4, 10, 35, 36, 46]. Dense detectors convert sparse voxel feature maps into dense feature maps, using dense convolution layers and dense detection heads to produce predictions similar to those in 2D object detectors [15, 29]. In contrast, sparse detectors leverage the inherent sparsity of point clouds by employing fully sparse networks and sparse detection heads for prediction.

Dense Detectors. VoxelNet [50] encodes voxel features using PointNet [27] and then extracts features from 3D feature maps through 3D convolutions. SECOND [42] improves the efficiency of encoding sparse voxel features by introducing 3D sparse convolution. PointPillars [18] divides the point cloud into pillar voxels, eliminating the need for 3D convolution and enabling high inference speed. Voxel R-CNN [6] replaces raw point features with 3D voxel features within the 3D backbone for second-stage refinement. HEDNet [45] employs encoder-decoder blocks to capture long-range dependencies among voxel features.

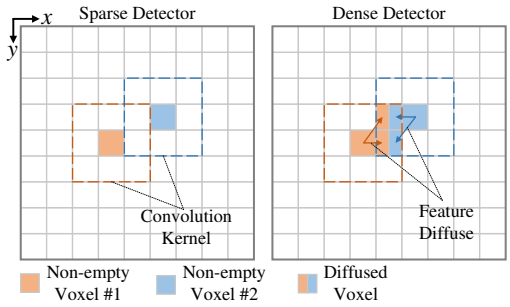

Figure 1. Comparison of voxel interactions between sparse and dense detectors. Left: two distant voxels do not interact because they fall outside each other's convolution kernel range. Right: interaction occurs between distant voxels through a diffused voxel located between them.

Sparse Detectors. RSN [35] performs foreground segmentation on range images, and then generates box predictions from sparse foreground points. FSD [10] uses an instance-level feature extraction module to mitigate the issue of center feature missing in sparse detectors. SWFormer [36] introduces sparse window transformers, creating a fully transformer-based network for 3D object detection. VoxelNeXt [4] directly predicts objects from sparse voxel features, eliminating the need for hand-crafted proxies. SAFDNet [46] incorporates an adaptive feature diffusion strategy to address the center feature missing issue.

2.2. Transformer on 3D Object Detection

Transformers have achieved significant success in both NLP [39] and image processing [8, 23], inspiring researchers to apply attention mechanisms to LiDAR-based 3D object detection. CT3D [30] introduces a channel-wise transformer to refine proposal bounding boxes. SST [9] employs a single-stride architecture using transformers to capture long-range dependencies. VoxSet [13] processes input point clouds through a set-to-set translation strategy with transformers. DSVT [40] and FlatFormer [24] adopt similar voxel partitioning strategies, sorting voxels within each window before grouping them into equal sizes. More recently, LION [25] applies linear attention to process grouped voxels within windows, while ScatterFormer [14] uses linear attention to handle voxels of varied lengths within each window.

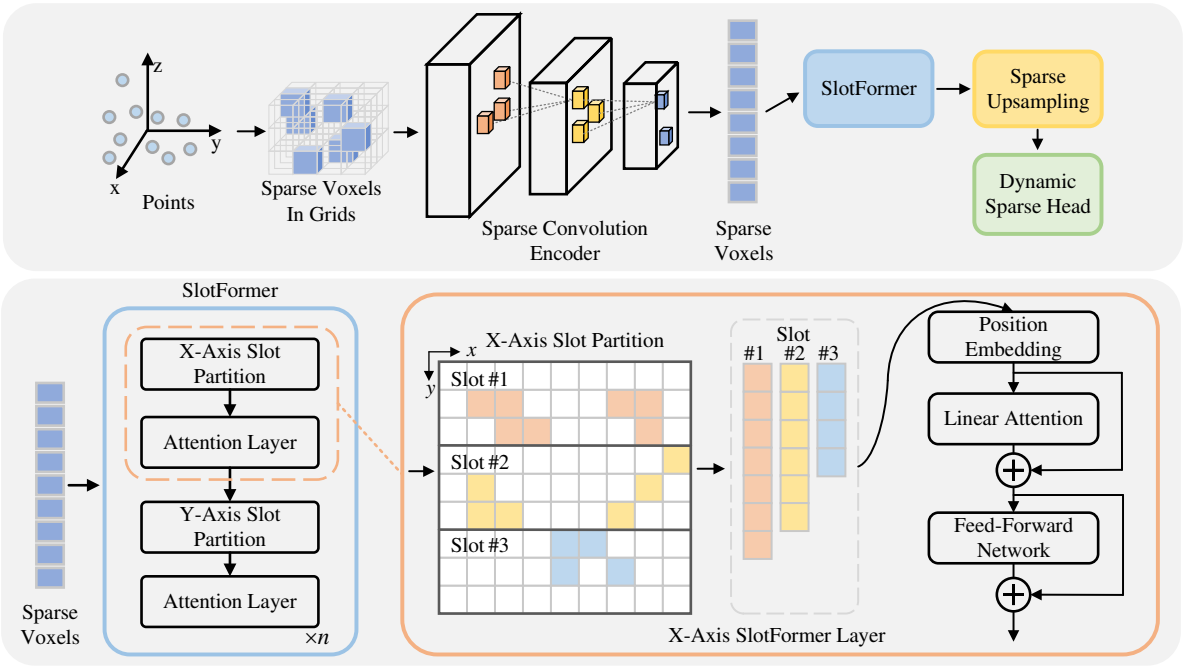

Figure 2. The overall framework of our FSHNet. Taking the raw point clouds as input, FSHNet first transforms points to voxels, and then it employs the sparse convolution encoder, SlotFormer, and sparse upsampling module to extract high-level and fine-grained sparse features. Finally, the dynamic sparse head is used to generate predictions.

2.3. Label Assignment

Label assignment is a key component of 2D and 3D object detection, critically impacting network optimization. In 2D detectors, RetinaNet [20] assigns anchors to the output grid map, FCOS [38] marks grid points within ground truth boxes as positive samples, and CenterNet [49] selects the center points of ground truth boxes as positive samples. ATSS [47] and AutoAssign [53] propose adaptive strategies for dynamic threshold selection and adjusting positive/negative confidence. For 3D dense detectors, label assignment typically follows anchor-based [6, 18, 42] or center-based [12, 16, 44] approaches. Existing 3D sparse detectors [4, 46] generally assign the nearest voxel to each object center as the positive sample. Recently, a 3D dense detector DCDet [21] dynamically assigns positive samples from a cross-shaped region around each object center. Inspired by this, we dynamically select positive samples from candidate sparse voxels near object centers.

3. Methodology

3.1. Fully Sparse Hybrid Network

Sparse detectors [4, 10, 11, 46] extract features solely from sparse voxels, eliminating the need for dense feature maps. This makes them more efficient for long-range detection tasks compared to dense detectors [21, 32, 42, 44]. However, existing sparse detectors face challenges in establishing long-range interactions between distant voxels. As shown in Figure 1, for the sparse detector, no interactions occur between distant voxels if they fall outside each other's receptive fields. In contrast, the dense detector has intermediate empty voxels that can act as bridges, facilitating interactions between distant voxels.

Increasing the kernel size can expand the receptive field, which benefits 3D object detection [3, 26]. However, naively enlarging the kernel size incurs significant time and space overhead. Some large kernel methods [3, 26] are difficult to integrate into existing sparse detectors and struggle to establish the long-range interactions necessary for perceiving the global scene. Attention mechanisms [17, 39] in vision Transformers [7, 8] are known for their ability to facilitate long-range interactions. Recently, several studies [14, 24, 25, 40] have used Transformers to extract features from raw point clouds. However, processing raw point clouds can be highly time- and resource-intensive due to their large size. In this work, we combine the sparse convolution encoder [4, 46] with attention mechanisms, leveraging the strengths of both. The sparse convolution encoder efficiently extracts downsampled sparse voxel features, while the attention mechanism enables long-range interactions among these downsampled voxels.

The overall framework of our fully sparse hybrid network is illustrated in Figure 2. It begins by converting

the input point clouds into sparse voxels using a voxel feature encoding (VFE) module, following previous methods [40, 46]. A sparse convolution encoder [4, 46] then extracts features from the sparse voxels and progressively downsamples them. Next, we use SlotFormer to establish long-range interactions among these downsampled voxels. The sparse voxels then pass through a sparse upsampling module to generate a more fine-grained representation. Finally, the upsampled sparse voxels are processed by the dynamic sparse head, which dynamically assigns positive samples for network optimization. The details of SlotFormer, sparse upsampling, and dynamic sparse head will be discussed in Sec. 3.2, Sec. 3.3 and Sec. 3.4, respectively.

3.2. SlotFormer Architecture

Existing transformer-based 3D detectors [14, 24, 25, 40] use a window partition strategy to split voxels, enabling them to achieve a large receptive field. However, they are still limited by the window size. To further expand the receptive field, we propose SlotFormer, which facilitates long-range interactions at the global scene level. We divide the scene into several slots along the X or Y axes, with each slot covering the entire scene along one axis, as illustrated in Figure 2. The primary challenge then lies in processing the voxels within each slot. Due to the large size of the slots and the sparse distribution of point clouds, the number of non-empty voxels in each slot can be large and vary greatly. Traditional self-attention has a computational complexity that grows quadratically with respect to input size, and it struggles to parallelize computations for inputs of varying lengths. As a result, self-attention is not the ideal solution for our approach.

To reduce computational costs and handle varying input lengths, we employ linear attention [5, 7, 17, 22] to process non-empty voxels within slots. Linear attention modifies the computation order of Q, K, and V compared to traditional self-attention. It first multiplies the matrices of K and V, and then multiplies the result by Q, thus avoiding the explicit computation of the attention matrix in self-attention. As a result, the computational complexity of linear attention is reduced from $\mathcal{O}\left(N^2\right)$ to $\mathcal{O}\left(N\right)$.

Specifically, given N non-empty voxels $\mathcal{V} = \{\nu_i \mid \nu_i = [(x_i, y_i)\,; f_i]\}_{i=1}^{N}$, where $(x_i, y_i) \in \mathbb{R}^2$ and $f_i \in \mathbb{R}^{1\times c}$ are the coordinate and feature of the voxel ν_i, respectively. We first partition these voxels into several slots with the size of $s \times w$, where s and w are the size of the point cloud scene and the width of the slot, respectively:

$$d_i^x = \lfloor \frac{y_i}{w} \rfloor, d_i^y = \lfloor \frac{x_i}{w} \rfloor, \tag{1}$$

where d_i^x and d_i^y are the X-axis and Y-axis slot indices of voxel ν_i, respectively. Next, we will demonstrate the calculation steps of the X-axis SlotFormer layer, therefore only d_i^x is used. Query matrix $Q = \left\{q_i \in \mathbb{R}^{1\times c}\right\}_{i=1}^{N}$, key matrix $K = \left\{k_i \in \mathbb{R}^{1\times c}\right\}_{i=1}^{N}$ and value matrix $V = \left\{v_i \in \mathbb{R}^{1\times c}\right\}_{i=1}^{N}$ are obtained as follows:

$$q_i = \phi(f_i W_q), k_i = \phi(f_i W_k), v_i = f_i W_v, \tag{2}$$

where W_q, W_k and W_v are learnable weight matrics, and $\phi(\cdot)$ denotes the kernel function implemented by ReLU function. Next, we calculate the $KV = \{kv_i \in \mathbb{R}^{c\times c}\}_{i=1}^{m}$ matrix, where m denotes the slot number:

$$kv_j = \sum_{i=0}^{N} k_i^T \cdot v_i \cdot \mathbb{I}[d_i^x == j], \tag{3}$$

where $\mathbb{I}[d_i^x == j]$ denotes whether the i-th voxel belongs to the j-th slot. The Eq. 3 sums up the products of k_i^T and v_i within a same slot. Here, we use the X-axis slots as an example. The process for Y-axis slots is similar to Eq. 3. Now, each voxel can query its new value matrix v_i' as follows:

$$v_i' = \frac{q_i \cdot kv_{d_i^x}}{q_i \sum_{j=1}^{N} k_j^T \cdot \mathbb{I}[d_j^x == d_i^x]}, \tag{4}$$

where $kv_{d_i^x}$ denotes the kv matrix of the d_i^x-th slot. The denominator of Eq. 4 is a normalization factor. Note that the above processes can be implemented with customized CUDA operations in parallel computation. Finally, the new value matrix $V' = \left\{v_i' \in \mathbb{R}^{1\times c}\right\}_{i=1}^{N}$ will go through a feed-forward network:

$$f_i' = \text{FFN}(v_i'). \tag{5}$$

The updated voxels, $\mathcal{V} = \{\nu_i \mid \nu_i = [(x_i, y_i)\,; f_i']\}_{i=1}^{N}$, will be input into the next layer. In the above case, we omit the residual connection and output projection in linear attention for simplicity.

3.3. Sparse Upsampling

Current 3D sparse encoders typically employ a multi-stride downsampling strategy, which, while reducing computational cost, often leads to the loss of fine-grained information critical for small objects. To recover the lost details, we introduce a simple yet effective module called sparse upsampling.

Let's begin by considering N non-empty voxels, $\mathcal{V} = \{\nu_i \mid \nu_i = [(x_i, y_i)\,; f_i]\}_{i=1}^{N}$, where $(x_i, y_i) \in \mathbb{R}^2$ and $f_i \in \mathbb{R}^{1\times c}$ represent the coordinates and features of voxel ν_i, respectively. Initially, we double the coordinates of these voxels (equivalently reducing the voxel size within the same point cloud range):

$$\mathcal{V}' = \{\nu_i' \mid \nu_i' = [(2x_i, 2y_i)\,; f_i]\}_{i=1}^{N}. \tag{6}$$

Then we employ a sparse convolution layer SpConv$(\cdot)$ with 3 kernel size and 1 stride to diffuse the voxels:

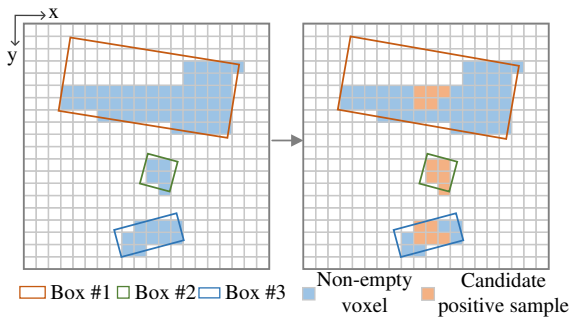

Figure 3. A demonstration of candidate positive samples in dynamic sparse label assignment.

$$\begin{aligned}\mathcal{V}^{up} =& \text{SpConv}(\mathcal{V}') \\ =& \{\nu_i^{up} \mid \nu_i^{up} = [(x_i^{up}, y_i^{up}); f_i^{up}]\}_{i=1}^{N^{up}},\end{aligned} \tag{7}$$

where (x_i^{up}, y_i^{up}) and f_i^{up} denote the coordinates and features of the upsampled voxel ν_i^{up}, respectively, and N^{up} represents the total number of upsampled voxels. The sparse upsampling module refines the voxel size to one-fourth of the original size and leverages neighboring features to recover lost detailed information.

3.4. Dynamic Sparse Head

Dynamic Sparse Label Assignment. Existing sparse detectors [4, 46] typically assign the voxel closest to the center of each annotated bounding box as a positive sample (hereafter referred to as center nearest assignment). However, this approach excludes many high-quality samples, leading to a suboptimal detection network. In contrast, the recent dense 3D detector DCDet [21] introduces a dynamic cross-based label assignment method that identifies positive samples within a cross-shaped area around each bounding box. Inspired by this, we propose a dynamic sparse label assignment strategy that selects positive samples from a set of the nearest candidate voxels to each bounding box center. This method aims to enhance network optimization by incorporating more high-quality positive samples during training. A visual demonstration of the dynamic sparse label assignment is shown in Figure 3.

Specifically, the dynamic sparse label assignment first calculates the distances between voxels and the centers of bounding boxes. For each annotated bounding box $\mathbf{b}^t$, we find the nearest n voxels $\mathcal{V}_b = \{\nu_i \mid \nu_i = [(x_i, y_i); f_i]\}_{i=1}^{n}$ to its center. And then we calculate the selection cost $\mathcal{C} = \{c_i\}_{i=1}^{n}$ of these voxels as follows:

$$c_i = \ell_{cls}(\text{Pred}(\nu_i), \mathbf{b}^t) + \lambda \ell_{reg}(\text{Pred}(\nu_i), \mathbf{b}^t), \nu_i \in \mathcal{V}_b, \tag{8}$$

where $\text{Pred}(\cdot)$ denotes the prediction module which outputs the predicted bounding boxes according to input voxels, $\ell_{cls}(\cdot,\cdot)$ and $\ell_{reg}(\cdot,\cdot)$ are the classification and regression loss functions respectively, and λ is the weight of regression loss. Then, we sum the IoUs between the ground truth $\mathbf{b}_i^t$ and predicted bounding boxes from the candidate voxels:

$$k = \max(\lfloor \sum_{\nu_i \in \mathcal{V}_b} \text{IoU}(\mathbf{b}^t, \text{Pred}(\nu_i)) \rfloor, 1). \tag{9}$$

Finally, top k voxels in $\mathcal{V}_b$ are selected as positive samples for ground truth $\mathbf{b}^t$ according the selection cost $\mathcal{C}$.

Detection Loss. We simply use two submanifold sparse convolutional layers with kernel size 1 to build the prediction module, without other complicated designs. For regression, predicted bounding boxes are generated based on the features of positive samples obtained by the dynamic sparse label assignment. We utilize the rotation-weighted IoU loss [21] as the regression loss. For classification, we adopt the heatmap target strategy. The weights of positive samples are set to 1, and the weights of negative samples from candidate voxels are set to the values of IoU between predicted and ground-truth boxes. As for the rest negative samples, the weights are all set to 0. We apply the focal loss [20] as the classification loss.

4. Experiments

4.1. Dataset

Waymo Open. We perform our primary experiments on the Waymo Open dataset [34]. It is a large and widely-used benchmark for LiDAR-based 3D object detection. The Waymo Open dataset comprises 1150 sequences, totaling over 200,000 frames, with 798 sequences for training, 202 for validation, and 150 for testing.

nuScenes. We also conduct experiments on the nuScenes dataset [2] to validate the generalization of our FSHNet. The dataset includes 1000 driving scenarios, with 700 scenarios for training, 150 for validation, and 150 for testing.

Argoverse2. Furthermore, we conduct experiments on the Argoverse2 (AV2) dataset [41] to validate the effectiveness of FSHNet in long-range detection. AV2 dataset encompasses 1000 driving scenarios, divided into 700 for training, 150 for validation, and 150 for testing. Compared to the Waymo Open and nuScenes datasets, AV2 features a greater number of long-range 3D cuboid annotations.

4.2. Implementation Details

Our method is implemented using the OpenPCDet framework [37]. All models are trained from scratch in an end-to-end fashion with the Adam optimizer and a learning rate of 0.003. There are two variants of FSHNet: one integrates the sparse encoder of VoxelNeXt with 4 SlotFormer layers, termed FSHNet$_{light}$; the other integrates the sparse encoder of SAFDNet with 8 SlotFormer layers, termed FSHNet$_{base}$. The slot width w for SlotFormer is set to 12. We set the

Method	LEVEL 2	LEVEL 1			LEVEL 2		
	mAP/mAPH	Vehicle	Pedestrian	Cyclist	Vehicle	Pedestrian	Cyclist
SECOND [42]	61.0/57.2	72.3/71.7	68.7/58.2	60.6/59.3	63.9/63.3	60.7/51.3	58.3/57.0
PointPillars [18]	62.8/57.8	72.1/71.5	70.6/56.7	64.4/62.3	63.6/63.1	62.8/50.3	61.9/59.9
IA-SSD [48]	66.8/63.3	70.5/69.7	69.4/58.5	67.7/65.3	61.6/61.0	60.3/50.7	65.0/62.7
SST [9]	67.8/64.6	74.2/73.8	78.7/69.6	70.7/69.6	65.5/65.1	70.0/61.7	68.0/66.9
CenterPoint [44]	68.2/65.8	74.2/73.6	76.6/70.5	72.3/71.1	66.2/65.7	68.8/63.2	69.7/68.5
Voxel R-CNN [6]	68.6/66.2	76.1/75.7	78.2/72.0	70.8/69.7	68.2/67.7	69.3/63.6	68.3/67.2
PV-RCNN [32]	69.6/67.2	78.0/77.5	79.2/73.0	71.5/70.3	69.4/69.0	70.4/64.7	69.0/67.8
CenterFormer [51]	71.1/68.9	75.0/74.4	78.6/73.0	72.3/71.3	69.9/69.4	73.6/68.3	69.8/68.8
PillarNeXt [19]	71.9/69.7	78.4/77.9	82.5/77.1	73.2/72.2	70.3/69.8	74.9/69.8	70.6/69.6
DSVT (Voxel) [40]	74.0/72.1	79.7/79.3	83.7/78.9	77.5/76.5	71.4/71.0	76.1/71.5	74.6/73.7
DCDet [21]	75.0/72.7	79.5/79.0	84.1/78.5	79.4/78.3	71.6/71.1	76.7/71.3	76.8/75.7
LION-Mamba [25]	75.1/73.2	79.5/79.1	84.9/80.4	79.7/78.7	71.1/70.7	77.5/73.2	76.7/75.8
HEDNet [45]	75.3/73.4	81.1/80.6	84.4/80.0	78.7/77.7	73.2/72.7	76.8/72.6	75.8/74.9
ScatterFormer [14]	75.7/73.8	81.0/80.5	84.5/79.9	79.9/78.9	73.1/72.7	77.0/72.6	77.1/76.1
SWFormer [36]†	-/-	77.8/77.3	80.9/72.7	-/-	69.2/68.8	72.5/64.9	-/-
VoxelNeXt [4]†	72.2/70.1	78.2/77.7	81.5/76.3	76.1/74.9	69.9/69.4	73.5/68.6	73.3/72.2
FSDv1 [10]†	72.9/70.8	79.2/78.8	82.6/77.3	77.1/76.0	70.5/70.1	73.9/69.1	74.4/73.3
FSDv2 [11]†	75.6/73.5	79.8/79.3	84.8/79.7	80.7/79.6	71.4/71.0	77.4/72.5	77.9/76.8
SAFDNet [46]†	75.7/73.9	80.6/80.1	84.7/80.4	80.0/79.0	72.7/72.3	77.3/73.1	77.2/76.2
FSHNet$_{light}$ (Ours)†	76.3/74.2	81.0/80.5	85.6/80.7	80.2/79.1	73.0/72.5	78.6/73.7	77.4/76.4
FSHNet$_{base}$ (Ours)†	77.1/74.9	82.2/81.7	85.9/80.8	80.5/79.4	74.5/74.0	78.9/73.9	78.0/76.9

Table 1. Performance comparisons on the Waymo Open *validation* set. The results of AP/APH are reported. † represents a fully sparse detector, the same as below. All models are trained under the single-frame setting.

Method	LEVEL 2	LEVEL 1			LEVEL 2		
	mAP/mAPH	Vehicle	Pedestrian	Cyclist	Vehicle	Pedestrian	Cyclist
CenterPoint [44]	-	80.2/79.7	78.3/72.1	-	72.2/71.8	72.2/66.4	-
AFDetV2 [16]	72.2/70.0	80.5/80.0	79.8/74.4	72.4/71.2	73.0/72.6	73.7/68.6	69.8/68.7
PV-RCNN++ [33]	72.4/70.2	81.6/81.2	80.4/75.0	71.9/70.8	73.9/73.5	74.1/69.0	69.3/68.2
FSDv2 [11]†	75.4/73.3	82.4/82.0	83.8/78.9	77.1/76.0	74.4/74.0	77.4/72.8	74.3/73.2
SAFDNet [46]†	76.5/74.6	83.9/83.5	84.3/79.8	77.5/76.3	76.6/76.2	78.4/74.1	74.6/73.4
FSHNet$_{light}$ (Ours)†	77.0/74.9	83.4/83.0	84.7/79.8	78.9/77.7	76.1/75.7	78.8/74.1	76.1/74.9
FSHNet$_{base}$ (Ours)†	77.4/75.2	84.9/84.5	85.3/80.0	77.7/76.5	77.8/77.4	79.5/74.3	74.8/73.7

Table 2. Performance comparisons on the Waymo Open *test* set by submitting to the official test evaluation server. The results are achieved by using the single-frame setting. No test-time augmentations are used.

number of candidate voxels in sparse dynamic label assignment to 5. We observe that our FSHNet achieves convergence more rapidly than previous methods. Consequently, the FSHNet variants are trained for 12, 36, and 12 epochs on the Waymo Open, nuScenes (without the CBGS strategy [52]), and Argoverse2 datasets, respectively. The detection ranges of the three datasets are 75, 54, and 200 meters, respectively. The voxel sizes are $(0.08, 0.08, 0.15)m$, $(0.075, 0.075, 0.2)m$, and $(0.1, 0.1, 0.2)m$ for the three datasets, respectively. All experiments are conducted on 2 A100 GPUs with a total batch size of 16. Additional experiments including visualizations, runtime, and hyper-parameter analysis are in supplementary materials.

4.3. Comparison with State-of-the-Art Methods

Results on Waymo Open. We conduct a comprehensive comparison of FSHNet against the current state-of-the-art (SOTA) 3D detectors on the validation set of the Waymo Open Dataset. As demonstrated in Table 1, our FSHNet$_{light}$ and FSHNet$_{base}$ outperform most dense and sparse detectors, including the most recent SOTA methods such as ScatterFormer [14] and SAFDNet [46]. Moreover, FSHNet$_{base}$ surpasses the previous methods by a substantial margin. Additionally, we evaluated our method on the Waymo Open test set by submitting the results to the official server. The performance comparisons are presented in Table 2, revealing that our FSHNet$_{light}$ and FSHNet$_{base}$ achieve SOTA per-

Method	NDS	mAP	Car	Truck	Bus	T.L.	C.V.	Ped.	M.T.	Bike	T.C.	B.R.
CenterPoint [44]	66.5	59.2	84.9	57.4	70.7	38.1	16.9	85.1	59.0	42.0	69.8	68.3
PillarNeXt [19]	68.4	62.2	85.0	57.4	67.6	35.6	20.6	86.8	68.6	53.1	77.3	69.7
VoxelNeXt [4]†	68.7	63.5	83.9	55.5	70.5	38.1	21.1	84.6	62.8	50.0	69.4	69.4
TransFusion-L[1]	70.1	65.5	86.9	60.8	73.1	43.4	25.2	87.5	72.9	57.3	77.2	70.3
FSDv2 [11]†	70.4	64.7	84.4	57.3	75.9	44.1	28.5	86.9	69.5	57.4	72.9	73.6
SAFDNet [46]†	71.0	66.3	87.6	60.8	78.0	43.5	26.6	87.8	75.5	58.0	75.0	69.7
DSVT [40]	71.1	66.4	87.4	62.6	75.9	42.1	25.3	88.2	74.8	58.7	77.8	70.9
FSHNet$_{base}$ (Ours)†	71.7	68.1	88.7	61.4	79.3	47.8	26.3	89.3	76.7	60.5	78.6	72.3

Table 3. Performance comparisons on the nuScenes *validation* set. 'T.L.', 'C.V.', 'Ped.', 'M.T.', 'T.C.', and 'B.R.' denote trailer, construction vehicle, pedestrian, motor, traffic cone, and barrier, respectively.

Method	mAP	Vehicle	Bus	Pedestrian	Stop Sign	Box Truck	Bollard	C-Barrel	Motorcyclist	MPC-Sign	Motorcycle	Bicycle	A-Bus	School Bus	Truck Cab	C-Cone	V-Trailer	Sign	Large Vehicle	Stroller	Bicyclist
CenterPoint [44]	22.0	67.6	38.9	46.5	16.9	37.4	40.1	32.2	28.6	27.4	33.4	24.5	8.7	25.8	22.6	29.5	22.4	6.3	3.9	0.5	20.1
HEDNet [45]	37.1	78.2	47.7	67.6	46.4	45.9	56.9	67.0	48.7	46.5	58.2	47.5	23.3	40.9	27.5	46.8	27.9	20.6	6.9	27.2	38.7
FSDv1 [10]†	28.2	68.1	40.9	59.0	29.0	38.5	41.8	42.6	39.7	26.2	49.0	38.6	20.4	30.5	14.8	41.2	26.9	11.9	5.9	13.8	33.4
FSDv2 [11]†	37.6	77.0	47.6	70.5	43.6	41.5	53.9	58.5	56.8	39.0	60.7	49.4	28.4	41.9	30.2	44.9	33.4	16.6	7.3	32.5	45.9
VoxelNeXt [4]*†	33.1	74.8	43.2	65.5	36.3	43.1	58.7	67.5	29.4	43.3	54.0	45.4	21.7	41.8	23.8	51.2	28.3	17.9	6.3	13.1	29.8
SAFDNet [46]*†	38.7	77.7	47.3	69.4	49.8	45.9	62.2	69.5	52.5	45.4	59.5	48.5	29.6	43.3	25.7	54.2	31.2	20.5	9.3	36.0	48.5
FSHNet$_{base}$ (Ours)†	40.2	77.4	48.3	72.9	50.0	47.2	63.4	69.9	56.1	43.8	62.5	53.7	31.8	44.8	28.9	56.6	32.0	23.0	8.7	34.5	41.0

Table 4. Performance comparisons on the Argoverse2 *validation* set. '*' represents that the method is re-implemented using the codebase OpenPCDet [37]. We exclude categories with few instances but still calculate the average across all categories, including those omitted.

formance according to the LEVEL 2 mAP/mAPH. Notably, FSHNet$_{base}$ significantly surpasses previous methods, particularly in the vehicle and pedestrian categories. The improvement mentioned above can be attributed to the components of FSHNet. SlotFormer enhances the long-range interaction capability, which is beneficial for detecting large objects. The sparse upsampling module captures fine-grained content for small objects. The sparse dynamic label assignment better optimizes the detection network.

Results on nuScenes. We further conduct experiments on the nuScenes dataset to compare our method with other SOTA LiDAR-based methods. Our FSHNet$_{base}$ are trained for 36 epochs without CBGS strategy [52], which typically extends training iterations by about 4.5 times. Compared to other methods trained for 20 epochs with CBGS, our FSHNet$_{base}$ undergoes fewer training iterations. Nevertheless, it still outperforms other methods in NDS and mAP metrics, as shown in Table 3.

Results on Argoverse2. To validate the effectiveness of FSHNet on long-range detection, we conduct experiments on the Argoverse2 dataset with a perception range of 200 meters. We re-implement two recent sparse detectors including VoxelNeXt and SAFDNet for fair comparison. The re-implemented detectors adopt the same training scheme as our method, including ground-truth sampling to alleviate the long-tail problem. As demonstrated in Table 4, our FSHNet$_{base}$ substantially outperforms the current SOTA sparse detector SAFDNet, across small objects (pedestrian +3.2% mAP, bicycle +5.2% mAP) and large objects (bus +1.0% mAP, A-bus +2.2% mAP).

4.4. Ablation Study

To further study the influence of each component of FSHNet, we perform a comprehensive ablation analysis on the Waymo Open dataset. Following prior works [21, 40], models are trained on 20% training samples and the LEVEL 2 AP/APH results on the whole *validation* set are reported.

Effect of Different Components. We employ VoxelNeXt with the center nearest label assignment as our baseline model. To assess the effectiveness of our proposed methods, we systematically integrate SlotFormer (*SF*), dynamic sparse label assignment (*DSLA*), and sparse upsampling (*SU*) into the baseline model. The ablation study results are shown in Table 5. We note a significant performance enhancement when incorporating SlotFormer, as evidenced by the results in the 2^{nd} and 6^{th} rows of Table 5. Particularly for large objects such as vehicles and cyclist categories, the improvements are marked, illustrating the importance of long-range interactions. Additionally, models trained with *DSLA* consistently yield significantly better performance, as portrayed in the 3^{rd} and 5^{th} rows of Table 5. This suggests that *DSLA* aids in the optimization of the detection network. We also observe that the sparse upsampling significantly enhances the performance on small objects (e.g., pedestrians)

SF	*DSLA*	*SU*	Vehicle	Pedestrian	Cyclist
			69.1/68.7	75.3/69.5	75.0/73.9
✓			70.3/69.9	75.9/70.5	76.2/75.1
	✓		69.9/69.5	75.5/69.7	75.7/74.5
		✓	69.3/68.8	76.6/71.0	75.2/74.1
	✓	✓	70.5/70.1	77.1/71.6	75.6/74.5
✓	✓	✓	72.5/72.0	77.9/72.6	77.2/76.1

Table 5. Effect of different components of FSHNet.

Voxel Partition	Vehicle	Pedestrian	Cyclist
baseline	70.5/70.1	77.1/71.6	75.6/74.5
Win.+Set w/ self-attn	71.8/71.3	77.3/72.0	76.8/75.7
Win.+Set w/ linear attn	72.0/71.5	77.7/72.3	77.1/76.0
Win. w/ linear attn	72.2/71.7	77.8/72.4	77.1/76.1
Slot w/ linear attn	72.5/72.0	77.9/72.6	77.2/76.1

Table 6. Comparison results of different voxel partition manners.

as demonstrated in the 4^{th} row of Table 5. This affirms the effectiveness of fine-grained details preserved by the sparse upsampling.

Different Types of Voxel Partition Manners. Table 6 presents a comparison of results across different voxel partition manners. We employ the FSHNet$_{light}$ without the SlotFormer as the baseline. 'Win.+Set' refers to partitioning voxels into windows, then sorting and grouping them into equal-sized sets, as in [40]. 'Win.' represents only assigning voxels to distinct windows, and 'Slot' is our partition approach, detailed in Sec. 3.2. 'w/ self-attn' and 'w/ linear attn' indicate the use of self-attention and linear attention mechanisms for voxel feature extraction, respectively. As shown in Table 6, the attention mechanisms combined with different voxel partition manners significantly improve the baseline model's performance, attributed to their long-range interaction capability. Among these manners, our slot partition with linear attention achieves the best results, as it effectively establishes global-level voxel interactions.

Different Types of Upsampling Strategies. Table 7 provides a comparison of various upsampling strategies for sparse voxels. We use FSHNet$_{light}$ without the sparse upsampling module as the baseline, shown in the 1^{st} row of Table 7. *SM-SU* denotes a method where the features of large voxels are repeated to their inner small voxels, followed by a submanifold sparse convolution layer [42] to process these features. *SP-SU*, in contrast, assigns the features of large voxels to one inner small voxel, then uses a sparse convolution layer [42] to diffuse the voxel features. Both the *SM-SU* and *SP-SU* upsampling strategies significantly improve detection performance on small objects (e.g., pedestrians) compared to the baseline model. Additionally, the *SP-SU* upsampling strategy enhances performance on large objects (e.g., vehicles and cyclists), likely because its voxel diffusion mitigates the issue of center feature missing.

Type	Vehicle	Pedestrian	Cyclist
-	71.4/70.9	75.9/70.4	76.6/75.4
SM-SU	71.7/71.2	77.9/72.4	76.6/75.6
SP-SU	72.5/72.0	77.9/72.6	77.2/76.1

Table 7. Comparison results of different sparse upsampling types.

n	Vehicle	Pedestrian	Cyclist
-	70.8/70.3	77.3/71.7	76.1/75.0
1	70.8/70.3	77.1/71.5	77.0/76.0
3	71.3/70.8	77.6/72.0	77.0/75.9
5	72.5/72.0	77.9/72.6	77.2/76.1
7	72.2/71.7	77.8/72.4	77.2/76.0

Table 8. Comparison results of different candidate numbers.

Different Candidate Number of *DSLA*. Table 8 demonstrates a comparison of different candidate numbers for *DSLA*. We use FSHNet$_{light}$ without *DSLA* (employing the center nearest assignment strategy) as the baseline, shown in the 1^{st} row of Table 8. Increasing the candidate number from 1 to 5 progressively enhances the detection performance, as demonstrated in the 2^{nd} to 4^{th} rows of Table 8. However, further increasing the candidate number does not yield additional improvement, as shown in the last row of Table 8. Therefore, we set the candidate number to 5 as the default configuration.

5. Conclusion

In this paper, we propose a sparse detector termed FSHNet which combines the efficiency of sparse convolutions with the long-range interaction capabilities of the attention mechanism within a single model. It incorporates the SlotFormer block to facilitate long-range interactions between sparse voxels. Additionally, FSHNet employs dynamic sparse label assignment to generate high-quality positive samples, enhancing network optimization. To further boost performance for small objects, we introduce a sparse upsampling module that preserves fine-grained details. Extensive experiments on several datasets demonstrate the effectiveness and generalization of FSHNet.

Limitations. We employ the SlotFormer to enable long-range interactions between sparse voxels. However, it introduces additional latency to sparse detectors. Specifically, for FSHNet$_{light}$ and FSHNet$_{base}$, SlotFormer adds approximately 16 *ms* (from 65 to 81 *ms*) and 29 ms (from 94 to 123 *ms*) of latency, respectively, when tested on a single RTX 3090 GPU. A more efficient sparse architecture could potentially achieve the same long-range interaction capability as FSHNet. We leave this for future research.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 62232008, in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2025A1515011485, in part by the National Natural Science Foundation of China under Grant 61902442, and in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2025A1515010252.

References

[1] Xuyang Bai, Zeyu Hu, Xinge Zhu, Qingqiu Huang, Yilun Chen, Hongbo Fu, and Chiew-Lan Tai. Transfusion: Robust lidar-camera fusion for 3d object detection with transformers. In *CVPR*, 2022.

[2] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *cvpr*, 2020.

[3] Yukang Chen, Jianhui Liu, Xiangyu Zhang, Xiaojuan Qi, and Jiaya Jia. Largekernel3d: Scaling up kernels in 3d sparse cnns. In *CVPR*, 2023.

[4] Yukang Chen, Jianhui Liu, Xiangyu Zhang, Xiaojuan Qi, and Jiaya Jia. Voxelnext: Fully sparse voxelnet for 3d object detection and tracking. In *CVPR*, 2023.

[5] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking attention with performers. In *ICLR*, 2021.

[6] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. In *AAAI*, 2021.

[7] Han Dongchen, Pan Xuran, Han Yizeng, Song Shiji, and Huang Gao. Flatten transformer: Vision transformer using focused linear attention. In *CVPR*, 2023.

[8] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[9] Lue Fan, Ziqi Pang, Tianyuan Zhang, Yu-Xiong Wang, Hang Zhao, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Embracing single stride 3d object detector with sparse transformer. In *CVPR*, 2022.

[10] Lue Fan, Feng Wang, Naiyan Wang, and Zhao-Xiang Zhang. Fully sparse 3d object detection. In *NeurIPS*, 2022.

[11] Lue Fan, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Fsd v2: Improving fully sparse 3d object detection with virtual voxels. *arXiv preprint arXiv:2308.03755*, 2023.

[12] Runzhou Ge, Zhuangzhuang Ding, Yihan Hu, Yu Wang, Sijia Chen, Li Huang, and Yuan Li. Afdet: Anchor free one stage 3d object detection. *arXiv preprint arXiv:2006.12671*, 2020.

[13] Chenhang He, Ruihuang Li, Shuai Li, and Lei Zhang. Voxel set transformer: A set-to-set approach to 3d object detection from point clouds. In *CVPR*, 2022.

[14] Chenhang He, Ruihuang Li, Guowen Zhang, and Lei Zhang. Scatterformer: Efficient voxel transformer with scattered linear attention. In *ECCV*, 2024.

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

[16] Yihan Hu, Zhuangzhuang Ding, Runzhou Ge, Wenxin Shao, Li Huang, Kun Li, and Qiang Liu. Afdetv2: Rethinking the necessity of the second stage for object detection from point clouds. In *AAAI*, 2022.

[17] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*, 2020.

[18] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019.

[19] Jinyu Li, Chenxu Luo, and Xiaodong Yang. Pillarnext: Rethinking network designs for 3d object detection in lidar point clouds. In *CVPR*, 2023.

[20] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

[21] Shuai Liu, Boyang Li, Zhiyu Fang, and Kai Huang. Dcdet: Dynamic cross-based 3d object detector. In *IJCAI*, 2024.

[22] Xinyu Liu, Houwen Peng, Ningxin Zheng, Yuqing Yang, Han Hu, and Yixuan Yuan. Efficientvit: Memory efficient vision transformer with cascaded group attention. In *CVPR*, 2023.

[23] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *CVPR*, 2021.

[24] Zhijian Liu, Xinyu Yang, Haotian Tang, Shang Yang, and Song Han. Flatformer: Flattened window attention for efficient point cloud transformer. In *CVPR*, 2023.

[25] Zhe Liu, Jinghua Hou, Xinyu Wang, Xiaoqing Ye, Jingdong Wang, Hengshuang Zhao, and Xiang Bai. Lion: Linear group rnn for 3d object detection in point clouds. In *ECCV*, 2024.

[26] Tao Lu, Xiang Ding, Haisong Liu, Gangshan Wu, and Limin Wang. Link: Linear kernel for lidar-based 3d perception. In *CVPR*, 2023.

[27] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017.

[28] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017.

[29] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *TPAMI*, 2016.

[30] Hualian Sheng, Sijia Cai, Yuan Liu, Bing Deng, Jianqiang Huang, Xian-Sheng Hua, and Min-Jian Zhao. Improving 3d object detection with channel-wise transformer. In *ICCV*, 2021.

[31] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019.

[32] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *CVPR*, 2020.

[33] Shaoshuai Shi, Li Jiang, Jiajun Deng, Zhe Wang, Chaoxu Guo, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection. *IJCV*, 2023.

[34] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *CVPR*, 2020.

[35] Pei Sun, Weiyue Wang, Yuning Chai, Gamaleldin Elsayed, Alex Bewley, Xiao Zhang, Cristian Sminchisescu, and Dragomir Anguelov. Rsn: Range sparse net for efficient, accurate lidar 3d object detection. In *CVPR*, 2021.

[36] Pei Sun, Mingxing Tan, Weiyue Wang, Chenxi Liu, Fei Xia, Zhaoqi Leng, and Dragomir Anguelov. Swformer: Sparse window transformer for 3d object detection in point clouds. In *ECCV*, 2022.

[37] OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. `https://github.com/open-mmlab/OpenPCDet`, 2020.

[38] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *ICCV*, 2019.

[39] A Vaswani. Attention is all you need. In *NeurIPS*, 2017.

[40] Haiyang Wang, Chen Shi, Shaoshuai Shi, Meng Lei, Sen Wang, Di He, Bernt Schiele, and Liwei Wang. Dsvt: Dynamic sparse voxel transformer with rotated sets. In *CVPR*, 2023.

[41] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, et al. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *NeurIPS Datasets and Benchmarks*, 2021.

[42] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 2018.

[43] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *CVPR*, 2020.

[44] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *CVPR*, 2021.

[45] Gang Zhang, Chen Junnan, Guohuan Gao, Jianmin Li, and Xiaolin Hu. Hednet: A hierarchical encoder-decoder network for 3d object detection in point clouds. In *NeurIPS*, 2023.

[46] Gang Zhang, Junnan Chen, Guohuan Gao, Jianmin Li, Si Liu, and Xiaolin Hu. Safdnet: A simple and effective network for fully sparse 3d object detection. In *CVPR*, 2024.

[47] Shifeng Zhang, Cheng Chi, Yongqiang Yao, Zhen Lei, and Stan Z Li. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *CVPR*, 2020.

[48] Yifan Zhang, Qingyong Hu, Guoquan Xu, Yanxin Ma, Jianwei Wan, and Yulan Guo. Not all points are equal: Learning highly efficient point-based detectors for 3d lidar point clouds. In *CVPR*, 2022.

[49] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.

[50] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, 2018.

[51] Zixiang Zhou, Xiangchen Zhao, Yu Wang, Panqu Wang, and Hassan Foroosh. Centerformer: Center-based transformer for 3d object detection. In *ECCV*, 2022.

[52] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection. *arXiv preprint arXiv:1908.09492*, 2019.

[53] Benjin Zhu, Jianfeng Wang, Zhengkai Jiang, Fuhang Zong, Songtao Liu, Zeming Li, and Jian Sun. Autoassign: Differentiable label assignment for dense object detection. *arXiv preprint arXiv:2007.03496*, 2020.