DSPDet3D: 3D Small Object Detection with Dynamic Spatial Pruning

Xiuwei Xu[1⋆], Zhihao Sun[2*], Ziwei Wang[3], Hongmin Liu[2†], Jie Zhou[1], Jiwen Lu[1†]
[1]Tsinghua University
[2]University of Science and Technology Beijing
[3]Carnegie Mellon University
xxw21@mails.tsinghua.edu.cn; d202210361@xs.ustb.edu.cn;
ziweiwa2@andrew.cmu.edu; hmliu@ustb.edu.cn;
{jzhou, lujiwen}@tsinghua.edu.cn

Abstract. In this paper, we propose an efficient feature pruning strategy for 3D small object detection. Conventional 3D object detection methods struggle on small objects due to the weak geometric information from a small number of points. Although increasing the spatial resolution of feature representations can improve the detection performance on small objects, the additional computational overhead is unaffordable. With in-depth study, we observe the growth of computation mainly comes from the upsampling operation in the decoder of 3D detector. Motivated by this, we present a multi-level 3D detector named DSPDet3D which benefits from high spatial resolution to achieves high accuracy on small object detection, while reducing redundant computation by only focusing on small object areas. Specifically, we theoretically derive a dynamic spatial pruning (DSP) strategy to prune the redundant spatial representation of 3D scene in a cascade manner according to the distribution of objects. Then we design DSP module following this strategy and construct DSPDet3D with this efficient module. On ScanNet and TO-SCENE dataset, our method achieves leading performance on small object detection. Moreover, DSPDet3D trained with only ScanNet rooms can generalize well to scenes in larger scale. It takes less than 2s to directly process a whole building consisting of more than 4500k points while detecting out almost all objects, ranging from cups to beds, on a single RTX 3090 GPU. Code.

Keywords: 3D small object detection · Spatial pruning · Efficient inference

1 Introduction

3D object detection is a fundamental scene understanding problem, which aims to detect 3D bounding boxes and semantic labels from a point cloud of 3D scene. With the recent advances of deep learning techniques on point cloud understanding [7, 13, 34, 35], 3D detection methods have shown remarkable progress [39, 40, 46, 56]. However, with 3D object detection being widely adopted in fields like robotics [30, 57] and autonomous driving [2] which require highly precise and fine-grained perception, small object detection becomes one of the most important yet unsolved problems. In autonomous driving

⋆ Equal contribution. † Corresponding author.

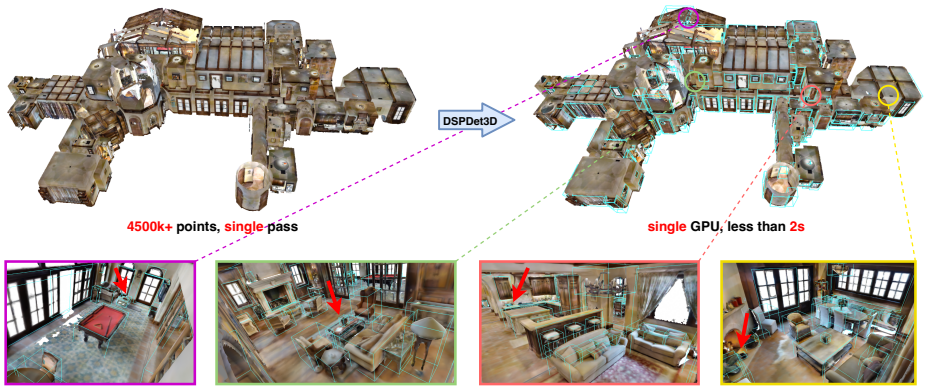

Fig. 1: Trained with only rooms from ScanNet, our DSPDet3D generalizes well to process a whole house with dozens of rooms. It takes less than 2s to generate fine-grained detection results with a RTX 3090 single GPU.

scenarios [12], we observe a significant performance gap between cars and pedestrians. In indoor scenes [4, 9] where the size variance is much larger (e.g. a bed is 1000x larger than a cup), detecting small objects is more difficult. We focus on indoor 3D object detection task where scenes are crowded with objects of multiple categories and sizes.

For indoor 3D object detection, although great improvement has been achieved in both speed and accuracy on previous benchmarks [1, 9, 43], they are still far from general purpose 3D object detection due to the limited range of object size they can handle. For instance, these methods focus on furniture-level objects such as bed and table, while smaller ones like laptop, keyboard and bottle are ignored. With the arrival of 3D small object benchmarks [37, 50, 51] which contain objects with wider size variance (e.g. from tabletop object like cup to large furniture like bed), it is shown that previous 3D detectors get very low accuracy on small objects and some even fail to detect any small objects. This is because extracting fine-grained representation for a large scene is too computationally expensive, so current methods aggressively downsamples the 3D features, which harms the representation of small objects.

In this paper, we propose a dynamic spatial pruning approach for 3D small object detection. Although increasing the spatial resolution of the feature representations is a simple and effective way to boost the performance of 3D small object detection, the large computational overhead makes this plan infeasible for real application. With in-depth study, we observe the memory footprint mainly comes from the huge number of features generated by the upsampling operation in the decoder of 3D detector. Inspired by the fact that small objects only occupy a small proportion of space, we adopt a multi-level detection framework to detect different sizes of objects in different levels. As the multi-level detector has already detected out larger objects in lower resolution, there are many redundant features in the scene representations of higher resolution. To this end, we propose to dynamically prune the features after detecting out objects in each level, which skips the upsampling operation at regions where there is no smaller object. Specifically, we first theoretically derive a pruning mask generation strategy to

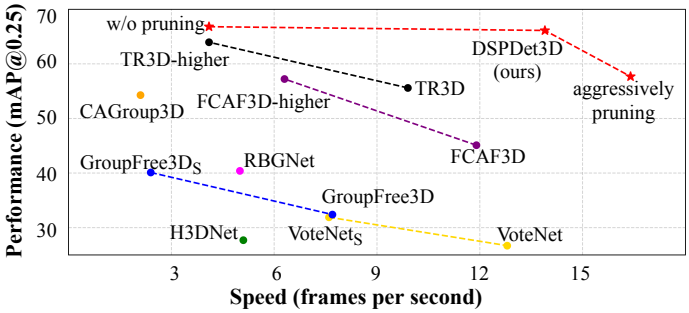

Fig. 2: Detection accuracy (mAP@0.25 of all categories) and speed (FPS) of mainstream 3D object detection methods on TO-SCENE dataset. Our DSPDet3D shows absolute advantage on 3D small object detection and provides flexible accuracy-speed tradeoff by simply adjusting the pruning threshold without retraining.

supervise the pruning module, which prunes as much features as possible while not affecting the features of object proposals. Then we design a dynamic spatial pruning (DSP) module according to the theoretical analysis and use it to construct a 3D object detector named DSPDet3D. On the popular ScanNet [9] dataset, DSPDet3D improves the mAP of all categories by 3% and mAP of small object by 14% compared with current state-of-the-art. On TO-SCENE [50] dataset with more tabletop objects, we improve the mAP of all categories by 8% while achieving leading inference speed among all mainstream indoor 3D object detection methods.

2 Related Work

Indoor 3D object detection: Since PointNet and PointNet++ [34, 35], deep learning-based 3D detection methods for point clouds begin to emerge in recent years, which can be mainly divided into three categories: voting-based [6,33,47,49,54], transformer-based [26, 28] and voxel-based [14, 38, 39, 46] methods. Inspired by 2D hough voting, VoteNet [33] proposes the first voting-based 3D detector, which aggregates the point features on surfaces into object center by 3D voting and predicts bounding boxes from the voted centers. Drawing on the success of transformer-based detector [3] in 2D domain, GroupFree3D [26] and 3DETR [28] adopts transformer architecture to decode the object proposals into 3D boxes. As extracting point features require time-consuming sampling and aggregation operation, GSDN [14] proposes a fully convolutional detection network based on sparse convolution [7, 13, 19, 52], which achieves much faster speed. FCAF3D [38] and TR3D [39] further improves the performance of GSDN with a simple anchor-free architecture. Our method also adopts voxel-based architecture considering its efficiency and scalability.

Small object detection: Small object detection [45] is a challenging problem in 2D vision due to the low-resolution features. To tackle this, a series of methods have been proposed, which can be categorized into three types: (1) small object augmentation and oversampling methods [17, 25, 58]; (2) scale-aware training and inference

strategy [11, 31, 41, 42]; (3) increasing the resolution of features or generating high-resolution features [5, 10, 21, 22, 48, 53]. However, there are far less works about 3D small object detection due to the limit of data and network capability. BackToReality [51] proposes ScanNet-md40 benchmark which contains small objects and finds many current methods suffer a lot in small object detection. TO-SCENE [50] proposes a new dataset and learning strategy for understanding 3D tabletop scenes. However, it relies on densely sampled points from CAD models, which is infeasible in practical scenarios where the points from small objects are very sparse. In contrast, we aim to directly detect small objects from naturally sampled point clouds.

Network pruning: Network pruning can be divided into two categories: architecture pruning [15, 16, 18, 20, 27, 29] and spatial pruning [24, 36, 55]. Architecture pruning aims to remove a portion of weights from a neural network to shrink the size of a network, which includes unstructured pruning [15, 18, 29] and structured pruning [16, 20, 27]. The former removes network weights without a predefined structure, while the latter removes whole channels or network layers. On the contrary, spatial pruning does not prune the parameters of a network, but spatially removing redundant computation on the feature maps. DynamicViT [36] prunes the tokens in vision transformer with an attention masking strategy. SPS-Conv [24] dynamically prunes the convolutional kernel to supress the activation on background voxels in sparse convolution layer. Ada3D [55] proposes a pruning framework for 3D and BEV features. Our dynamic spatial pruning method also belongs to spatial pruning, which directly removes redundant voxel features level by level according to the distribution of objects.

3 Approach

In this section, we describe our DSPDet3D for efficient 3D small object detection. We first revisit the multi-level 3D detector and analyze the computational cost distribution. Then we propose dynamic spatial pruning with theoretical analysis on how to prune features without affecting detection performance. Finally we design DSP module according to the theoretical analysis and use it to construct DSPDet3D.

3.1 Analysis on Multi-level 3D Detector

Preliminaries: We choose multi-level FCOS-like [44] 3D detector [38, 39] with sparse convolution [7, 13] for small object detection due to its high performance on both accuracy and speed (more detail can be found in Table 1 and 2).

As shown in Figure 3 (middle), after extracting backbone features, multi-level detector iteratively upsamples the voxel feature representations to different levels. In each level, all voxels are regarded as object proposals to predict bounding boxes and category scores. Generative upsampling is widely adopted in this kind of architectures [14, 38, 39] to expand the voxels from object surfaces to the whole 3D space, where object proposals located at object centers can produce accurate predictions. During training, ground-truth bounding boxes are assigned to different levels and each box assigns several nearby voxels as positive object proposals. Only box predictions from positive object proposals will be supervised. While at inference time all voxel features from the decoder are used to predict bounding boxes, which are then filtered by 3D NMS.

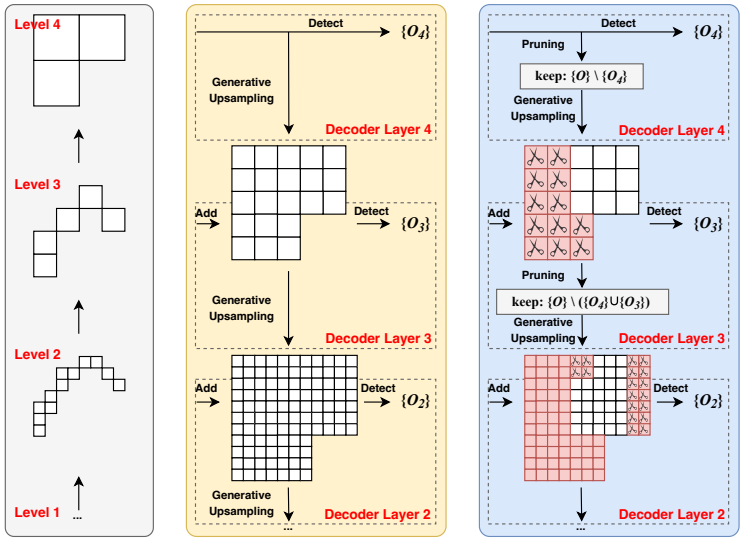

Fig. 3: Comparison of the decoder in typical multi-level 3D object detector [39] and our DSPDet3D. Note that the sparsity of voxels in decoder is changed due to the generative upsampling operation. After detecting out objects in a level, DSPDet3D prunes redundant voxel features according to the distribution of objects before each upsampling operation. Red boxes indicate all pruned voxels and 'scissor' boxes indicate voxels pruned in the previous layer. $\{O\}$ is the set of all objects and $\{O_i\}$ is the set of objects assigned to level i.

Increasing spatial resolution: Based on multi-level architecture, a simple way to boost the performance of small object detection is to increase the spatial resolution of feature maps, i.e., voxelizing the point clouds into smaller voxels to better preserve geometric information. Taking TR3D [39] for example, we double its spatial resolution and show the results in Figure 4. It can be seen that the performance on small object really benefits from larger resolution, but the computational overhead grows dramatically at the same time. As 3D object detection is usually adopted in tasks which requires real-time inference under limited resources, such as AR/VR and robotic navigation, directly increasing spatial resolution is infeasible. Notably, we find the computation growth is imbalanced: the decoder layers (including detection heads) account for the most memory footprint and have larger memory growth ratio than the backbone. This indicates the generative upsampling operation will significantly increase the number of voxels when the spatial resolution is high, which is the main challenge for scaling up the spatial resolution of multi-level detectors.

3.2 Dynamic Spatial Pruning

Since small objects only occupy a small proportion of space, we assume there is a large amount of redundant computation in decoder layers, especially when the resolution is high. For instance, if a bed is detected in Layer 4, the region near this bed may be less

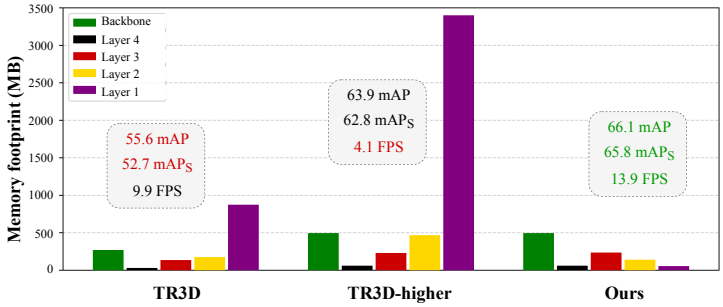

Fig. 4: The memory footprint distribution of different multi-level detectors. Layer 4 to Layer 1 refer to decoder layers (including detection heads) from coarse to fine. If doubling the spatial resolution of TR3D, the performance on 3D small object detection improves from 52.7% to 62.8% while memory footprint increases dramatically. We find decoder layers accounts for most of the costs. DSPDet3D efficiently reduces redundant computation on these layers, achieving both fast speed and high accuracy.

informative for detecting other objects in the follow decoder layers. If we can skip the upsampling operation at these regions, the voxels will be sparsified level by level, as shown in Figure 3 (right). In this way, small objects can be detected in Layer 1 from only a small number of voxels. Inspired by this, we propose to dynamically prune the voxel features according to the distribution of objects.

However, pruning a voxel will not only reduce the number of object proposals in the following levels, but also change the following voxel features computed based on the pruned voxel. Therefore, in order to reduce the redundant computation of multi-level detector without degrading the detection performance, a carefully designed pruning strategy is required. We give theoretical derivation as below.

Problem formulation: For each scene, we denote $\{O\}$ as the set of all objects, $\{O_i\}$ as the set of objects assigned to level i[1] during training, $f_i \in \mathbb{R}^{N\times(3+C)}$ as the voxel features of level i. We aim to prune f_i after detecting out $\{O_i\}$, where the objective is to remove as many voxels as possible while keeping the predictions of $\{O\}\backslash\{O_i\}$ unaffected after the pruning. For each object o_j in level j $(j < i)$, we assume the prediction of it is unaffected if the voxel features at level j near its center $\boldsymbol{c_j}$ are unaffected. We make this assumption because most true positive predictions are from object proposals located at the center of bounding boxes [14, 44]. We denote the expected unaffected neighborhood as $\mathcal{C}_j(\boldsymbol{c_j}, P)$, which means a cube centered at $\boldsymbol{c_j}$ with $P \times P \times P$ voxels at level j. Given the symmetry, P should be odd. Then we formulate the objective of our pruning strategy at level i as:

$$\begin{aligned} &\underset{\mathcal{K}_i}{\text{minimize}} \sum_{x,y,z} M_i[x][y][z], M_i = \textstyle\bigwedge_{j=1}^{i-1} \mathcal{K}_i(\boldsymbol{c_j}), \\ &s.t.\ \forall j < i,\ \mathcal{C}_j(\boldsymbol{c_j}, P) \cap \mathcal{A}_{i,j}(\neg\mathcal{K}_i(\boldsymbol{c_j}) \star f_i) = \varnothing \end{aligned} \tag{1}$$

[1] We adopt the same definition of level as in Figure 3, where level i is finer than level $i+1$.

where $M_i \in \mathbb{R}^N$ is a binary pruning mask sharing the same length with f_i, where 0 indicates removing and 1 indicates keeping during the pruning operation $\star$. $\mathcal{K}_i(\cdot)$ is the generation strategy of pruning mask for each object, which generates a binary pruning mask conditioned on the object center. $\mathcal{A}_{i,j}(f)$ is defined as the *affecting field* of f, which represents the voxels at level j that will be affected by pruning f at level i. Without loss of generality, here we choose only one object at each level for simplicity of presentation.

Overview of problem solving: We solve (1) by mathematical induction. Specifically, for pruning strategy M_i at level i, we first consider how to generate pruning mask $\mathcal{K}_i(\boldsymbol{c_{i-1}})$ to ensure the predictions of $\{O_{i-1}\}$ are unaffected. Then we show that by following our pruning strategy $\mathcal{K}_i$, 'the predictions of $\{O_j\}$ are unaffected' can be derived by 'the predictions of $\{O_{j+1}\}$ are unaffected'.[2]

Solving $\mathcal{K}_i(\boldsymbol{c_{i-1}})$: To make sure $\mathcal{C}_{i-1}(\boldsymbol{c_{i-1}}, P) \cap \mathcal{A}_{i,i-1}(\cdot) = \varnothing$, we need to compute the affecting field of each voxel v_i in level i. Obviously, the upper bound of affecting field of v_i expands in shape of cube with sparse convolution. Assume there are m sparse convolution with stride 1 and kernel x_k $(1 \le k \le m)$ between pruning and generative upsampling in level i, one generative transposed convolution with stride 2 and kernel y, and n sparse convolution with stride 1 and kernel z_k $(1 \le k \le n)$ until detecting out objects in level $i-1$. Then the affecting field from pruning (level i) to detecting (level $i-1$) can be written as:

$$\mathcal{A}_{i,i-1}(v_i) = \mathcal{C}_{i-1}(v_i, aff(\{x_k\}, y, \{z_k\})) \tag{2}$$

where $aff(\{x_k\}, y, \{z_k\})$ is the range of affecting field represented by the kernel sizes, which we will detail in supplementary material. Since the shape of the expected unaffected voxel features is a $P \times P \times P$ cube, $\mathcal{K}_i(\boldsymbol{c_{i-1}})$ can be formulated as:

$$\begin{aligned} \mathcal{K}_i(\boldsymbol{c_{i-1}})[x][y][z] &= \mathbb{I}(2 \cdot |\boldsymbol{x} - \boldsymbol{c_{i-1}}|_\infty \le rS_i) \\ r &= \lceil \frac{P + aff(\{x_k\}, y, \{z_k\}) - 2}{2} \rceil \end{aligned} \tag{3}$$

where S_i is the size of voxel in level i. $\mathbb{I}(\cdot)$ is the indicative function. $\boldsymbol{x} = (x, y, z)$ is the voxel coordinates of f_i.

Recursion of $\mathcal{K}_i$: We now derive when the pruning strategy $\mathcal{K}_i$ in (3) also works for $\boldsymbol{c_j}$ $(j < i-1)$. We can regrad $\boldsymbol{c_j}$ as the center of object in level $i-1$ and use (3) to generate the pruning mask. In this way, $\mathcal{C}_{i-1}(\boldsymbol{c_j}, P)$ are unaffected. As $\mathcal{C}_j(\boldsymbol{c_j}, P)$ is covered by $\mathcal{C}_{i-1}(\boldsymbol{c_j}, P)$, so $\mathcal{C}_j(\boldsymbol{c_j}, P)$ is unaffected as well. We should also ensure pruning in level i has no cumulative impact on pruning in level $i-1$:

$$(\mathcal{K}_{i-1}(\boldsymbol{c_j}) \star f_{i-1}) \subseteq \mathcal{C}_{i-1}(\boldsymbol{c_j}, P) \tag{4}$$

this means when generating pruning mask of $\boldsymbol{c_j}$ in level $i-1$ using $\mathcal{K}_{i-1}$, the kept voxels should be covered by the unaffected voxels after pruning in level i. So we have:

$$r \cdot S_{i-1} \le P \cdot S_{i-1} \tag{5}$$

The minimum P can be acquired by solving (5). In this case, strategy $\mathcal{K}_i$ in (3) works for all $\boldsymbol{c_j}$ $(j < i)$.

[2] We provide illustrated examples in supplementary material for better understanding.

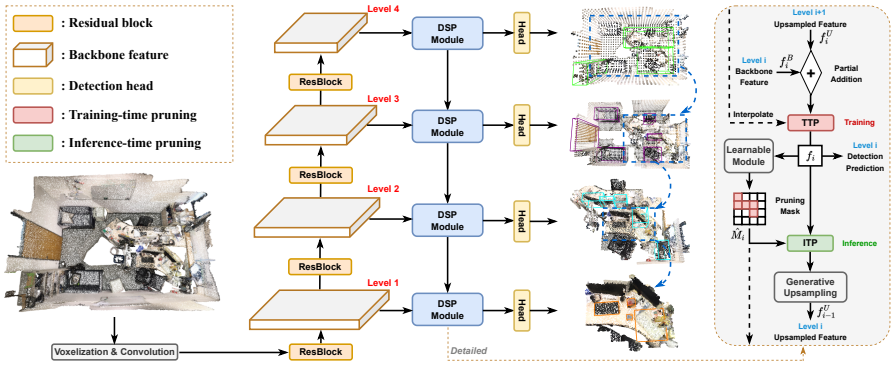

Fig. 5: Illustration of DSPDet3D. The voxelized point clouds are fed into a high-resolution sparse convolutional backbone, which output four levels of scene representations. Four dynamic spatial pruning (DSP) modules are stacked to construct a multi-level decoder and detect objects from coarse to fine. DSP module utilizes a light-weight learnable module to predict the pruning mask. During inference, we discretize the pruning mask and use it to guide pruning before generative upsampling. While during training we interpolate the pruning mask to next level and prune the voxel features after generative upsampling.

3.3 DSPDet3D

Based on the theoretical analysis, we devise a dynamic spatial pruning (DSP) module to approximate the ideal pruning strategy. We further construct a 3D small object detector named DSPDet3D with the proposed DSP module.

DSP module: As shown in Figure 3, we modify the layers of a typical multi-level decoder to DSP modules, which prunes redundant voxel features after detecting out objects at each level for efficient feature upsampling. Formally, given the upsampled voxel feature f_i^U and the backbone feature f_i^B at level i, DSP module first add them for detection. However, f_i^U may be much sparser than f_i^B due to pruning, directly adding by taking union of them is inefficient. Therefore, we propose a new operator called *partial addition* to fit our pruning strategy:

$$f_i = f_i^B \overrightarrow{+} f_i^U \tag{6}$$

where addition is constrained to be operated only on the voxels of f_i^U. Then objects are detected using a shared detection head across all levels: $\{O_i\} = Detect(f_i)$. Once objects at level i are detected out, we prune the voxel features according to the derived strategy described in Section 3.2. Here we devise a light-weight MLP-based learnable pruning module to decide where smaller objects (i.e. objects in level j $(j < i)$) may appear, and then prune other locations:

$$\bar{f}_i = t(\hat{M}_i) \star f_i, \ \hat{M}_i = \mathrm{MLP}_i(f_i) \tag{7}$$

where $\hat{M}_i$ is the pruning mask predicted from f_i, which represents the probability of retention for each voxel. We utlize FocalLoss [23] to supervise $\hat{M}_i$ with the generated

M_i in (1). During inference, a threshold function $t(\cdot)$ sets probability lower than τ to be 0, others be 1 to guide pruning. After pruning, the generative upsampling is applied to acquire features for the next level: $f_{i-1}^{U} = GeConv(\bar{f}_i)$.

During training, as $\hat{M}_i$ may not be so accurate (especially at beginning), we find applying the above learnable pruning module makes training difficult to converge. Instead, we switch the pruning to weak mode for context preservation. As shown in Figure 5, the weak pruning is applied after generative upsampling. For level i, we upsample the pruning mask $\hat{M}_{i+1}$ to level i with nearest neighbor interpolation. Then we sort the interpolated scores and keep only N_{max} voxels with the highest scores. This weak pruning mechanism aims to stablize training, which only works when the amount of voxels is too large to conduct following operations.

Since our theoretical analysis sets the expected unaffected neighborhood to be a $P \times P \times P$ cube, we also modify the assigning strategy of positive object proposals accordingly for robust training. Specifically, for a ground-truth bounding box of o_i assigned to level i, we sample the nearest N_{pos} voxels to $\boldsymbol{c_i}$ inside the cube centered at $\boldsymbol{c_i}$ with length $P \cdot S_i$. If there are less than N_{pos} voxels in the cube, we simply sample all voxels inside it. Our assigning method is independent of the size of bounding box, which ensures there are enough positive proposals even for small objects.

DSPDet3D: Based upon the top-performance multi-level detector TR3D [39], we remove the max pooling layer to increase the spatial resolution of backbone features. Then we replace the decoder in TR3D with four stacked DSP modules to remove redundant voxel features level by level, which achieves efficient upsampling without affecting the detection performance. To train DSPDet3D, we keep the same loss for classification and box regression as in TR3D and add additional FocalLoss to supervise $\hat{M}_i$ with M_i.

Compare with FCAF3D: Similar to our training-time weak pruning, FCAF3D [38] also adopts a pruning strategy in the decoder to prevent the number of voxels from getting too large, which is unable to remove redundant features in early decoder layers during inference. Moreover, it directly utilizes the classification scores for bounding boxes to sort and prune the voxel features, which cannot accurately preserve geometric information for small objects.

4 Experiment

In this section, we conduct experiments to investigate the performance of our approach on 3D small object detection. We first describe the datasets and experimental settings. Then we compare DSPDet3D with the state-of-the-art 3D object detection methods. We also design ablation experiments to study the effectiveness of the proposed methods. Finally we transfer DSPDet3D to extremely large scenes to show its efficiency and generalization ability.

4.1 Experimental Settings

Datasets and metrics: We conduct experiments on two indoor datasets including ScanNet [9] and TO-SCENE [50]. ScanNet is a richly annotated dataset of indoor scenes

Table 1: 3D objects detection results and computational costs of different methods on ScanNet-md40. DSPDet3D with the best pruning threshold is highlighted in gray. We set best scores in bold, runner-ups underlined.

Method	Decoder	mAP		mAP_S		Speed	Memory
		@0.25	@0.5	@0.25	@0.5		
VoteNet	Voting	51.02	33.69	0.30	0	**13.4**	1150
VoteNet_S	Voting	48.62	31.55	1.04	0	8.5	1500
H3DNet	Hybrid	53.51	39.23	3.08	0.90	7.2	1550
GroupFree3D	Transformer	56.77	41.39	11.7	0.81	7.8	1450
GroupFree3D_S	Transformer	29.44	11.94	0.20	0	3.2	2000
RBGNet	Voting	55.23	32.64	5.81	0	6.6	1700
FCAF3D	Multi-level	59.49	48.75	18.38	8.21	12.3	<u>850</u>
CAGroup3D	Voting	60.29	49.90	16.62	8.63	3.1	3250
TR3D	Multi-level	61.59	49.98	27.53	12.91	10.8	1250
FCAF3D-higher	Multi-level	62.65	51.01	27.68	16.23	7.1	4000
TR3D-higher	Multi-level	<u>65.18</u>	54.03	41.70	29.56	5.2	4450
Ours($\tau = 0$)	Multi-level	**65.39**	**54.59**	**44.79**	**31.55**	4.4	4200
Ours($\tau = 0.3$)	Multi-level	65.04	<u>54.35</u>	<u>43.77</u>	<u>30.38</u>	<u>12.5</u>	**700**

with 1201 training scenes and 312 validation scenes. Each object in the scenes are annotated with texts and then mapped to category IDs. We follow the ScanNet-md40 benchmark proposed by [51], which contains objects in 22 categories with large size variance. TO-SCENE is a mixed reality dataset which provides three variants called TO_Vanilla, TO_Crowd and TO_ScanNet with different numbers of tabletop objects and scene scales. We choose the room-scale TO_ScanNet benchmark, which contains 3600 training scenes and 800 validation scenes with 70 categories. However, TO_ScanNet adopts non-uniform sampling to acquire about 2000 points per tabletop object, which is infeasible in practical settings. To this end, we downsample the small objects and control the density of them to be similar with other objects and backgrounds. We name this modified version as TO-SCENE-down benchmark. We take the point clouds without color as inputs for all methods. More details about ScanNet-md40 and TO-SCENE-down benchmarks can be found in supplementary material.

We report the mean average precision (mAP) with threshold 0.25 and 0.5. To measure the performance on different categories, we use two kinds of metrics: mAP and mAP_S, which refer to the mean AP of all objects and of small objects respectively. Here we define categories of small object as ones with average volume smaller than $0.05m^3$ for both benchmarks.

Implementation details: We implement our approach with PyTorch [32], MinkowskiEngine [7] and MMDetection3D [8]. We follow the same training strategy / hyperparameters as TR3D [39] for fair comparison. Training converges within 4 hours on a 4 GPU machine. The stride of the sparse convolution in the preencoder of DSPDet3D is set to 2, thus the voxel size of f_1^B is *4cm* and S_i equals to $2^i \cdot$ *2cm*. We set $N_{pos} = 6$ and $N_{max} = 100000$ during training. The weight of the FocalLoss between M_i and $\hat{M}_i$ is

Table 2: 3D objects detection results and computational costs of different methods on TO-SCENE-down benchmark. DSPDet3D with the best pruning threshold is highlighted in gray. We set best scores in bold, runner-ups underlined.

Method	Decoder	mAP		mAP_S		Speed	Memory
		@0.25	@0.5	@0.25	@0.5		
VoteNet	Voting	26.72	14.01	14.51	4.78	<u>12.8</u>	1300
$VoteNet_S$	Voting	31.87	14.89	21.75	7.40	7.6	1650
H3DNet	Hybrid	27.69	17.38	14.83	7.39	5.1	1650
GroupFree3D	Transformer	32.41	20.43	20.17	10.13	7.7	1700
$GroupFree3D_S$	Transformer	40.14	23.55	33.33	16.15	2.4	2200
RBGNet	Voting	40.42	30.27	29.69	21.61	5.0	1850
FCAF3D	Multi-level	45.13	37.21	37.18	31.65	11.9	<u>1000</u>
CAGroup3D	Voting	54.28	47.58	48.49	43.85	2.2	3500
TR3D	Multi-level	55.58	45.95	52.72	44.01	9.9	1400
FCAF3D-higher	Multi-level	57.23	50.39	53.07	48.76	6.3	4250
TR3D-higher	Multi-level	63.96	56.06	62.84	57.14	4.1	4600
Ours($\tau = 0$)	Multi-level	**66.81**	**59.41**	**66.53**	**61.57**	4.1	5300
Ours($\tau = 0.5$)	Multi-level	<u>66.12</u>	<u>58.55</u>	<u>65.82</u>	<u>60.73</u>	**13.9**	**800**

0.01. In terms of block structure, we have $\{x_k\} = \varnothing$, $y = 3$ and $\{z_k\} = \{3, 3\}$. So we set $r = 7$ and $P = 7$ according to (3).

4.2 Comparison with State-of-the-art

We compare our method with popular and state-of-the-art 3D object detection methods, including VoteNet [33], H3DNet [54], GroupFree3D [26], RBGNet [47], CAGroup3D [46], FCAF3D [38] and TR3D [39]. We also follow [50] to reduce the radius of ball query in the PointNet++ backbone for VoteNet and GroupFree3D. The modified models is distinguished by subscript S. Note that the original TR3D only uses two detection head at level 2/3 and removes the last generative upsampling. However, detecting small objects heavily relies on high-resolution feature map, so we add the upsampling back. This will make it slightly slower but much more accurate on the 3D small object detection benchmarks.

For all methods, we use their official code and the same training strategy / hyperparameters to train them on ScanNet-md40 and TO-SCENE-down.

Table 1 and 2 shows the experimental results on ScanNet-md40 and TO-SCENE-down respectively. Consistent with the observation of [51], we find point-based (VoteNet, H3DNet, RBGNet) and transformer-based (GroupFree3D) methods almost fail to detect small objects on ScanNet-md40. This is because the PointNet++ backbone used by these methods adopts set abstraction (SA) operation to aggressively downsample the point clouds and extract scene representation. Since the number of small objects in ScanNet is limited, furthest point sampling has a low probability to sample points on small objects, which leads to inaccurate representation of small objects. For methods (CAGroup3D, FCAF3D, TR3D) with sparse convolutional backbone, they achieve

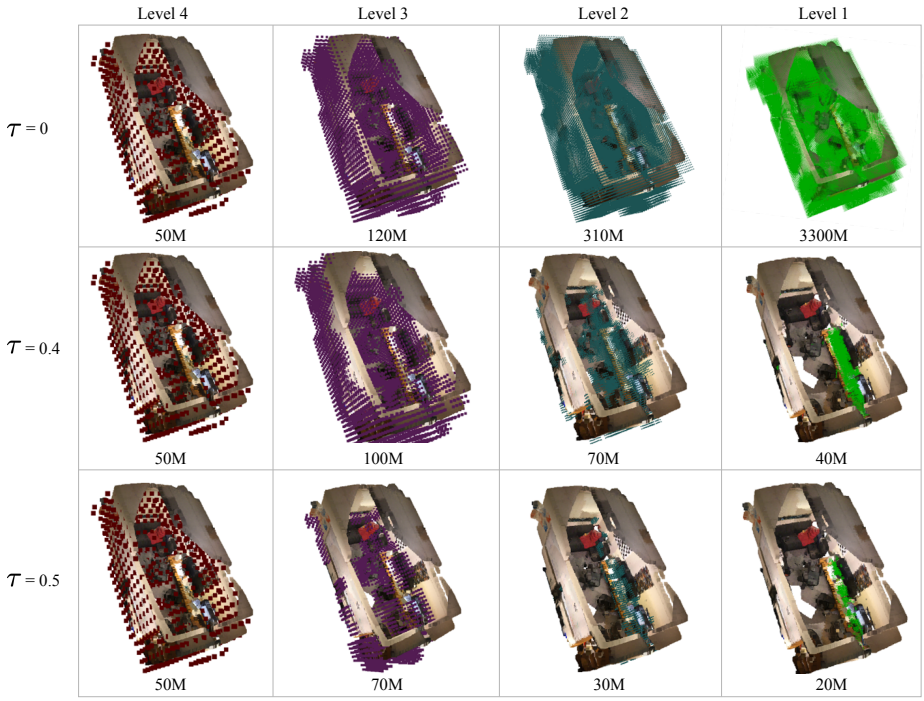

Fig. 6: Visualization of pruning process on ScanNet. We show the kept voxels in each level under different thresholds. The memory footprint of each level is also listed at bottom.

relatively much higher mAP_S due to sparse convolution [7, 13] can extract fine-grained scene representation with high efficiency. However, two-stage method like CAGroup3D is both slow and memory-consuming. Multi-level methods like FCAF3D and TR3D are efficient and get good performance on small object detection due to the FPN-like architecture, but they are still limited by resolution. On the contrary, our DSPDet3D with a proper threshold takes advantage of the high-resolution scene representation to achieve much higher performance. Furthermore, DSPDet3D is the most memory-efficient model among all mainstream methods.

4.3 Ablation Study

We conduct ablation studies on ScanNet-md40 to study the effects of hyperparameters and different design choices.

Pruning process: We visualize the pruning process under different thresholds in Figure 6, where the voxels in each level after pruning are shown. We also list the memory footprint of each level. It can be seen that our method significantly reduce the memory footprint by pruning most of the uninformative voxels. Our pruning module only keeps regions where there are smaller objects than current level.

Hyperparameters: We study two hyperparameters: r and N_{pos}, which is highly relevant to 3D small object detection. Note that $r = \lceil \frac{P+9-2}{2} \rceil$, thus r and P should be

Table 3: Ablation studies on several design choices. We control the speed of each method to 10 FPS and report the accuracy in mAP@0.25 and mAP_S@0.25.

Method	mAP	mAP_S
Remove partial addition	55.3	35.5
Addition by taking union	57.9	36.4
Addition by interpolation	62.1	40.9
Spherical keeping mask	63.0	41.1
Remove training-time pruning	–	–
Positive proposal inside bounding box	62.4	40.7
The full design of DSP module	**65.1**	**44.1**

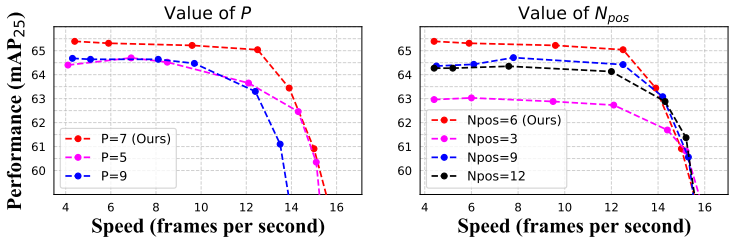

Fig. 7: Ablation studies on the value of r and N_{pos}. For each value we report performance under different pruning threshold τ.

changed simultaneously. As shown in Figure 7 (left), setting $r = 7$ achieves the best performance. If r is smaller than 7 then $r > P$, which conflicts with Equation (5) and the features will be affected by pruning. While a larger r will make the pruning less aggressive, resulting in a large number of redundant voxel features. Figure 7 (right) shows that the number of positive object proposals should be set properly, which is important to balance the ratio between positive and negative samples during classification.

Design choices: We also study the design choices of DSPDet3D in Table 3. Observing the second, third and fourth rows, we conclude that the partial addition is important for efficient feature fusion. Although taking union can preserve more information, this operation will reduce the sparsity of voxels and thus make our pruning less efficient. The fifth row shows that generate the keeping mask according to the shape of affecting field is better than using a spherical shape. According to the sixth row, removing training-time pruning will significantly increase the memory footprint during training, which makes the network unable to train. The seventh row validates the effectiveness of our assigning method for positive object proposals.

4.4 Transferring to Larger Scenes

We further validate the efficiency and generalization ability of different 3D detectors by transferring them to scenes of much larger scale. We first train 3D detectors on rooms from ScanNet training set in a category-agnostic manner, which is done by regarding

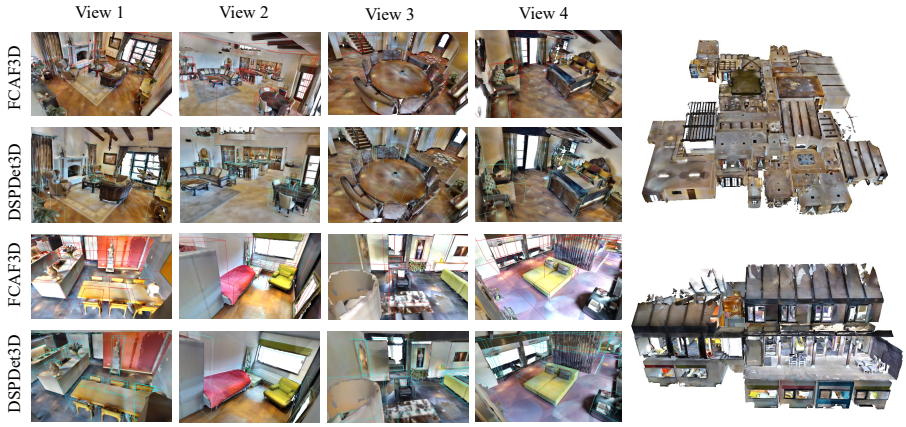

Fig. 8: Visualization of the transferring results of different 3D object detectors. The 3D detector is trained on rooms from ScanNet and directly adopted to process a whole building-level 3D scene from Matterport3D.

every labeled object as the same category. Then we directly adopt them to process the building-level scenes in Matterport3D [4]. We find previous methods almost all fail to process the extremely large scenes due to unaffordable memory footprint, so we only compare DSPDet3D with FCAF3D as shown in 8. It is shown that FCAF3D cannot detect out any small object and even struggles on relatively large objects like chairs when the scene is too large. On the contrary, DSPDet3D is able to accurately detect small objects like cups and thin pictures.

5 Conclusion

In this paper, we have presented an efficient feature pruning strategy for 3D small object detection. Inspired by the fact that small objects only occupy a small proportion of space, we adopt a multi-level detection framework to detect different sizes of objects in different levels. Then we present a dynamic spatial pruning strategy to prune the voxel features after detecting out objects in each level. Specifically, we first design the dynamic spatial pruning strategy by theoretical analysis on how to prune voxels without affecting the features of object proposals. Then we propose dynamic spatial pruning (DSP) module according to the strategy and use it to construct DSPDet3D. Extensive experiments on ScanNet and TO-SCENE datasets show that our DSPDet3D achieves leading detection accuracy and speed. We also conduct transferring experiment on Matterport3D to show DSPDet3D also generalizes well to extremely large scenes.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grant 62125603, Grant 62321005, and Grant 62336004.

References

1. Armeni, I., Sener, O., Zamir, A.R., Jiang, H., Brilakis, I., Fischer, M., Savarese, S.: 3d semantic parsing of large-scale indoor spaces. In: ICCV. pp. 1534–1543 (2016) 2
2. Bansal, M., Krizhevsky, A., Ogale, A.: Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. arXiv preprint arXiv:1812.03079 (2018) 1
3. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: ECCV. pp. 213–229. Springer (2020) 3
4. Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., Song, S., Zeng, A., Zhang, Y.: Matterport3d: Learning from rgb-d data in indoor environments. 3DV (2017) 2, 14
5. Chen, C., Liu, M.Y., Tuzel, O., Xiao, J.: R-cnn for small object detection. In: ACCV. pp. 214–230. Springer (2017) 4
6. Cheng, B., Sheng, L., Shi, S., Yang, M., Xu, D.: Back-tracing representative points for voting-based 3d object detection in point clouds. In: CVPR. pp. 8963–8972 (2021) 3
7. Choy, C., Gwak, J., Savarese, S.: 4d spatio-temporal convnets: Minkowski convolutional neural networks. In: CVPR. pp. 3075–3084 (2019) 1, 3, 4, 10, 12
8. Contributors, M.: Mmdetection3d: Openmmlab next-generation platform for general 3d object detection (2020) 10
9. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: CVPR. pp. 5828—-5839 (2017) 2, 3, 9
10. Deng, C., Wang, M., Liu, L., Liu, Y., Jiang, Y.: Extended feature pyramid network for small object detection. TMM **24**, 1968–1979 (2021) 4
11. Gao, M., Yu, R., Li, A., Morariu, V.I., Davis, L.S.: Dynamic zoom-in network for fast object detection in large images. In: CVPR. pp. 6926–6935 (2018) 4
12. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: CVPR. pp. 3354–3361 (2012) 2
13. Graham, B., Engelcke, M., Van Der Maaten, L.: 3d semantic segmentation with submanifold sparse convolutional networks. In: CVPR. pp. 9224–9232 (2018) 1, 3, 4, 12
14. Gwak, J., Choy, C., Savarese, S.: Generative sparse detection networks for 3d single-shot object detection. In: ECCV. pp. 297–313. Springer (2020) 3, 4, 6
15. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. NeurIPS **28** (2015) 4
16. Huang, Z., Wang, N.: Data-driven sparse structure selection for deep neural networks. In: ECCV. pp. 304–320 (2018) 4
17. Kisantal, M., Wojna, Z., Murawski, J., Naruniec, J., Cho, K.: Augmentation for small object detection. arXiv preprint arXiv:1902.07296 (2019) 3
18. LeCun, Y., Denker, J., Solla, S.: Optimal brain damage. NeurIPS **2** (1989) 4
19. Lee, J., Choy, C., Park, J.: Putting 3d spatially sparse networks on a diet. arXiv preprint arXiv:2112.01316 (2021) 3
20. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710 (2016) 4
21. Li, J., Liang, X., Wei, Y., Xu, T., Feng, J., Yan, S.: Perceptual generative adversarial networks for small object detection. In: CVPR. pp. 1222–1230 (2017) 4
22. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: CVPR. pp. 2117–2125 (2017) 4
23. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV. pp. 2980–2988 (2017) 8

24. Liu, J., Chen, Y., Ye, X., Tian, Z., Tan, X., Qi, X.: Spatial pruned sparse convolution for efficient 3d object detection. In: NeurIPS (2022) 4
25. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: ECCV. pp. 21–37 (2016) 3
26. Liu, Z., Zhang, Z., Cao, Y., Hu, H., Tong, X.: Group-free 3d object detection via transformers. arXiv preprint arXiv:2104.00678 (2021) 3, 11
27. Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C.: Learning efficient convolutional networks through network slimming. In: ICCV. pp. 2736–2744 (2017) 4
28. Misra, I., Girdhar, R., Joulin, A.: An end-to-end transformer model for 3d object detection. In: ICCV. pp. 2906–2917 (2021) 3
29. Molchanov, P., Tyree, S., Karras, T., Aila, T., Kautz, J.: Pruning convolutional neural networks for resource efficient inference. arXiv preprint arXiv:1611.06440 (2016) 4
30. Mousavian, A., Eppner, C., Fox, D.: 6-dof graspnet: Variational grasp generation for object manipulation. In: ICCV. pp. 2901–2910 (2019) 1
31. Najibi, M., Singh, B., Davis, L.S.: Autofocus: Efficient multi-scale inference. In: ICCV. pp. 9745–9755 (2019) 4
32. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. NeurIPS **32** (2019) 10
33. Qi, C.R., Litany, O., He, K., Guibas, L.J.: Deep hough voting for 3d object detection in point clouds. In: ICCV. pp. 9277–9286 (2019) 3, 11
34. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: CVPR. pp. 652–660 (2017) 1, 3
35. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: NeurIPS. pp. 5099–5108 (2017) 1, 3
36. Rao, Y., Zhao, W., Liu, B., Lu, J., Zhou, J., Hsieh, C.J.: Dynamicvit: Efficient vision transformers with dynamic token sparsification. NeurIPS **34**, 13937–13949 (2021) 4
37. Rozenberszki, D., Litany, O., Dai, A.: Language-grounded indoor 3d semantic segmentation in the wild. In: ECCV. pp. 125–141. Springer (2022) 2
38. Rukhovich, D., Vorontsova, A., Konushin, A.: Fcaf3d: fully convolutional anchor-free 3d object detection. In: ECCV. pp. 477–493. Springer (2022) 3, 4, 9, 11
39. Rukhovich, D., Vorontsova, A., Konushin, A.: Tr3d: Towards real-time indoor 3d object detection. arXiv preprint arXiv:2302.02858 (2023) 1, 3, 4, 5, 9, 10, 11
40. Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., Li, H.: Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In: CVPR. pp. 10529–10538 (2020) 1
41. Singh, B., Davis, L.S.: An analysis of scale invariance in object detection snip. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3578–3587 (2018) 4
42. Singh, B., Najibi, M., Davis, L.S.: Sniper: Efficient multi-scale training. NeurIPS **31** (2018) 4
43. Song, S., Lichtenberg, S.P., Xiao, J.: Sun rgb-d: A rgb-d scene understanding benchmark suite. In: CVPR. pp. 567–576 (2015) 2
44. Tian, Z., Shen, C., Chen, H., He, T.: Fcos: Fully convolutional one-stage object detection. In: ICCV. pp. 9627–9636 (2019) 4, 6
45. Tong, K., Wu, Y., Zhou, F.: Recent advances in small object detection based on deep learning: A review. IVC **97**, 103910 (2020) 3
46. Wang, H., Ding, L., Dong, S., Shi, S., Li, A., Li, J., Li, Z., Wang, L.: Cagroup3d: Class-aware grouping for 3d object detection on point clouds. arXiv preprint arXiv:2210.04264 (2022) 1, 3, 11
47. Wang, H., Shi, S., Yang, Z., Fang, R., Qian, Q., Li, H., Schiele, B., Wang, L.: Rbgnet: Ray-based grouping for 3d object detection. In: CVPR. pp. 1110–1119 (2022) 3, 11

48. Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., et al.: Deep high-resolution representation learning for visual recognition. TPAMI **43**(10), 3349–3364 (2020) 4
49. Xie, Q., Lai, Y.K., Wu, J., Wang, Z., Zhang, Y., Xu, K., Wang, J.: Mlcvnet: Multi-level context votenet for 3d object detection. In: CVPR. pp. 10447–10456 (2020) 3
50. Xu, M., Chen, P., Liu, H., Han, X.: To-scene: A large-scale dataset for understanding 3d tabletop scenes. In: ECCV. pp. 340–356. Springer (2022) 2, 3, 4, 9, 11
51. Xu, X., Wang, Y., Zheng, Y., Rao, Y., Zhou, J., Lu, J.: Back to reality: Weakly-supervised 3d object detection with shape-guided label enhancement. In: CVPR. pp. 8438–8447 (2022) 2, 4, 10, 11
52. Xu, X., Wang, Z., Zhou, J., Lu, J.: Binarizing sparse convolutional networks for efficient point cloud analysis. arXiv preprint arXiv:2303.15493 (2023) 3
53. Yang, C., Huang, Z., Wang, N.: Querydet: Cascaded sparse query for accelerating high-resolution small object detection. In: CVPR. pp. 13668–13677 (2022) 4
54. Zhang, Z., Sun, B., Yang, H., Huang, Q.: H3dnet: 3d object detection using hybrid geometric primitives. In: ECCV. pp. 311–329 (2020) 3, 11
55. Zhao, T., Ning, X., Hong, K., Qiu, Z., Lu, P., Zhao, Y., Zhang, L., Zhou, L., Dai, G., Yang, H., et al.: Ada3d: Exploiting the spatial redundancy with adaptive inference for efficient 3d object detection. arXiv preprint arXiv:2307.08209 (2023) 4
56. Zheng, W., Tang, W., Jiang, L., Fu, C.W.: Se-ssd: Self-ensembling single-stage object detector from point cloud. In: CVPR. pp. 14494–14503 (2021) 1
57. Zhu, Y., Mottaghi, R., Kolve, E., Lim, J.J., Gupta, A., Fei-Fei, L., Farhadi, A.: Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: ICRA. pp. 3357–3364 (2017) 1
58. Zoph, B., Cubuk, E.D., Ghiasi, G., Lin, T.Y., Shlens, J., Le, Q.V.: Learning data augmentation strategies for object detection. In: ECCV. pp. 566–583. Springer (2020) 3