DSPDet3D: Dynamic Spatial Pruning for 3D Small Object Detection

Xiuwei Xu[1], Zhihao Sun[2], Ziwei Wang[1], Hongmin Liu[2] Jie Zhou[1], Jiwen Lu[1]*
[1]Tsinghua University [2]University of Science and Technology Beijing
{xxw21, wang-zw18}@mails.tsinghua.edu.cn; d202210361@xs.ustb.edu.cn;
hmliu@ustb.edu.cn; {jzhou, lujiwen}@tsinghua.edu.cn

Abstract: Fine-grained 3D object detection is a core ability for agents to understand their 3D environment and interact with surrounding objects. However, current methods and benchmarks mainly focus on relatively large stuff. 3D object detectors still struggle on small objects due to weak geometric information. With in-depth study, we find increasing the spatial resolution of the feature maps significantly boosts the performance of 3D small object detection. And more interestingly, though the computational overhead increases dramatically with resolution, the growth mainly comes from the upsampling operation of the decoder. Inspired by this, we present a high-resolution multi-level detector with dynamic spatial pruning named DSPDet3D, which detects objects from large to small by iterative upsampling and meanwhile prunes the spatial representation of the scene at regions where there is no smaller object to be detected in higher resolution. We organize two benchmarks on ScanNet and TO-SCENE dataset to evaluate the ability of fine-grained 3D object detection, where our DSPDet3D improves the detection performance of small objects to a new level while achieving leading inference speed compared with existing 3D object detection methods. Moreover, DSPDet3D trained with only ScanNet rooms can generalize well to scenes in larger scale. It takes less than 2s for DSPDet3D to directly process a whole house or building consisting of dozens of rooms while detecting out almost all objects, ranging from bottles to beds, on a single RTX 3090 GPU. Project page.

Keywords: 3D scene understanding, generalizable and fine-grained detection

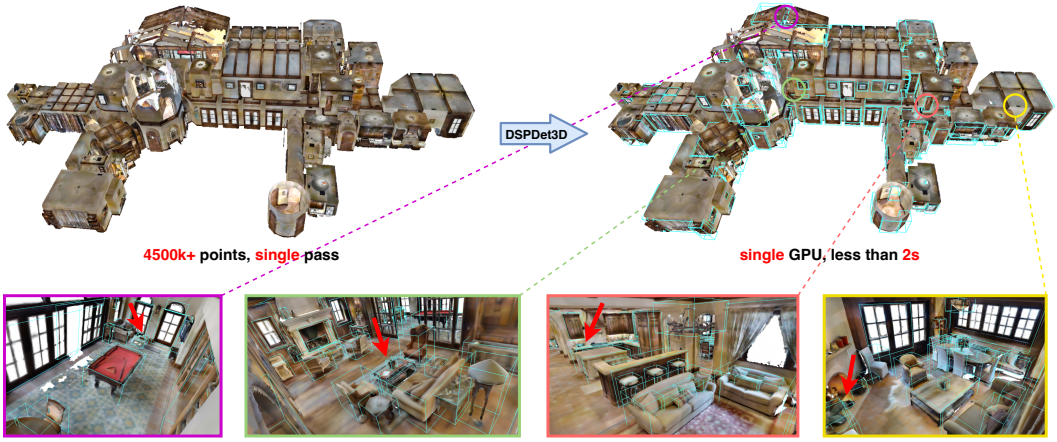


Figure 1: Trained with only rooms from ScanNet, our DSPDet3D generalizes well to process a whole house with dozens of rooms. It takes less than 2s to generate fine-grained detection results with a single GPU.

1 Introduction

3D object detection is a fundamental scene understanding problem, which plays an important role in robotic perception, navigation and manipulation. With the recent advances of deep learning techniques on point cloud understanding [1, 2, 3, 4], 3D detection methods have shown remarkable

*Corresponding author.

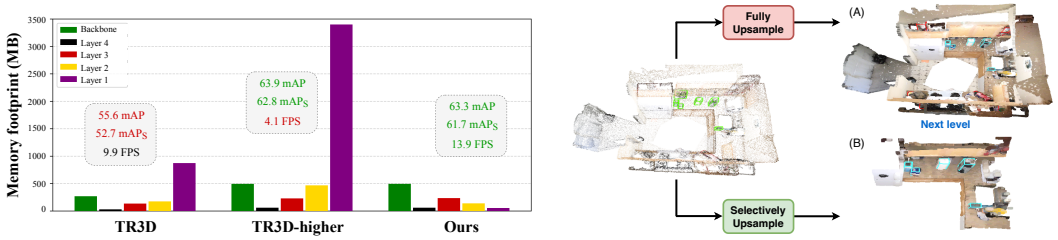
progress [5, 6, 7, 8]. However, small object detection still remains a huge challenge for 3D. In autonomous driving scenarios [9], we observe a significant performance gap between cars and pedestrians. In indoor scenes [10, 11] where the size variance is much larger (e.g. a bed is 1000x larger than a cup), things are only going to get worse. We focus on the more challenging indoor 3D small object detection task where the 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 furniture-level benchmarks [10, 12, 13], they are still far from practical application due to the limited range of object size they can handle. For example, it is hard for previous methods to detect small tabletop objects [14], which limits the agent to execute instruction like *'find and bring a cup of coffee'*. 3D small object detection is very challenging 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. More recently, [15] proposes a tabletop-aware learning strategy to boost the performance of 3D small object detection. However, this method relies on densely sampled points from small objects (about 2000 points per tabletop object, even more than the points of a furniture), which is infeasible in practical scenarios where the points from small objects are very sparse. Therefore, directly detecting small objects from naturally sampled point clouds is a key problem in 3D object detection.

In this paper, we propose DSPDet3D with dynamic spatial pruning for small object detection. We show using multi-level FPN-like [16] detection framework and increasing the spatial resolution of the feature maps significantly boosts the performance of 3D small object detection. Although the additional computational overhead brought by higher resolution is unaffordable, we find the computation growth is imbalanced: the increased memory footprint mainly comes from the huge number of features generated by the upsampling operation of the detector. As the detector only needs to predict sparse bounding boxes and the prediction follows a coarse-to-fine manner, there is a large amount of useless features at regions where there is no object to be detected in higher resolution. Inspired by this, we construct a high-resolution multi-level 3D detector with sparse convolution, which detects objects from large to small. To enable detecting small objects in high resolution with controllable computational cost, we devise a dynamic spatial pruning (DSP) block to generate object proposals and predict the locations where smaller objects in higher resolution are assigned. Upsampling is constrained to these locations by spatial pruning in a cascaded manner at inference time, while during training we switch the pruning to weak mode for context preservation. To demonstrate the effectiveness of DSPDet3D, we first organize two benchmarks on ScanNet and TO-SCENE datasets following [14, 15] to evaluate the ability of fine-grained 3D object detection and then validate the transferring ability of DSPDet3D on larger scale of scenes from Matterport3D [11]. While previous methods struggle on room-level 3D small object detection, DSPDet3D (trained on rooms) can further perform fine-grained detection on house/building-level scenes due to the high generalization ability and efficiency of the proposed dynamic spatial pruning method, as shown in Figure 1.

2 Related Work

Indoor 3D object detection: Thanks to PointNet and PointNet++ [1, 2], 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 [17, 18, 19, 20, 21], transformer-based [22, 23] and voxel-based [24, 25, 7, 8] methods. Inspired by 2D hough voting, VoteNet [17] 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 [26] in 2D domain, GroupFree3D [23] and 3DETR [22] adopts transformer architecture to extract features and decode the object proposals into 3D boxes. As extracting point features require time-consuming sampling and aggregation operation, GSDN [24] proposes a fully convolutional detection network based on sparse convolution [3, 4, 27, 28], which achieves much faster speed. FCAF3D [25] further improves the performance of GSDN with a simple anchor-free architecture. By modifying suboptimal design in FCAF3D, TR3D [8] achieves the state-of-the-art performance with even faster speed. Our method also adopts voxel-based architecture considering its efficiency



(a) Resolution and memory footprint (b) Useless computation

Figure 2: Our observation about 3D small object detection. (a) We find multi-level FPN-like architecture performs best for small object detection. By increasing the resolution of the backbone features, this kind of methods can achieve satisfactory accuracy on 3D small object detection. However, the computational overhead brought by higher resolution is unaffordable, especially in the last upsampling layer. (b) If we only focus on the detection results in a certain level, there are a large amount of useless computation during upsampling.

and scalability. Recently, [29] proposes a spatial pruned convolution for autonomous driving, which supresses the activation on background voxels in sparse convolution layer. Though making sparse convolution more efficient, this method cannot reduce the number of voxels and do not consider size variance of objects, which makes it hard to handle 3D small object detection problem.

Small object detection: Small object detection [30] 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 [31, 32, 33]; (2) scale-aware training and inference strategy using image pyramids [34, 35, 36, 37]; (3) increasing the resolution of features or generating high-resolution features [38, 16, 39, 40, 41, 42]. However, there are far less works about 3D small object detection due to the limit of data and network capability. BackToReality [14] proposes ScanNet-md40 benchmark which contains small objects and finds many current methods suffer a lot in small object detection. TO-SCENE [15] proposes a new dataset and learning strategy for understanding 3D tabletop scenes. Based on these efforts, we conduct in-depth study on the problem of 3D small object detection and propose an efficient framework to achieve both high accuracy and fast inference speed.

3 Approach

In this section, we describe our DSPDet3D for 3D small object detection. We first summarize the key factors to improve the performance of 3D small object detection. Then we show the overall framework of DSPDet3D. Finally we detail the training process for our dynamic spatial pruning module.

3.1 Analysis

We first explore the most suitable architecture for detecting objects with large variance of size. Following the choice of current top-performance 3D detectors [25, 7, 8], we fix sparse convolutional network (MinkResNet [4]) as the backbone and study the effects of different decoders on the performance of small object detection. By comparing two-stage decoder [7] and multi-level FPN-like [16] decoder [8], we find the latter achieves better performance on both accuracy and speed (more detail can be found in Table 1). This is because the multi-level architecture efficiently utilizes the middle feature maps by detecting objects of different sizes under different resolution.

However, existing multi-level methods [24, 25, 8] need to quantize the point clouds of a scene into tiny voxels such as *1cm* for information preservation and then apply aggressive downsampling in backbone to control the computational cost, which harms the representation of small objects. Taking TR3D [8] for example, we enhance the features of small objects by doubling the spatial resolution of the backbone features and observe that the performance on small objects improves from 52.7% to 62.8%, which validates the importance of increasing spatial resolution for small object detection. Although the additional computational overhead brought by higher resolution limits the application of the model, we find the computation growth is imbalanced. According to Figure 2a,

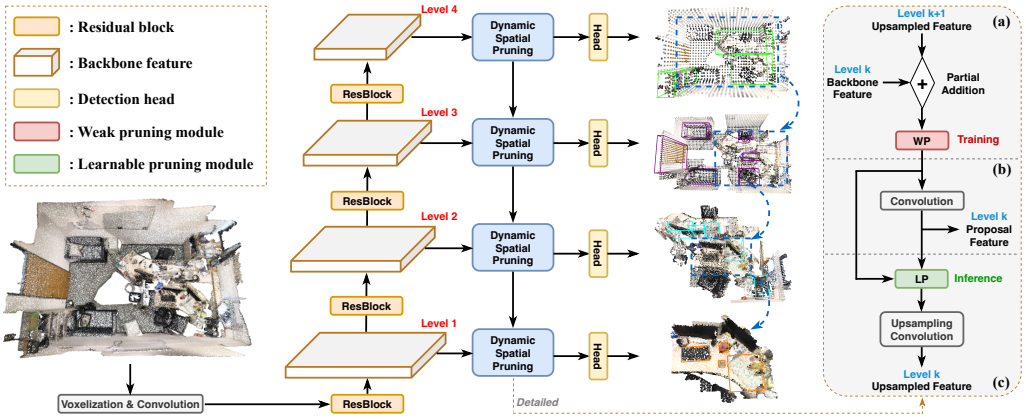


Figure 3: Illustration of DSPDet3D. The voxelized point clouds are fed into a high-resolution sparse convolutional backbone, which output four levels of scene representations. Four dynamic spatial pruning (DSP) blocks are stacked to construct a multi-level FPN-like decoder and detect objects from coarse to fine. Each DSP block merges the backbone and upsampled features, generates object proposals for detection and selectively upsamples the feature map by pruning uninformative voxels. During training, we switch the pruning to weak mode for context preservation. We detail DSP block on the right. Note that the part (a) and (c) of DSP are absent in level 4 and level 1 respectively.

the upsampling layers (including detection heads) account for the most memory footprint and have larger memory growth ratio than the backbone.

Since the detector only needs to predict sparse bounding boxes, we assume there is a large amount of useless computation in upsampling layers. For instance, backgrounds like wall/floor and regions where there is no object to be detected in higher resolution are less informative and may not need to be upsampled. As shown in Figure 2b, even if we only upsample the regions which contain objects in the next level, the detection results in this selectively upsampled scene is as good as in the fully upsampled scene. Therefore, if we can remove the useless computation on uninformative regions during upsampling, the memory footprint and inference time will be significantly reduced without performance degradation.

In conclusion, we summarize three key points for designing an effective and efficient 3D detector for small object detection: (1) multi-level FPN-like architecture; (2) increasing the spatial resolution; (3) removing the useless computation in upsampling layers.

3.2 Overall Framework

We show the overall framework of DSPDet3D in Figure 3. A high-resolution backbone is first utilized to extract feature maps for each level. Then we iteratively apply DSP block to generate object proposals for current level and selectively upsample the feature map to next level with higher resolution. Below we describe each part of our approach.

High-resolution backbone: As shown in Figure 2a, increasing the resolution of backbone will significantly boost the performance on 3D small object detection and the memory footprint growth on backbone itself is acceptable. Motivated by this, we design a high-resolution backbone which consists of a preencoder and a ResNet34 [43] backbone implemented with 3D sparse convolution [3, 4]. Different from previous sparse convolution-based methods [24, 25, 8], we remove the max pooling layer to increase the spatial resolution of backbone features. Given an input point cloud scene, we first quantize it into sparse voxels with *1cm* size and apply one sparse convolution to extract the initial feature map F_0^B. Then we adopt four residual blocks to successively extract the backbone features of different levels:

$$F_{i+1}^B = \text{ResBlock}_i(F_i^B),\ i = 0, 1, 2, 3 \tag{1}$$

where each ResBlock downsamples the feature map with stride 2.

Decoder with dynamic spatial pruning: The decoder of DSPDet3D aims to predict bounding boxes for each level, which consists of four DSP blocks and a shared detection head at each level. DSP block regards each voxel of the feature map as an object proposal and iteratively upsamples the feature map to generate object proposals for each level. By pruning uninformative voxels, the upsampling operation is constrained to regions where there are smaller objects to be detected in the following levels. The detection head predicts box regression and classification score for each proposal. Heads at different levels are expected to predict objects in different sizes: the higher the level, the larger objects the head detects. Then predictions from four levels are fused by 3D NMS as the final output. Below we detail how DSP block works.

DSP block first merges the backbone feature and the upsampled feature from previous DSP block:

$$F_i^M = F_i^B \vec{+} F_{i+1}^U, i = 3, 2, 1 \tag{2}$$

where $\vec{+}$ is a new operation called *partial addition* proposed by us. Since the upsampled feature F_{i+1}^U may not share the same voxels with F_i^B due to spatial pruning, we constrain the addition to be operated on the voxels of F_{i+1}^U to keep the spatial sparsity. We detail the design choice in Section 4. For DSP block in level 4, we directly set $F_4^M = F_4^B$. Given the merged feature F_i^M, we apply sparse convolution on each voxel to further aggregate features, which is then outputted as the object proposals in level i: $F_i^P = \mathrm{SparseConv}(F_i^M)$. F_i^P is also upsampled to generate high-resolution feature map for level $i-1$ to detect smaller objects.

As we adopt a high-resolution backbone and the upsampling convolution [24] will generate much more voxels than the corresponding backbone feature map, we should remove useless computation in upsampling as much as possible. Motivated by the irregular representation of 3D sparse tensor, we propose to conduct dynamic pruning on the feature map, which directly remove voxels that are uninformative for detecting objects in the following levels. We name it *spatial pruning* as it is different from previous pruning methods conducted on the network parameters [44, 45, 46]. Here we devise a light-weight learnable pruning module to decide where smaller objects (i.e. objects in level j ($j < i$)) may appear and prune other locations:

$$\bar{F_i^P} = \mathrm{LP}(F_i^P, \hat{K}_i),\ \hat{K}_i = \mathrm{MLP}_i(F_i^M), i = 4, 3, 2 \tag{3}$$

where $\hat{K}_i$ is the keeping mask predicted from F_i^M, which represents the probability of retention for each voxel. During inference, the learnable pruning module LP removes the voxels with probability lower than τ. Different from zeroing out pixels in 2D case, our spatial pruning can reduce both storage and computation costs as the 3D sparse tensor only stores valid voxels and sparse convolution only operates at these locations. Our method is also more flexible than the zoom-in strategy [36, 37], as we can prune voxels into any shape while the zoom-in can only crop out a single rectangle on the image.

Training-time pruning: During training, we do not apply the learnable pruning module as it damages the structural information of the scene and makes training difficult (especially at beginning). Instead, we switch the pruning to weak mode for context preservation, which will remain the voxels unless the amount of them is too large. As shown in Figure 3, the weak pruning module is applied after the partial addition in level 3/2/1. For level i, we upsample the keeping mask $\hat{K_{i+1}}$ to the voxels of level i with nearest neighbor interpolation. Then we sort the interpolated scores and keep N_{max} voxels with the highest scores. Other voxels are pruned as in inference time.

Now we discuss the reason why the pruning module is applied at different locations during training and inference. At inference time we aim to reduce the computation of upsampling, so the pruning module needs to keep as less voxels as possible. By applying pruning after the proposal features, we are able to prune more aggressively as only objects in the following levels need to be considered. While during training, we aim to preserve geometric information. So we apply pruning after the upsampling only if the amount of voxels is too large to conduct following operations.

3.3 Training

Ground-truth bounding boxes are used to supervise the training of DSPDet3D, where the predicted bounding boxes and keeping masks are involved into the computation of loss. The overall loss is:

$$L_{all} = \sum_{i=1}^{4} L_i^{box} + \lambda \sum_{i=2}^{4} L_i^{keep} \tag{4}$$

where L_i^{box} and L_i^{keep} refer to the loss for boxes and keep mask in level i. To calculate the losses, we first assign ground-truth bounding boxes to each level according to the volume, where larger objects are assigned to higher levels. In order to reduce hyperparameters, we set the volume thresholds based on the sizes of sparse voxels. We define the voxel size and minimum volume of box in level i as S_i and V_i respectively. V_i can be computed by:

$$V_i = N_{vol} \cdot S_i^3,\ S_i = 2S_{i-1}, i = 2, 3, 4 \tag{5}$$

where N_{vol} is a hyperparameter to control the size. We set $V_1 = 0$ and $V_5 = +\infty$, then box with volume falling into $[V_i, V_{i+1})$ is assigned to level i. Compared with previous assigning method based on the number of inside voxels [25] or object category [8], our assignment is independent of the input scene or labeled categories, which is robust across datasets and strictly ensures objects in higher levels have larger sizes.

For L^{box}, we keep the training of classification and regression as same as in TR3D. For L^{keep}, we use FocalLoss [47] to supervise $\hat{K}_i$ with a generated ground-truth K_i: K_i is a 0-1 mask where 1 indicates the voxels that should be kept, i.e., the coarse locations of all objects in lower level j ($j < i$). As too much kept voxels leads to increasing computational cost while too few results in poor context information, the ground-truth should be generated with careful design. Here we propose to set K_i according to the *receptive field* of object proposals near ground-truth object centers. By keeping all voxels in the receptive field of an object proposal, the prediction from this proposal will not be affected by pruning. Since the upper bound of receptive field of a voxel expands in shape of cube with sparse convolution, we utilize a cube range to decide where to set 1 in K_i. Specifically, we first set K_i all to zero. Then for each object o in level j ($j < i$), we use its center $c_j^o = (x_j^o, y_j^o, z_j^o)$ to define a cube with side length proportional to the receptive field and set K_i to 1 in this area:

$$K_i[x][y][z] = 1, if\ 2 \cdot \max\{|x - x_j^o|, |y - y_j^o|, |z - z_j^o|\} < rS_j \tag{6}$$

where (x, y, z) is the voxel coordinates of K_i and r is set according to the size of receptive field. We analyze the choice of r in supplementary material.

4 Experiment

In this section, we conduct experiments to investigate the performance of our approach on 3D small object detection. We first describe our benchmarks and experimental settings. Then we compare DSPDet3D with the mainstream 3D object detection methods. We also visualize the pruning process of DSPDet3D for better understanding. Next we train DSPDet3D in a category-agnostic way and transfer it to large scale building to further demonstrate its usefulness. Finally we discuss the limitations of our approach.

4.1 Experimental Settings

Datasets and metrics: We conduct experiments on two indoor datasets including ScanNet [10] and TO-SCENE [15]. ScanNet is a richly annotated dataset of indoor scenes 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 [14], 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

Table 1: 3D objects detection results and computational costs of different methods on ScanNet-md40 and TO-SCENE-down benchmark. We report mAP@0.25 for mAP, mAP_S and mAP_O as well as the speed (FPS) and memory footprint (MB). DSPDet3D with the best pruning threshold is highlighted in gray.

Method	ScanNet-md40					TO-SCENE-down				
	mAP	mAP_S	mAP_O	Speed	Memory	mAP	mAP_S	mAP_O	Speed	Memory
VoteNet	51.02	0.30	62.27	13.4	1150	26.72	14.51	61.94	12.8	1300
VoteNet_S	48.62	1.04	59.17	8.5	1500	31.87	21.75	61.08	7.6	1650
H3DNet	53.51	3.08	64.70	7.2	1550	27.69	14.83	64.97	5.1	1650
GroupFree3D	56.77	11.7	66.82	7.8	1450	32.41	20.17	67.64	7.7	1700
GroupFree3D_S	29.44	0.20	35.89	3.2	2000	40.14	33.33	59.74	2.4	2200
RBGNet	55.23	5.81	66.18	6.6	1700	40.42	29.69	71.39	5.0	1850
CAGroup3D	60.29	16.62	70.01	3.1	3250	54.28	48.49	70.98	2.2	3500
FCAF3D	59.49	18.38	68.64	12.3	850	45.13	37.18	67.92	11.9	1000
TR3D	61.59	27.53	69.16	10.8	1250	55.58	52.72	63.98	9.9	1400
Ours($\tau = 0$)	65.25	44.16	69.94	4.2	4200	63.67	62.05	68.34	4.1	5300
Ours($\tau = 0.3/0.5$)	64.39	41.32	69.52	12.7	700	63.26	61.73	67.69	13.9	600

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.

We report the mean average precision (mAP) with threshold 0.25. To measure the performance on different categories, we use three kinds of metrics: mAP, mAP_S and mAP_O, which refer to the mean AP of all objects, small objects and others respectively. Here we define categories of small object as ones with average volume smaller than $0.05m^3$ for both benchmarks. More details about ScanNet-md40 and TO-SCENE-down benchmarks and more quantitative results can be found in supplementary material.

Implementation details: We implement our approach with PyTorch [48] and MinkowskiEngine [4]. We set max epoch as 12, weight decay as 0.0001 and initial learning rate as 0.001. We use AdamW with a stepwise scheduler to optimize the network parameters, which steps at 8 and 11 epoch to reduce the learning rate by a factor of 10. 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_0^B is *2cm* and S_i equals to $2^i \cdot$ *2cm*. We set $\lambda = 0.01$, $N_{vol} = 27$, $r = 13$ and $N_{max} = 100000$ during training. The channel number of F_0^B and F_i^B $(i = 1, 2, 3, 4)$ are set to 64 and 128 respectively. All sparse convolutional layers with stride 1 have kernel size 3, while layers with stride 2 have kernel size 1 for downsampling and 3 for upsampling.

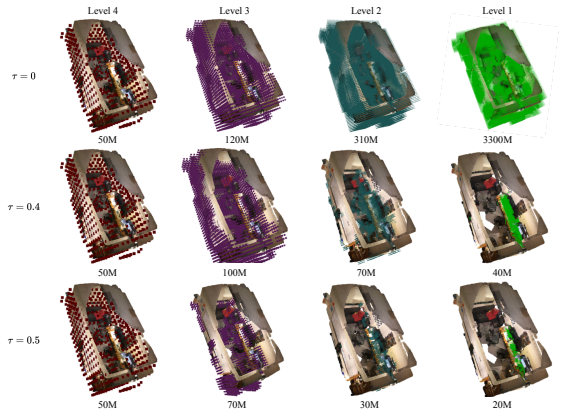


Figure 4: Visualization of pruning process on ScanNet. We show the kept voxels in each level under different thresholds. The memory footprint of each level is also listed at bottom.

4.2 Benchmark Evaluation

We compare our method with popular and state-of-the-art 3D object detection methods, including VoteNet [17], H3DNet [19], GroupFree3D [23], RBGNet [21], CAGroup3D [7], FCAF3D [25] and TR3D [8]. We also follow [15] to reduce the radius of ball query in the PointNet++ backbone for VoteNet and GroupFree3D. The modified models is distinguished by subscript S.

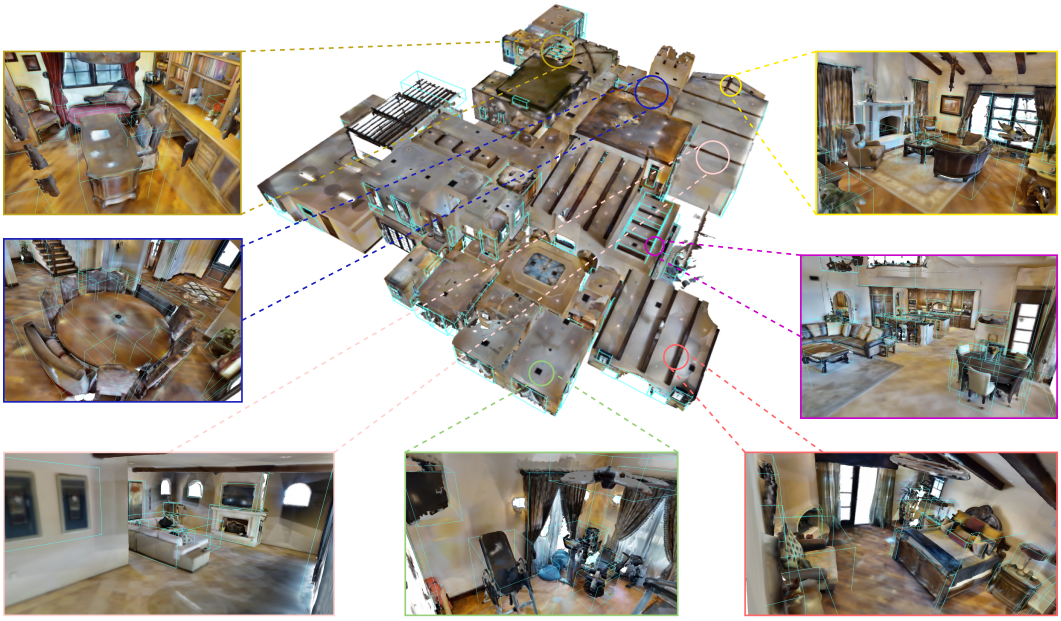


Figure 5: Visualization results of DSPDet3D transferring to building-level scenes.

Table 1 shows the experimental results on ScanNet-md40 and TO-SCENE-down. Consistent with the observation of [14], we find point-based (VoteNet, H3DNet, RBGNet) and transformer-based (GroupFree3D) methods almost fail to detect small objects. For methods (CAGroup3D, FCAF3D, TR3D) with sparse convolutional backbone, they achieve relatively much higher mAP_S due to sparse convolution [3, 4] 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 Visualization on Pruning

We visualize the pruning process under different thresholds in Figure 4, 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. With the increase of τ, DSP block prunes the voxels more and more aggressively with larger reduction on memory footprint while not losing the geometric information of small objects. As we will show in Section 4.4, our pruning method has greater advantage for scenes of larger scale.

4.4 Transferring to Larger Scenes

We further validate the efficiency and generalization ability of DSPDet3D by transferring it to scenes of much larger scale. We first train DSPDet3D on rooms from ScanNet training set in a category-agnostic manner, which is done by regarding every labeled object as the same category. Then we directly adopt it to process the building-level scenes in Matterport3D [11].

We find previous methods fail to process the extremely large scenes due to unaffordable memory footprint, or producing very bad results due to too aggressive downsampling. On the contrary, for a huge building consisting of 7000k+ points, DSPDet3D is able to process it in a single pass within 3s, using only a single RTX 3090 GPU and maintaining the peak memory footprint less than 10G. Several visualization results are shown in Figure 5. More can be found in supplementary material.

4.5 Limitations and Future Work

There are two main limitations of DSPDet3D. First, although it achieves high efficiency, the training cost is still large. We find using the learnable pruning module during training will lead to a large performance drop and so we have to adopt a weak pruning strategy. By solving this problem, DSPDet3D can be trained on scenes of larger scales in less time. Second, currently the input of DSPDet3D is a reconstructed point clouds. We will work on extending it to online 3D detection with RGB-D videos as input, which can support a wider range of practical application.

5 Conclusion

In this paper, we have presented a new detection framework for 3D small object detection. With in-depth analysis, we summarize three key points for designing an effective and effective 3D detector for small object detection: (1) multi-level FPN-like architecture; (2) increasing the overall spatial resolution of feature maps; (3) removing useless computation during upsampling operation. To this end, we propose DSPDet3D, a high-resolution multi-level 3D detector with sparse convolution. To detect small objects in high resolution with controllable computational cost, we devise a dynamic spatial pruning (DSP) block to generate object proposals from large to small and selectively upsample the feature map at regions where there are smaller objects to be detected in higher resolution. Upsampling is constrained to these regions by spatial pruning in a cascaded manner at inference time, while during training we switch the pruning to weak mode for context preservation. Extensive experiments on two benchmarks and transferring setting demonstrate the usefulness of DSPDet3D.

Supplementary Material

This supplementary material is organized as follows:

- Section A presents the analysis on the size of receptive field and the choice of r.
- Section B details the ScanNet-md40 and TO-SCENE-down benchmarks.
- Section C provides more visualization results of DSPDet3D on ScanNet and Matterport3D.
- Section D provides ablation study of our approach.
- Section E details the experimental results with per-category APs.

A Analysis on Receptive Field

Usually the receptive field refers to the range of pixels that the feature corresponds to in the original image. Here we define receptive field as the range of voxels that object proposal in level $i-1$ corresponds to in $\bar{F}_i^P$, which represents how many voxels will participate in the prediction of a bounding box. For each ground-truth center, we define the closest 27 voxels as the nearby object proposals and aim to keep the voxels in the receptive fields of these proposals unpruned. Below we first analyze the size of receptive field, and then we conduct ablation study to find out the optimal value of r.

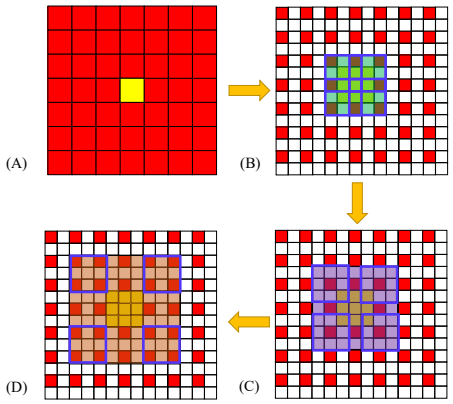


Figure 6: Illustration of the expanding of receptive field. We show the process in 2D for clear visualization. A, B, C and D refer to $\bar{F}_i^P$, F_{i-1}^M, F_{i-1}^P and object proposal features after convolution in the detection head respectively. Yellow voxels refer to the 27 closest object proposals to the ground-truth object center.

Analysis: We expect the pruned feature map $\bar{F}_i^P$ to keep the context information for objects in level $i-1$. Here we assume the voxels are dense to derive the upper bound of the receptive field. From $\bar{F}_i^P$ to the predicted bounding boxes in level $i-1$, there are three sparse convolution with kernel size 3 (an upsampling convolution and two normal convolution) and one with kernel size 1, which results in receptive field with a maximum size of $9 \times 9 \times 9$ for the 27 closest object proposals to the ground-truth center as shown in Figure 6. Therefore, r should be set close to 9. Considering the optimization of learnable pruning module may not be so perfect, we conduct ablation studies below to find out the optimal r.

Experiments: We train DSPDet3D with different r close to 9. We select seven values: $\{5, 7, 9, 11, 13, 15, 17\}$. For each of them, we train DSPDet3D 5 times and select the best model. As shown in Figure 7, setting $r = 9$ leads to a high performance, which validates our theoretical analysis. We empirically find $r = 13$ works slightly better, which may due to better optimization of the learnable pruning module. We set $r = 13$ for all other experiments.

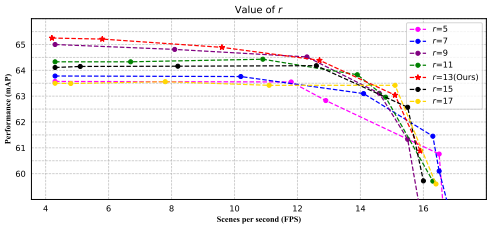


Figure 7: Ablation study on the value of r.

B Details on Benchmarks

We demonstrate the detailed categories and the number of objects in each class for both benchmarks in Figure 8. We observe significant long tail effect in both benchmarks and find that the number of

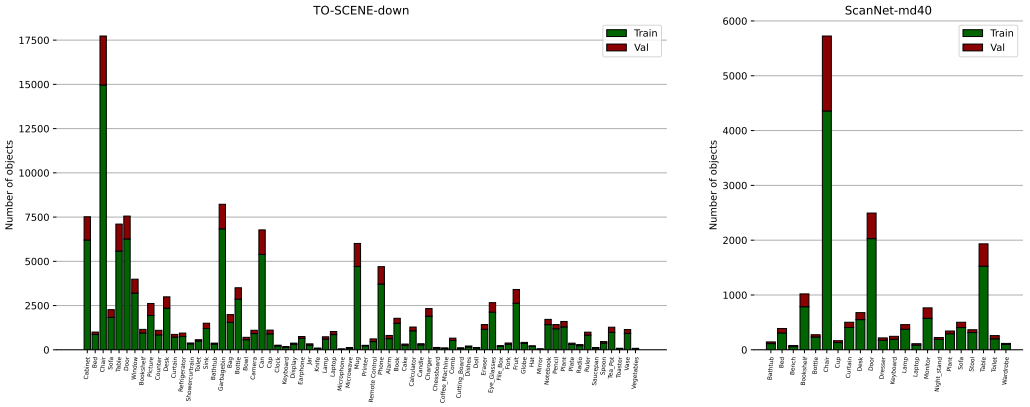


Figure 8: Number of objects in each category for ScanNet-md40 and TO-SCENE-down. For each category, we report the total number of objects on training and validation sets and use red and blue bars to distinguish the numbers on each set.

small objects in the real world dataset (i.e. ScanNet) is usually small, which poses great challenge to the 3D object detector.

In order to prove the necessity of downsampling for small objects in TO-SCENE-down, we further visualize the scenes in TO-SCENE before and after downsampling in Figure 9. It can be seen that the original dataset [15] contains densely and non-uniformly sampled small objects, whose density is obviously much larger than other objects and backgrounds. While after our downsampling, the overall scenes are closer to naturally sampled real scenes.

C More Visualization Results

We show visualization of the detection results on ScanNet-md40. We also provide more transferring results on Matterport3D.

Detection: We compare the predictions of DSPDet3D after NMS with the ground-truth bounding boxes in Figure 10. It can be seen that DSPDet3D produces accurate detection results with few false positives and successively detects small objects like cups and laptops, as shown in the red box.

Transferring: Figure 11 demonstrates more transferring results on building-level scenes, which is a supplement for Section 4.4 in the main paper.

D Ablation Study

We conduct ablation studies on ScanNet-md40 to study different design choices. We report mAP@0.25 and FPS at different threshold τ.

Partial additon: For partial addition $F_i^M = F_i^B \vec{+} F_{i+1}^U$, we add the features on the shared voxels of F_i^B and F_{i+1}^U, and directly use the original features on voxels unique to F_{i+1}^U. We compare this design with: (A) interpolate F_i^B to the voxels of F_{i+1}^U, then apply normal addition; (B) directly adding F_i^B and F_{i+1}^U by taking union of them. As shown in Figure 12, strategy B gets the worst speed-accuracy tradeoff. This is because the voxels of F_i^B make up for the pruned voxels of F_{i+1}^U, which results in huge computational costs due to the large amount of high-resolution features. Our strategy also outperforms strategy A by a remarkable margin, which indicates the simple 'add and ignore' operation is effective.

Block structure: In our learnable pruning module, the keeping mask is predicted from F_i^M. We compare this design with three variants: (A) predict keeping mask from F_i^P; (B) add the predictor for keeping mask to the shared detection head, parallel with the classification and regression branch; (C) similar to B but do not share the predictor across levels. Figure 12 shows that our design achieves the best speed-accuracy tradeoff. We think this is because the optimization objectives of the learnable

Before Downsampling | After Downsampling

Figure 9: Visualization of scenes in TO-SCENE before (originally provided in [15]) and after (our TO-SCENE-down benchmark) downsampling. After downsampling, the density of points on small objects is closer to others, which is more realistic.

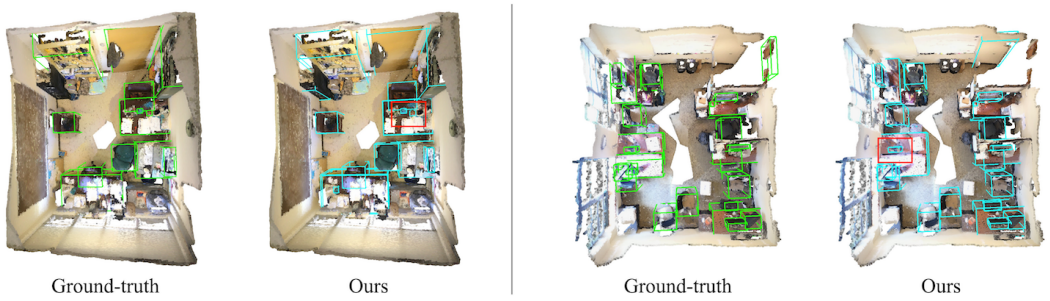


Figure 10: Visualization results of 3D object detection on ScanNet. Red boxes highlight our performance on detecting small objects.

pruning module and detection head is different: the pruning module aims to predict where objects from lower levels may appear, while the detection head aims to predict where are objects at current level. Our design does not rely on the proposal features and thus decouples the two optimization problems.

Ground-truth generation: We compare our ground-truth generation approach for with three variants: (A) replace the S_j with S_{i-1}, which is a fixed range for objects in different level; (B) replace the $\max\{|x - x_j^o|, |y - y_j^o|, |z - z_j^o|\}$ with $||(x, y, z) - c_j^o||_2$, which changes the cube range to its

Figure 11: Visualization results of DSPDet3D transferring to building-level scenes.

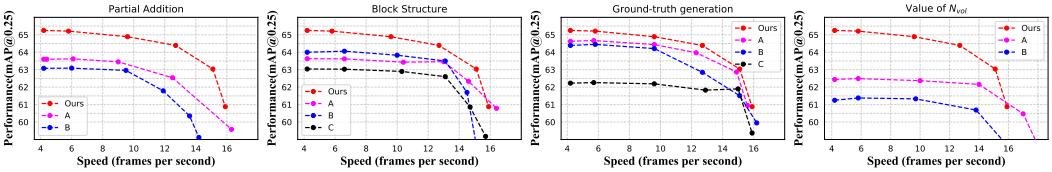


Figure 12: Ablation stduies on partial addition, structure of DSP block, ground-truth generation and hyperparameters on ScanNet.

inscribed ball; (C) combine A and B. As shown in Figure 12, using cube range is better than sphere range as cube better represents the shape of receptive field. Setting the side length proportional to the voxel size of current level achieves higher performance than a fixed length, which indicates a more precise K_i is better for training DSPDet3D.

Hyperparameters: We further study the effects of N_{vol}. We compare 27 with 18 (A) and 36 (B) in Figure 12. It is observed that an inappropriate N_{vol} leads to imbalanced assignment of ground-truth bounding boxes to each level, which degrades the performance.

E Class-specific Results

We provide more detailed experimental results on ScanNet-md40 and TO-SCENE-down with class-specific APs. Table 2, 3, 5 and 6 refer to AP@0.25 on ScanNet-md40, AP@0.5 on ScanNet-md40, AP@0.25 on TO-SCENE-down and AP@0.5 on TO-SCENE-down respectively. We highlight the categories of small objects in blue. It can be seen that DSPDet3D achieves much better performance on small objects compared with the state-of-the-arts.

We further conduct experiments on the recent ScanNet200 [49] benchmark, which shares the same scenes with ScanNet-md40 but has more categories (198 for object detection and instance segmentation). Note that total number of objects in some tail categories is even less than 10, which makes this benchmark extremely challenging without using the text embedding of CLIP [50] to assist training. We compare DSPDet3D with FCAF3D [25] and TR3D [8] under the same metric as in the main paper. As shown in Table 4, DSPDet3D achieves the highest detection accuracy on all metrics, as well as the fastest speed and lowest memory footprint, which further validates the effectiveness of our approach.

Table 2: The class-specific detection results (AP@0.25) of different methods on ScanNet-md40 benchmark. We highlight the categories of small objects in blue.

	VoteNet	*VoteNet$_S$*	*H3DNet*	*GroupFree3D*	*GroupFree$_S$*	*RBGNet*	*CAGroup3D*	*FCAF3D*	*TR3D*	*Ours($\tau = 0$)*	***Ours($\tau = 0.3$)***
Bathtub	90.39	84.26	91.46	92.70	71.36	44.22	44.69	83.65	93.93	83.13	83.16
Bed	87.30	89.03	88.46	89.11	71.58	90.74	87.74	86.91	87.64	88.78	88.80
Bench	48.34	40.56	46.89	38.20	10.01	47.79	42.09	39.20	53.05	55.32	55.50
Bookshelf	55.47	54.43	59.88	58.90	20.97	61.86	69.09	65.88	66.17	63.97	64.10
Bottle	0.00	0.06	1.22	4.16	0.00	0.20	5.56	2.98	3.23	28.23	17.27
Chair	89.03	89.15	91.70	93.20	56.76	94.16	95.04	94.96	95.58	95.99	95.58
Cup	0.00	0.00	0.00	4.37	0.01	0.00	0.00	5.85	12.95	26.91	27.58
Curtain	56.69	59.17	67.00	70.31	27.31	62.81	72.45	68.00	62.54	65.80	66.00
Desk	68.81	60.05	76.48	75.13	40.43	76.83	79.21	74.94	74.27	72.69	72.71
Door	53.17	48.57	59.45	61.62	16.70	59.55	61.78	60.26	60.91	64.65	64.79
Dresser	32.63	38.88	34.70	48.60	28.88	40.23	60.42	31.62	40.82	33.96	31.33
Keyboard	0.03	0.19	0.09	3.19	0.00	0.00	22.37	27.96	46.58	49.61	49.33
Lamp	49.26	43.29	49.33	59.05	8.76	56.90	61.19	61.00	61.32	68.79	68.01
Laptop	1.17	3.91	11.09	19.10	0.81	23.11	38.45	36.95	47.31	71.87	71.10
Monitor	67.18	68.76	74.88	83.35	30.12	83.63	86.93	88.71	89.20	90.78	89.81
Night_Stand	81.35	78.26	86.12	83.77	56.51	80.86	91.62	90.39	91.44	94.00	92.17
Plant	29.36	18.08	29.45	21.71	4.24	50.47	59.87	43.29	35.78	59.01	57.16
Sofa	88.89	87.33	89.09	84.79	53.99	91.32	91.41	88.38	91.32	90.98	90.99
Stool	44.34	37.72	37.63	44.54	17.66	51.41	55.25	52.51	42.65	51.88	52.23
Table	64.35	60.42	65.69	74.31	32.61	74.41	70.37	71.47	69.99	68.89	68.62
Toilet	94.01	95.48	97.89	95.82	82.95	97.76	99.73	99.39	98.81	98.91	99.11
Wardrobe	20.67	12.04	18.72	43.01	16.02	26.80	31.12	34.48	29.49	11.30	11.33

Table 3: The class-specific detection results (AP@0.5) of different methods on ScanNet-md40 benchmark. We highlight the categories of small objects in blue.

	VoteNet	*VoteNet$_S$*	*H3DNet*	*GroupFree3D*	*GroupFree$_S$*	*RBGNet*	*CAGroup3D*	*FCAF3D*	*TR3D*	*Ours($\tau = 0$)*	***Ours($\tau = 0.3$)***
Bathtub	79.13	84.26	86.16	85.47	44.68	39.46	42.97	83.65	84.21	76.72	76.73
Bed	82.43	80.35	82.69	81.85	42.71	78.17	81.93	80.91	82.32	83.84	83.84
Bench	1.63	4.57	22.45	2.36	0.51	1.53	14.50	27.95	34.15	31.71	31.79
Bookshelf	34.16	29.23	43.99	44.51	4.07	44.06	55.92	56.06	54.22	54.39	54.48
Bottle	0.00	0.00	0.00	0.08	0.00	0.00	1.44	0.48	1.50	17.91	11.57
Chair	74.22	69.87	80.52	83.41	14.31	82.33	90.19	89.98	91.13	92.06	91.59
Cup	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.69	5.98	24.22	24.89
Curtain	19.27	17.46	30.86	44.91	1.16	15.56	44.52	43.62	34.97	36.39	36.49
Desk	36.27	32.54	49.13	47.80	10.95	45.39	60.25	54.36	58.66	57.72	57.74
Door	22.37	18.54	32.57	38.00	2.05	34.14	46.36	42.34	43.72	45.43	45.38
Dresser	22.24	23.72	26.66	35.84	4.80	24.53	47.19	25.14	35.14	28.10	25.58
Keyboard	0.00	0.00	0.00	0.84	0.00	0.00	5.40	2.89	13.36	33.51	33.50
Lamp	21.95	11.31	28.96	35.29	0.08	26.10	53.80	47.10	42.60	56.50	55.74
Laptop	0.00	0.04	3.60	2.30	0.00	0.00	27.37	26.77	29.47	53.16	49.62
Monitor	28.18	24.70	35.05	42.82	2.01	30.24	68.77	66.31	74.31	76.31	75.51
Night_Stand	70.94	59.96	78.42	71.77	31.82	71.30	91.62	83.49	89.86	84.76	86.78
Plant	11.59	9.82	20.37	13.21	0.44	26.55	52.90	35.56	30.17	39.25	37.53
Sofa	72.68	76.60	74.89	70.23	24.38	67.25	83.29	78.04	78.14	82.23	82.23
Stool	28.24	21.08	25.53	40.49	7.10	15.68	51.59	47.54	40.50	46.90	47.34
Table	48.72	39.26	47.40	59.83	9.22	38.50	61.56	59.82	60.41	58.70	58.54
Toilet	81.74	86.43	85.06	89.99	61.14	75.64	96.08	95.72	87.51	91.74	91.88
Wardrobe	5.42	4.36	8.75	19.58	1.25	1.65	20.15	22.08	27.23	8.97	8.79

Table 4: 3D objects detection results and computational costs of different methods on ScanNet200 benchmark. We report both mAP@0.25 and mAP@0.5 for mAP, mAP$_S$ and mAP$_O$. DSPDet3D with the best pruning threshold is highlighted in gray.

Method	mAP		mAP$_S$		mAP$_O$		Speed	Memory
	@0.25	@0.5	@0.25	@0.5	@0.25	@0.5	(FPS)	(MB)
FCAF3D	28.74	21.81	17.34	10.85	33.45	26.36	8.1	1200
TR3D	30.18	22.58	21.08	10.85	33.72	26.42	7.2	2200
Ours($\tau = 0$)	33.39	25.94	23.68	18.59	37.41	28.98	3.6	7600
Ours($\tau = 0.4$)	32.74	25.37	22.47	18.43	36.57	28.25	8.6	900

Table 5: The class-specific detection results (AP@0.25) of different methods on To-Scene benchmark. We highlight the categories of small objects in blue.

	VoteNet	*VoteNet$_S$*	*H3DNet*	*GroupFree3D*	*GroupFree$_S$*	*RBGNet*	*CAGroup3D*	*FCAF3D*	*TR3D*	*Ours($\tau = 0$)*	***Ours($\tau = 0.5$)***
Cabinet	53.34	49.72	58.94	61.90	57.12	65.36	66.65	63.93	61.56	66.71	65.67
Bed	84.96	82.54	80.13	82.13	80.25	92.11	84.66	80.93	78.29	73.02	73.00
Chair	87.79	85.54	89.54	92.55	88.29	92.57	94.20	92.70	94.14	94.01	93.88
Sofa	92.00	89.87	87.43	91.32	86.33	88.96	86.99	89.75	90.65	90.38	90.44
Table	67.19	64.44	68.24	74.24	69.97	86.21	73.25	75.72	79.42	79.43	79.26
Door	51.39	52.88	53.85	60.60	53.62	59.99	59.99	50.37	55.88	58.77	58.23
Window	41.05	43.40	44.87	48.21	41.62	58.83	53.19	46.33	43.79	43.66	43.03
Bookshelf	29.69	24.08	30.24	28.43	27.74	27.52	26.29	21.24	29.02	30.71	31.07
Picture	6.60	6.78	8.26	11.73	8.28	28.63	26.59	14.04	11.80	27.39	18.73
Counter	56.73	51.35	63.49	62.73	63.39	71.11	71.83	67.90	63.98	60.24	60.32
Desk	59.65	57.20	56.39	59.23	49.80	64.55	64.29	63.38	59.20	61.75	61.78
Curtain	48.39	46.38	54.85	54.23	56.41	37.68	65.68	54.77	43.18	48.83	50.05
Refrigerator	49.68	54.76	69.77	65.12	41.31	67.51	74.50	75.35	65.85	77.05	77.16
Showercurtrain	75.65	78.95	75.76	78.47	81.55	83.46	79.70	78.48	39.93	60.84	59.27
Toilet	99.73	99.98	100.00	100.00	100.00	100.00	99.58	100.00	100.00	100.00	100.00
Sink	66.61	73.02	85.24	85.41	84.65	84.52	85.49	90.04	92.18	90.91	90.58
Bathtub	95.80	95.42	95.74	95.91	96.26	94.54	96.02	93.14	94.80	96.30	96.30
Garbagebin	51.42	49.21	56.03	68.63	58.57	68.96	69.86	68.51	66.30	70.17	69.75
Bag	39.22	49.08	44.83	60.35	69.97	73.75	89.27	83.63	89.28	92.49	91.80
Bottle	15.00	30.67	20.20	33.24	53.09	58.26	69.44	35.88	70.83	82.95	82.70
Bowl	11.79	23.27	8.72	12.58	44.17	39.89	85.21	55.08	73.95	93.30	93.36
Camera	7.88	15.57	9.40	6.91	24.00	13.06	53.30	44.11	62.24	74.15	74.65
Can	12.40	22.05	14.65	26.08	41.43	45.35	76.29	42.40	76.81	90.01	90.36
Cap	19.46	46.49	15.53	34.34	60.86	45.89	84.58	67.04	82.38	86.55	86.65
Clock	1.29	1.87	0.94	3.56	5.42	14.13	11.30	2.65	26.69	38.04	37.48
Keyboard	0.14	14.35	3.05	5.20	26.71	0.86	56.43	38.04	76.35	83.47	83.04
Display	53.27	47.56	54.15	46.78	66.93	81.61	88.27	86.55	87.09	89.26	89.33
Earphone	8.94	28.85	6.06	17.31	46.37	27.75	73.42	56.00	75.10	75.07	75.16
Jar	5.31	29.54	6.75	7.75	28.42	25.47	34.58	25.63	23.92	31.25	30.47
Knife	0.63	0.00	0.18	1.72	1.26	1.26	0.00	0.24	16.52	17.06	14.57
Lamp	34.45	53.44	41.38	58.99	67.95	72.67	89.10	78.45	83.13	90.33	89.80
Laptop	65.25	66.98	67.75	82.17	94.18	88.77	96.86	95.94	96.83	97.02	97.02
Microphone	0.04	0.00	0.01	0.01	0.04	0.16	0.00	0.16	0.96	1.29	1.30
Microwave	50.07	52.79	50.18	54.93	52.49	63.78	82.98	78.13	69.59	83.90	83.71
Mug	13.90	29.48	15.80	24.09	48.59	39.16	85.77	53.40	78.14	93.68	93.55
Printer	27.09	22.25	42.07	43.61	35.57	54.19	65.72	68.49	66.63	66.69	64.18
Remote Control	0.36	2.33	0.22	0.32	2.00	2.68	12.38	1.68	34.58	34.28	34.63
Phone	1.52	6.22	1.61	2.89	15.96	14.54	29.13	9.97	66.23	79.31	79.02
Alarm	3.07	12.11	3.56	9.51	19.04	13.44	39.08	20.60	30.59	48.68	49.06
Book	20.37	31.58	27.02	31.09	38.88	33.74	57.75	34.95	63.88	73.52	73.50
Cake	20.69	27.18	22.09	27.41	38.48	31.28	64.14	56.94	54.51	65.84	63.82
Calculator	1.51	6.34	1.99	2.88	13.71	11.74	21.56	16.73	34.34	43.94	44.05
Candle	28.00	29.63	21.62	42.58	49.31	53.19	56.87	41.87	65.45	64.62	66.25
Charger	0.03	2.12	0.33	0.53	1.78	8.09	22.22	6.33	37.47	55.25	55.54
Chessboard	6.80	45.14	19.33	27.38	74.76	71.96	87.45	78.31	77.94	86.30	86.36
Coffee_Machine	41.21	27.28	32.04	34.88	47.09	53.52	77.94	62.97	38.69	54.37	51.13
Comb	0.30	1.21	0.12	1.67	6.28	4.82	11.05	2.44	23.96	52.58	51.92
Cutting_Board	10.57	8.29	14.22	17.90	38.76	30.21	0.00	32.18	65.59	67.24	67.81
Dishes	11.25	26.30	9.16	21.03	40.12	26.11	70.50	42.89	64.15	63.34	62.01
Doll	1.14	1.89	0.70	7.24	17.74	2.12	9.55	1.68	14.54	18.80	18.55
Eraser	0.00	0.04	0.00	0.00	0.00	0.55	0.00	0.29	34.87	57.26	57.59
Eye_Glasses	5.67	23.20	7.84	12.57	40.69	31.52	81.29	58.37	91.95	96.94	96.90
File_Box	56.97	49.39	33.74	40.07	45.78	55.42	63.47	60.84	67.27	66.79	67.21
Fork	0.84	0.61	1.76	1.31	1.86	1.49	6.52	6.84	17.14	29.20	28.83
Fruit	2.56	9.54	2.15	7.53	29.77	20.58	62.04	32.09	52.27	80.58	80.58
Globe	30.87	29.41	19.91	39.84	52.62	35.67	75.65	64.64	65.02	79.33	78.79
Hat	1.87	22.49	2.95	4.18	13.18	6.83	53.65	23.64	32.08	50.39	50.90
Mirror	0.70	17.36	0.76	0.20	0.54	5.63	0.58	28.93	1.30	4.60	4.47
Notebook	3.90	8.12	8.52	8.33	23.19	26.65	37.44	17.61	59.38	65.24	65.25
Pencil	0.01	1.14	0.08	0.06	1.00	1.51	0.00	3.47	32.09	57.18	56.56
Plant	31.59	49.74	35.44	53.65	65.60	60.77	87.03	73.41	78.99	88.87	88.64
Plate	5.31	20.83	7.71	10.31	27.52	38.64	60.91	19.40	80.18	96.00	95.99
Radio	1.95	5.50	5.59	3.13	6.80	12.77	15.56	10.42	18.59	19.46	15.77
Ruler	0.02	1.25	0.42	0.13	1.68	0.96	3.99	1.09	35.56	53.27	55.83
Saucepan	31.34	32.74	24.73	39.52	47.05	37.65	73.89	37.29	30.28	62.53	60.70
Spoon	0.05	1.76	1.19	0.65	4.04	5.97	9.89	2.90	27.35	34.45	33.37
Tea_Pot	21.10	28.32	17.05	33.06	48.87	42.51	89.43	76.64	77.67	86.34	86.51
Toaster	13.88	21.95	19.51	11.42	16.27	19.90	34.61	31.32	29.11	30.89	31.88
Vase	25.95	37.29	17.00	31.59	46.84	37.72	65.79	52.50	54.39	65.82	65.52
Vegetables	0.43	7.12	0.40	0.66	18.35	10.27	0.01	4.73	9.62	6.89	6.56

Table 6: The class-specific detection results (AP@0.5) of different methods on To-Scene benchmark. We highlight the categories of small objects in blue.

	VoteNet	*VoteNet$_S$*	*H3DNet*	*GroupFree3D*	*GroupFree$_S$*	*RBGNet*	*CAGroup3D*	*FCAF3D*	*TR3D*	*Ours($\tau = 0$)*	***Ours($\tau = 0.5$)***
Cabinet	19.75	18.39	25.36	34.62	25.64	40.75	48.45	36.36	37.93	42.84	40.85
Bed	73.12	76.46	78.68	74.08	71.35	90.37	74.66	75.89	67.55	65.49	65.47
Chair	71.68	66.72	79.05	84.35	77.56	85.99	90.35	87.31	89.64	88.92	88.83
Sofa	86.54	82.29	84.88	86.89	79.51	83.64	81.79	88.81	90.65	88.51	88.57
Table	51.04	47.21	58.03	66.25	59.27	76.61	69.99	71.10	72.52	73.44	73.24
Door	25.72	24.80	33.47	43.78	33.32	41.23	49.61	39.67	42.73	45.24	44.91
Window	18.49	188.00	20.23	22.56	18.05	34.76	31.92	24.36	24.19	23.15	23.76
Bookshelf	22.22	14.42	15.49	12.83	16.29	20.03	22.90	10.33	22.22	25.68	26.02
Picture	0.69	0.35	1.75	6.88	3.42	13.88	13.48	5.12	3.50	18.32	11.81
Counter	18.08	9.07	17.45	37.21	31.22	40.43	49.28	43.81	40.71	27.83	27.89
Desk	35.78	26.70	35.47	45.83	37.67	45.68	37.99	42.25	42.13	42.67	42.67
Curtain	11.85	13.01	6.57	15.00	19.05	12.10	19.58	10.60	15.42	26.07	26.67
Refrigerator	38.74	33.15	47.15	27.88	21.22	53.58	64.15	54.13	50.53	65.77	65.84
Showercurtrain	41.76	41.14	70.98	71.58	40.11	78.05	68.52	72.77	30.12	52.50	48.45
Toilet	92.55	92.28	100.00	100.00	100.00	100.00	99.58	100.00	100.00	100.00	100.00
Sink	19.97	22.51	38.04	43.85	42.29	40.28	51.54	46.37	55.16	63.02	65.59
Bathtub	72.46	81.61	79.25	80.75	80.94	81.01	81.27	88.77	85.38	92.09	92.09
Garbagebin	27.98	23.46	40.25	50.67	41.58	56.65	64.57	60.39	57.68	62.95	62.04
Bag	12.48	18.08	20.08	26.86	41.31	57.46	82.31	75.13	81.16	88.94	88.06
Bottle	3.35	7.77	6.43	8.22	13.78	43.00	66.27	27.81	65.95	80.86	80.60
Bowl	0.81	5.34	0.50	0.49	12.83	19.49	82.56	52.06	68.56	93.30	93.36
Camera	2.19	2.05	3.38	1.52	10.81	8.98	49.93	35.38	57.19	71.84	72.28
Can	2.06	3.50	3.60	5.47	10.44	36.19	73.51	31.74	68.29	87.36	87.47
Cap	1.69	16.38	6.49	10.87	33.43	37.72	80.34	64.26	82.36	86.53	86.64
Clock	0.00	0.19	0.00	2.73	0.40	8.94	8.68	0.97	14.65	31.75	30.99
Keyboard	0.00	7.46	0.00	1.07	2.21	0.46	46.01	18.12	51.17	72.51	73.51
Display	11.84	12.10	26.07	31.46	28.19	51.38	82.24	77.47	80.11	85.47	85.40
Earphone	2.59	6.62	0.92	4.52	22.85	20.82	3370.73	49.99	71.87	72.29	72.33
Jar	2.10	13.77	2.96	2.29	21.22	22.42	33.81	21.27	22.18	31.25	30.47
Knife	0.00	0.00	0.00	0.00	1.00	0.00	0.00	0.00	1.40	7.34	6.33
Lamp	13.97	27.06	25.67	43.75	52.96	62.17	89.10	73.67	80.54	90.33	89.80
Laptop	24.34	17.31	42.20	52.85	81.32	75.85	94.51	91.86	94.32	95.63	95.63
Microphone	0.00	0.00	0.00	0.01	0.01	0.06	0.00	0.02	0.96	0.64	0.65
Microwave	16.18	25.78	36.97	41.91	47.12	54.50	82.98	77.52	66.45	83.90	83.71
Mug	2.60	7.84	4.74	5.00	13.03	27.83	83.00	42.79	73.64	92.15	92.15
Printer	10.56	6.99	26.64	32.36	26.62	45.87	65.05	66.80	61.81	66.69	64.18
Remote Control	0.00	0.33	0.01	0.00	0.00	0.19	1.66	0.85	21.92	25.11	24.82
Phone	0.04	0.39	0.02	0.09	0.50	1.95	18.86	4.90	34.50	59.05	58.48
Alarm	0.49	4.37	0.85	2.44	6.84	7.79	35.16	17.02	26.70	44.66	45.66
Book	3.91	5.70	6.59	5.72	8.87	19.42	54.52	30.53	57.04	72.31	72.29
Cake	15.39	15.40	20.09	23.34	33.24	30.27	61.54	56.42	53.17	64.14	62.11
Calculator	0.02	0.19	0.68	0.05	0.61	2.56	16.88	10.28	27.70	40.04	40.20
Candle	11.18	10.59	11.51	18.20	17.60	36.10	53.74	36.69	26.06	61.50	62.89
Charger	0.01	0.01	0.07	0.00	0.01	0.71	10.35	0.90	12.03	33.55	33.22
Chessboard	0.00	3.19	0.11	8.91	36.56	45.26	78.72	62.71	67.45	76.07	76.31
Coffee_Machine	11.13	20.39	20.38	18.43	34.55	53.51	77.56	57.93	38.69	54.37	51.13
Comb	0.00	0.00	0.00	0.04	0.08	0.01	5.77	0.03	2.13	38.05	39.29
Cutting_Board	2.79	0.69	2.19	4.57	19.23	15.97	0.00	27.23	54.40	51.02	51.30
Dishes	1.30	2.28	1.73	5.15	10.01	15.77	65.85	29.00	61.69	63.28	62.01
Doll	0.14	1.01	0.00	6.84	5.52	2.06	9.12	1.44	14.48	18.75	18.50
Eraser	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	10.29	20.42	19.18
Eye_Glasses	0.15	1.78	0.65	0.48	2.96	9.69	72.73	37.83	77.15	90.26	90.17
File_Box	15.22	15.48	23.51	26.95	35.66	53.42	63.36	62.91	66.86	66.79	67.21
Fork	0.00	0.00	0.00	0.00	0.17	0.00	0.05	0.00	2.98	11.84	10.65
Fruit	0.77	0.88	0.41	1.82	7.11	11.44	58.14	22.36	42.00	77.24	77.45
Globe	25.48	25.42	16.80	37.32	49.97	32.95	75.62	64.77	64.98	79.33	78.79
Hat	0.04	13.90	2.46	1.66	3.41	6.23	46.89	24.34	32.08	50.39	50.90
Mirror	0.13	16.71	0.03	0.00	0.00	0.90	0.47	3.76	0.97	4.60	4.38
Notebook	0.31	0.95	1.19	0.41	2.04	6.87	24.77	7.09	45.73	52.30	52.13
Pencil	0.00	0.03	0.00	0.00	0.02	0.03	0.00	0.21	7.83	30.62	29.91
Plant	12.55	25.19	24.01	39.13	49.13	52.49	84.15	66.91	75.74	87.27	87.05
Plate	0.36	0.69	0.41	0.87	1.61	16.54	38.90	18.06	74.66	86.40	86.38
Radio	0.07	0.13	2.14	0.02	1.85	3.39	4.81	3.18	11.42	9.10	9.72
Ruler	0.00	0.06	0.00	0.00	0.00	0.00	0.17	0.07	8.07	17.79	20.61
Saucepan	21.64	16.14	11.19	26.24	17.81	31.15	73.49	30.25	24.75	62.53	60.70
Spoon	0.00	0.00	0.00	0.00	0.10	0.20	0.56	1.21	5.45	20.12	17.88
Tea_Pot	7.03	12.42	9.29	21.50	31.24	34.85	87.60	73.44	76.46	85.68	85.87
Toaster	2.18	6.18	10.51	5.12	4.86	15.79	32.65	26.29	27.72	30.34	31.21
Vase	7.43	17.93	11.02	19.63	36.39	33.34	64.57	52.35	52.75	64.76	64.46
Vegetables	0.00	0.18	0.13	0.20	4.91	9.75	0.01	5.00	9.62	6.89	6.54

References

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

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

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

[4] C. Choy, J. Gwak, and S. Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *CVPR*, pages 3075–3084, 2019.

[5] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *CVPR*, pages 10529–10538, 2020.

[6] W. Zheng, W. Tang, L. Jiang, and C.-W. Fu. Se-ssd: Self-ensembling single-stage object detector from point cloud. In *CVPR*, pages 14494–14503, 2021.

[7] H. Wang, L. Ding, S. Dong, S. Shi, A. Li, J. Li, Z. Li, and L. Wang. Cagroup3d: Class-aware grouping for 3d object detection on point clouds. *arXiv preprint arXiv:2210.04264*, 2022.

[8] D. Rukhovich, A. Vorontsova, and A. Konushin. Tr3d: Towards real-time indoor 3d object detection. *arXiv preprint arXiv:2302.02858*, 2023.

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

[10] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, pages 5828—-5839, 2017.

[11] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *3DV*, 2017.

[12] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3d semantic parsing of large-scale indoor spaces. In *ICCV*, pages 1534–1543, 2016.

[13] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *CVPR*, pages 567–576, 2015.

[14] X. Xu, Y. Wang, Y. Zheng, Y. Rao, J. Zhou, and J. Lu. Back to reality: Weakly-supervised 3d object detection with shape-guided label enhancement. In *CVPR*, pages 8438–8447, 2022.

[15] M. Xu, P. Chen, H. Liu, and X. Han. To-scene: A large-scale dataset for understanding 3d tabletop scenes. In *ECCV*, pages 340–356. Springer, 2022.

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

[17] C. R. Qi, O. Litany, K. He, and L. J. Guibas. Deep hough voting for 3d object detection in point clouds. In *ICCV*, pages 9277–9286, 2019.

[18] Q. Xie, Y.-K. Lai, J. Wu, Z. Wang, Y. Zhang, K. Xu, and J. Wang. Mlcvnet: Multi-level context votenet for 3d object detection. In *CVPR*, pages 10447–10456, 2020.

[19] Z. Zhang, B. Sun, H. Yang, and Q. Huang. H3dnet: 3d object detection using hybrid geometric primitives. In *ECCV*, pages 311–329, 2020.

[20] B. Cheng, L. Sheng, S. Shi, M. Yang, and D. Xu. Back-tracing representative points for voting-based 3d object detection in point clouds. In *CVPR*, pages 8963–8972, 2021.

[21] H. Wang, S. Shi, Z. Yang, R. Fang, Q. Qian, H. Li, B. Schiele, and L. Wang. Rbgnet: Ray-based grouping for 3d object detection. In *CVPR*, pages 1110–1119, 2022.

[22] I. Misra, R. Girdhar, and A. Joulin. An end-to-end transformer model for 3d object detection. In *ICCV*, pages 2906–2917, 2021.

[23] Z. Liu, Z. Zhang, Y. Cao, H. Hu, and X. Tong. Group-free 3d object detection via transformers. *arXiv preprint arXiv:2104.00678*, 2021.

[24] J. Gwak, C. Choy, and S. Savarese. Generative sparse detection networks for 3d single-shot object detection. In *ECCV*, pages 297–313. Springer, 2020.

[25] D. Rukhovich, A. Vorontsova, and A. Konushin. Fcaf3d: fully convolutional anchor-free 3d object detection. In *ECCV*, pages 477–493. Springer, 2022.

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

[27] J. Lee, C. Choy, and J. Park. Putting 3d spatially sparse networks on a diet. *arXiv preprint arXiv:2112.01316*, 2021.

[28] X. Xu, Z. Wang, J. Zhou, and J. Lu. Binarizing sparse convolutional networks for efficient point cloud analysis. *arXiv preprint arXiv:2303.15493*, 2023.

[29] J. Liu, Y. Chen, X. Ye, Z. Tian, X. Tan, and X. Qi. Spatial pruned sparse convolution for efficient 3d object detection. In *Advances in Neural Information Processing Systems*, 2022.

[30] K. Tong, Y. Wu, and F. Zhou. Recent advances in small object detection based on deep learning: A review. *IVC*, 97:103910, 2020.

[31] M. Kisantal, Z. Wojna, J. Murawski, J. Naruniec, and K. Cho. Augmentation for small object detection. *arXiv preprint arXiv:1902.07296*, 2019.

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

[33] B. Zoph, E. D. Cubuk, G. Ghiasi, T.-Y. Lin, J. Shlens, and Q. V. Le. Learning data augmentation strategies for object detection. In *ECCV*, pages 566–583. Springer, 2020.

[34] B. Singh and L. S. Davis. An analysis of scale invariance in object detection snip. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3578–3587, 2018.

[35] B. Singh, M. Najibi, and L. S. Davis. Sniper: Efficient multi-scale training. *NeurIPS*, 31, 2018.

[36] M. Gao, R. Yu, A. Li, V. I. Morariu, and L. S. Davis. Dynamic zoom-in network for fast object detection in large images. In *CVPR*, pages 6926–6935, 2018.

[37] M. Najibi, B. Singh, and L. S. Davis. Autofocus: Efficient multi-scale inference. In *ICCV*, pages 9745–9755, 2019.

[38] J. Li, X. Liang, Y. Wei, T. Xu, J. Feng, and S. Yan. Perceptual generative adversarial networks for small object detection. In *CVPR*, pages 1222–1230, 2017.

[39] C. Chen, M.-Y. Liu, O. Tuzel, and J. Xiao. R-cnn for small object detection. In *ACCV*, pages 214–230. Springer, 2017.

[40] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, et al. Deep high-resolution representation learning for visual recognition. *TPAMI*, 43(10): 3349–3364, 2020.

[41] C. Deng, M. Wang, L. Liu, Y. Liu, and Y. Jiang. Extended feature pyramid network for small object detection. *TMM*, 24:1968–1979, 2021.

[42] C. Yang, Z. Huang, and N. Wang. Querydet: Cascaded sparse query for accelerating high-resolution small object detection. In *CVPR*, pages 13668–13677, 2022.

[43] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[44] J.-H. Luo, J. Wu, and W. Lin. Thinet: A filter level pruning method for deep neural network compression. In *ICCV*, pages 5058–5066, 2017.

[45] L. Liu, Z. Pan, and B. Lei. Learning a rotation invariant detector with rotatable bounding box. *arXiv preprint arXiv:1711.09405*, 2017.

[46] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell. Rethinking the value of network pruning. In *ICLR*, 2018.

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

[48] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 32, 2019.

[49] D. Rozenberszki, O. Litany, and A. Dai. Language-grounded indoor 3d semantic segmentation in the wild. In *ECCV*, pages 125–141. Springer, 2022.

[50] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763. PMLR, 2021.