This CVPR 2020 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

3DSSD: Point-based 3D Single Stage Object Detector

Zetong Yang[1] Yanan Sun[2] Shu Liu[3] Jiaya Jia[1,3]
[1]The Chinese University of Hong Kong [2]Hong Kong University of Science and Technology
[3]SmartMore
{tomztyang, now.syn}@gmail.com sliu@smartmore.com leojia@cse.cuhk.edu.hk

Abstract

Prevalence of voxel-based 3D single-stage detectors contrast with underexplored point-based methods. In this paper, we present a lightweight point-based 3D single stage object detector 3DSSD to achieve decent balance of accuracy and efficiency. In this paradigm, all upsampling layers and the refinement stage, which are indispensable in all existing point-based methods, are abandoned. We instead propose a fusion sampling strategy in downsampling process to make detection on less representative points feasible. A delicate box prediction network, including a candidate generation layer and an anchor-free regression head with a 3D center-ness assignment strategy, is developed to meet the demand of high accuracy and speed. Our 3DSSD paradigm is an elegant single-stage anchor-free one. We evaluate it on widely used KITTI dataset and more challenging nuScenes dataset. Our method outperforms all state-of-the-art voxel-based single-stage methods by a large margin, and even yields comparable performance with two-stage point-based methods, with amazing inference speed of 25+ FPS, 2× faster than former state-of-the-art point-based methods.

1. Introduction

3D scene understanding has attracted much attention since it benefits many applications, such as autonomous driving [7] and augmented reality [17]. In this paper, we focus on the fundamental task of 3D object detection, which predicts 3D bounding boxes and class labels for each instance within a point cloud.

Although great breakthrough has been made in 2D detection, it is still not possible to directly apply these 2D methods to 3D because of the unique characteristics of point cloud. Compared with 2D images, point cloud is sparse, unordered and locality sensitive, making it hard to use convolution neural networks (CNNs) for parsing. How to convert and utilize raw point cloud data has become the primary problem in the detection task.

Several existing methods convert point clouds from sparse formation to compact representations by projecting them to images [4, 11, 8, 18, 5], or subdividing them to equally distributed voxels [16, 26, 33, 29, 28, 12]. We call these methods voxel-based ones, which require voxelization on the whole point cloud. Features in each voxel are generated by either PointNet-like backbones [21, 22] or handcrafted features. Then a variety of 2D detection paradigms can be applied in the compact voxel space. Although these methods are straightforward and efficient, they suffer from information loss during voxelization and encounter performance bottleneck.

Another stream is with point-based methods [31, 32, 23]. They take raw point clouds as input, and predict bounding boxes based on each point. Specifically, they are composed of two stages. In the first stage, *set abstraction* (SA) layers are used for downsampling and extracting context features. Afterwards, *feature propagation* (FP) layers are applied for upsampling and broadcasting features to points, which are discarded during downsampling. A 3D region proposal network (RPN) is then applied for generating proposals centered at each point. Based on these proposals, a refinement module is developed in the second stage to give final prediction. These methods achieve better performance. But inference usually takes much longer time.

Our Contributions Different from all previous methods, we develop a lightweight and efficient point-based 3D single stage object detection framework. Our key observation is that in point-based methods, FP layers and the refinement stage consume half of the inference time. However, it is non-trivial to abandon FP layers. Under the current sampling strategy in SA with only furthest-point-sampling based on 3D Euclidean distance (D-FPS), foreground instances with only a few interior points may be lost after sampling. Consequently, it is impossible for them to be detected, which leads to huge performance drop.

In STD [32], without upsampling and only conducting detection on remaining downsampled points, the performance drops by about 9%. That is why FP layers must be used for point upsampling, albeit a large amount of extra computation is consumed. To deal with this issue, we first

propose a new sampling strategy based on feature distance, called F-FPS, which effectively preserves interior points of various instances. Our final sampling strategy becomes a fusion version of F-FPS and D-FPS.

To better exploit the *representative points* retained after SA layers, we develop a box prediction network, which utilizes a candidate generation layer (CG), an anchor-free regression head and a 3D center-ness assignment strategy. In the CG layer, we first shift representative points from F-FPS to generate *candidate points*. This shifting operation is supervised by the relative locations between the representative points and centers of their corresponding instances.

Then, we treat these candidate points as centers, find their surrounding points from the whole set of representative points from both F-FPS and D-FPS, and extract their features through multi-layer perceptron (MLP) networks. These features are finally fed into an anchor-free regression head to predict 3D bounding boxes. We also design a 3D center-ness assignment strategy, which assigns higher classification scores to candidate points closer to instance centers, in order to retrieve precise localization prediction.

We evaluate our method on widely used KITTI [6] dataset, and more challenging nuScenes [3] dataset. Experiments show that our model outperforms all state-of-the-art voxel-based single-stage methods by a large margin, and even achieves comparable performance with all two-stage point-based methods at a much faster inference speed. Our primary contribution is manifold.

- We propose a lightweight and effective point-based 3D single-stage object detector 3DSSD. We remove computational-heavy FP layers and the refinement module, which are however indispensable in all existing point-based methods.
- A novel fusion sampling strategy in SA layers is developed to keep adequate interior points of different foreground instances. It preserves rich information for regression and classification.
- We design a box prediction network to better effectiveness and efficiency. Experimental results show that our framework outperforms all single-stage methods, and yields comparable performance to state-of-the-art two-stage methods with much higher efficiency (38ms per scene).

2. Related Work

3D Object Detection with Multiple Sensors There are several methods exploiting the way to fuse information from multiple sensors for object detection. MV3D [4] projects LiDAR point cloud to bird-eye view (BEV) in order to generate proposals. These proposals with other information from images, front view and BEV are then sent to the second stage to predict final bounding boxes. AVOD [11] extends MV3D by introducing image features in the proposal generation stage. MMF [14] fuses information from depth maps, LiDAR point clouds, images and maps to accomplish multiple tasks including depth completion, 2D object detection and 3D object detection. These tasks benefit each other and enhance final performance on 3D object detection.

3D Object Detection with LiDAR Only Mainly two streams of methods deal with 3D object detection only using LiDAR data. One is voxel-based, which applies voxelization on the entire point cloud. The difference among these voxel-based methods lies on the initialization of voxel features. In [26], each non-empty voxel is encoded with 6 statistical quantities by the points within this voxel. Binary encoding is used in [13] for each voxel grid. VoxelNet [33] utilizes PointNet [21] to extract features of each voxel. Compared to [33], SECOND [28] applies sparse convolution layers [9] for parsing the compact representation. PointPillars [12] treats pseudo-images as the representation after voxelization.

Another line is point-based, which takes raw point cloud as input, and generates predictions based on each point. F-PointNet [20] and IPOD [31] adopt 2D-mechanism-like detection or segmentation to filter most useless points, and generate predictions from kept useful points. PointRCNN [23] utilizes PointNet++ [22] with SA and FP layers to extract features for each point, proposes a region proposal network (RPN) to generate proposals, and applies a refinement module to predict bounding boxes and class labels. These methods outperform voxel-based ones, and yet with much longer inference time. They cannot be applied to real-time autonomous driving systems.

STD [32] takes advantage of both point- and voxel-based methods. It uses raw point cloud as input, applies PointNet++ to extract features, proposes a PointsPool layer for converting features from sparse to dense representation, and finally utilizes CNNs in the refinement module. It's speed is faster than former point-based methods, and meanwhile is still much slower than voxel-based ones. As analyzed above, all point-based methods are composed of two stages of proposal generation – including SA layers and FP layers – and refinement for accurate prediction. It is the first attempt in this paper not to use FP layers and the refinement module, so as to speed up the whole procedure.

3. Our Framework

In this section, we first analyze the bottleneck of point-based methods, and describe our proposed fusion sampling strategy. Next, we present the box prediction network including a candidate generation layer, anchor-free regression head and our 3D center-ness assignment strategy. Fi-

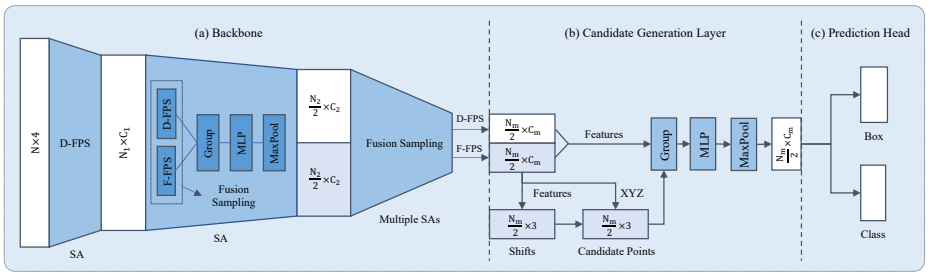

Figure 1. Illustration of the 3DSSD framework. It has a backbone box prediction network that includes a candidate generation layer and an anchor-free prediction head. (a) Backbone network. It takes the raw point cloud (x, y, z, r) as input, and generates global features for all representative points through several SA layers with fusion sampling (FS) strategy. (b) Candidate generation layer (CG). It downsamples, shifts and extracts features for representative points after SA layers. (c) Anchor-free prediction head.

nally, we discuss the loss function. The whole framework of 3DSSD is illustrated in Figure 1.

3.1. Fusion Sampling

Motivation As aforementioned, there are two streams of methods in 3D object detection, which are point-based and voxel-based frameworks. Albeit accurate, point-based methods are more time-consuming compared to voxel-based ones. All current point-based methods [32, 23, 31] are composed of the two stages of proposal generation and prediction refinement.

In first stage, SA layers are applied to downsample points for better efficiency and enlarging receptive fields, while FP layers are applied to broadcast features for dropped points during downsampling process, in order to recover all points. In the second stage, a refinement module optimizes proposals from RPN to get more accurate prediction. SA layers are necessary for extracting features of points. We reiterate that FP layers and the refinement module limit the efficiency, as shown in Table 1. We are thus motivated to design a lightweight and effective point-based single stage detector.

Challenge It is non-trivial to remove FP layers. SA layers in backbone utilize D-FPS to choose a subset of points as the downsampled *representative points*. Without FP layers, the box prediction network has to be conducted on those surviving representative points. Nonetheless, this sampling method only takes the relative locations among points into consideration. Consequently a large portion of surviving representative points are actually background ones, due to the large amount.

Now with a limited number N_m of the total representative points, for remote (or small) instances, their inner points are not likely to be selected, because the amount is much smaller than that of background points. The situation becomes even worse on more complex datasets, like nuScenes [3].

Statistically, we use points recall – the quotient between the number of instances whose interior points survived in the sampled representative points and the total number of instances, to help illustrate this fact. As listed in the first row of Table 2, with 1,024 (or 512) representative points, point recalls are only 65.9% (or 51.8%) respectively, which means nearly half of the instances are totally erased and cannot be detected. To ameliorate this problem, most of existing methods apply FP layers to recall those abandoned useful points during downsampling, under the heavy cost of computation during inference.

Methods	SA layers (ms)	FP layers (ms)	Refinement Module (ms)
Baseline	40	14	35

Table 1. Running time of different components in our reproduced PointRCNN [23] model, which has 4 SA layers and 4 FP layers for feature extraction, and a refinement module with 3 SA layers for prediction.

Methods	4,096	1,024	512
D-FPS	99.7 %	65.9 %	51.8 %
F-FPS (λ=0.0)	99.7 %	83.5 %	68.4 %
F-FPS (λ=0.5)	99.7 %	84.9 %	74.9 %
F-FPS (λ=1.0)	99.7 %	89.2 %	76.1 %
F-FPS (λ=2.0)	99.7 %	86.3 %	73.7 %

Table 2. Points recall among different sampling strategies on nuScenes dataset. "4,096", "1,024" and "512" stand for the amounts of representative points in the subset.

Instance Num(points sampled)/
Instance Num(ALL)

Feature-FPS In order to preserve *positive points* (interior points within any instance) and erase those useless *negative points* (points locating on background), we consider not only spatial distance but also semantic information of each point during the sampling process. We note that semantic information is well captured by the deep neural network. So, utilizing the feature distance as the criterion in FPS can remove many similar negative points on background. It is intriguing that positive points of remote objects can still survive because semantic features of points from different objects are distinct from each other.

However, only taking the semantic feature distance as the sole criterion would preserve quite a number of points

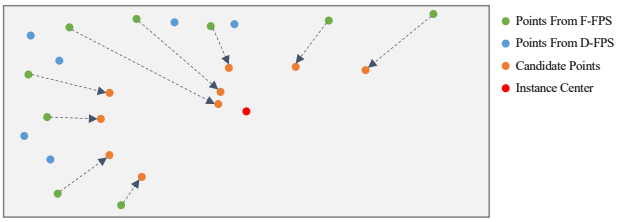

Figure 2. Illustration of the shifting operation in the CG layer. The gray rectangle represents an instance with all positive representative points from F-FPS (green) and D-FPS (blue). The red dot represents instance center. We only shift points from F-FPS under the supervision of their distances to the center of an instance.

within one instance, which introduces redundancy. For example, given a car, there is prominent difference between features of points around the windows and those of wheels. As a result, points around the two parts are respectively sampled, while points in either part are already informative for regression.

Therefore, to reduce the redundancy and increase the diversity, we apply both spatial distance and semantic feature distance as the criteria in FPS. It is formulated as

$$C(A, B) = \lambda L_d(A, B) + L_f(A, B), \tag{1}$$

where $L_d(A, B)$ and $L_f(A, B)$ represent $L2$ $X - Y - Z$ distance and $L2$ feature distance between two points. λ is the balance factor. We call this sampling method Feature-FPS (F-FPS). The comparison of using different λ is shown in Table 2, which demonstrates that combining the two distances in the downsampling operation is more powerful than only using feature distance where λ is set to 0.

Moreover, as illustrated in Table 2, using F-FPS with 1,024 representative points and setting λ to 1 guarantee that 89.2% of the instances are preserved in nuScenes [3] dataset, 23.3% higher than the D-FPS sampling strategy.

Fusion Sampling A large amount of positive points within different instances are preserved through SA layers thanks to F-FPS. However, with the limited number N_m of total representative points, many negative points are discarded during the downsampling process, which benefits regression and yet hampers classification. During the grouping stage in a SA layer, which aggregates features from neighboring points, a negative point is unable to find enough surrounding points, making it impossible to enlarge its receptive field.

As a result, it is difficult to distinguish between positive and negative points, leading to poor performance in classification. Our experiments also demonstrate this limitation in ablation study. Although the model with F-FPS yields a higher recall rate and better localization accuracy than the one with D-FPS, it mistakenly treats several negative points as positive ones, leading to drop of classification accuracy.

The analysis above indicates that, after a SA layer, not only positive points should be sampled as many as possible, but also we need to gather enough negative points for more reliable classification. We present a novel fusion sampling strategy (FS), in which both F-FPS and D-FPS are applied during a SA layer, to retain more positive points for localization and enough negative points for classification as well.

Specifically, we sample $\frac{N_m}{2}$ points respectively with F-FPS and D-FPS and feed the two sets together to the following grouping operation in a SA layer.

3.2. Box Prediction Network

Candidate Generation Layer After the backbone network implemented with several SA layers and fusion sampling, we gain a subset of points from both F-FPS and D-FPS, which are used for final prediction. In former point-based methods, another SA layer is applied to extract features before the prediction head. There are three steps in a normal SA layer, including center point selection, surrounding points extraction and semantic feature generation.

In order to further reduce computation cost and fully utilize the advantage of fusion sampling, we present a candidate generation layer (CG) before our prediction head, which is a variant of SA layer. Since most of representative points from D-FPS are negative, useless in bounding box regression, we only take those from F-FPS as initial center points. They are shifted under the supervision of their relative locations to their corresponding instances as illustrated in Figure 2, same as the way of VoteNet [19]. We call these new points after shifting *candidate points*.

Then we treat these candidate points as the center ones in our CG layer. We use candidate points rather than original ones as the center for the sake of performance, which will be discussed in detail later. Next, we find the surrounding points of each candidate point from the whole representative point set containing points from both D-FPS and F-FPS with a pre-defined range threshold and concatenate their normalized location and semantic features as input. MLP layers are finally applied to extract features. These features are sent to the prediction head for regression and classification. This entire process is illustrated in Figure 1.

Anchor-free Regression Head With fusion sampling strategy and the CG layer, our model can safely remove the time-consuming FP layers and the refinement module. In the regression head, we have two options of building anchor-based or anchor-free prediction network. For anchor-based head, we need to construct multi-scale and multi-orientation anchors to cover objects with various sizes and orientations. In complex scenes like those in the nuScenes dataset [3], objects are from 10 different categories with a wide range of orientations. We thus need at least 20 anchors, including 10 different sizes and 2 different

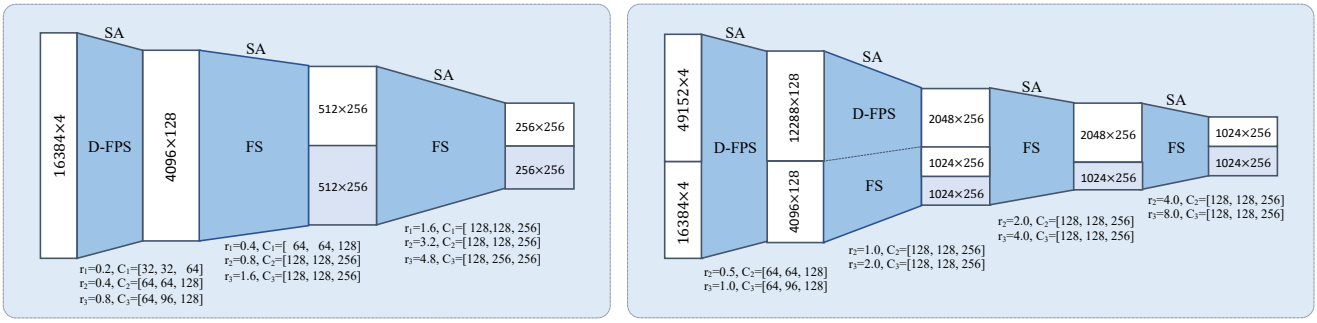

Figure 3. Backbone network of 3DSSD on KITTI (left) and nuScenes (right) datasets.

orientations $(0, \pi/2)$ in an anchor-based model. To avoid this cumbersome setting with multiple anchors and stick with our lightweight design, we utilize anchor-free regression head instead.

In the regression head, for each candidate point, we predict distance (d_x, d_y, d_z) to its corresponding instance, as well as size (d_l, d_w, d_h) and orientation of its corresponding instance. Since there is no prior orientation of each point, we apply hybrid of classification and regression formulation following [20] in orientation angle regression. Specifically, we define N_a equally split orientation angle bins and classify the proposal orientation angle into one of these bins. Residual is regressed with respect to the bin value. N_a is set to 12 in our experiments.

3D Center-ness Assignment Strategy In the training process, we need an assignment strategy to assign labels for each candidate point. In 2D single-stage detectors, intersection-over-union (IoU) [15] threshold or mask [25, 30] can be used. FCOS [25] adopts a continuous center-ness label, which replaces original binary classification label to further help distinguish among pixels. It assigns higher center-ness scores to pixels closer to instance centers, leading to relatively better performance compared to IoU- or mask-based assignment strategy.

However, it is not optimal to directly apply center-ness labels to the 3D detection task. Given that all LiDAR points are located on surfaces of objects, the center-ness labels are all very small and similar. It is almost impossible to distinguish good predictions from other points.

Instead of utilizing original representative points in point cloud, we resort to the predicted candidate points, which are supervised to be close to instance centers. Candidate points closer to instance centers tend to get more accurate localization predictions. Thus 3D center-ness labels are able to distinguish among them easily.

For each candidate point, we define its center-ness label in two steps. We first determine if it is within an instance l_{mask}, which is a binary value. Then we draw a center-ness label according to its distance to 6 surfaces of its corresponding instance. The center-ness label is calculated as

$$l_{ctrness} = \sqrt[3]{\frac{min(f,b)}{max(f,b)} \times \frac{min(l,r)}{max(l,r)} \times \frac{min(t,d)}{max(t,d)}}, \quad (2)$$

where (f, b, l, r, t, d) represent the distance to front, back, left, right, top and bottom surfaces respectively. The final classification label is the multiplication of l_{mask} and $l_{ctrness}$.

3.3. Loss Function

The overall loss is composed of classification loss, regression loss and shifting loss, as

$$\begin{aligned} L = &\frac{1}{N_c}\sum_i L_c(s_i, u_i) + \lambda_1 \frac{1}{N_p}\sum_i [u_i > 0] L_r \\ &+ \lambda_2 \frac{1}{N_p^*} L_s, \end{aligned} \quad (3)$$

where N_c and N_p are the numbers of total candidate points and positive candidate points for foreground instances. In the classification loss, we denote s_i and u_i as the predicted classification score and center-ness label for point i respectively and use cross entropy loss as L_c.

The regression loss L_r includes distance regression loss L_{dist}, size regression loss L_{size}, angle regression loss L_{angle}, and corner loss L_{corner}. We utilize the smooth-l_1 loss for L_{dist} and L_{size}, in which the targets are offsets from candidate points to their corresponding instance centers and sizes of corresponding instances respectively.

Angle regression loss contains orientation classification loss and residual prediction loss as

$$L_{angle} = L_c(d_c^a, t_c^a) + D(d_r^a, t_r^a), \quad (4)$$

where d_c^a and d_r^a are predicted angle class and residual, while t_c^a and t_r^a are their targets. Corner loss is the distance between the predicted 8 corners and assigned ground-truth, expressed as

$$L_{corner} = \sum_{m=1}^{8} \|P_m - G_m\|, \quad (5)$$

where P_m and G_m are the location of ground-truth and prediction for point m.

As for the shifting loss L_s, which is the supervision of shifts prediction in CG layer, we utilize a smooth-l_1 loss to calculate the distance between the predicted shifts and residuals from representative points to their corresponding instance centers. N_p^* is the amount of positive representative points from F-FPS.

4. Experiments

We evaluate our model on two datasets. They are the widely adopted KITTI Object Detection Benchmark [6, 7], and the larger and more complex nuScenes dataset [3].

4.1. KITTI

There are 7,481 training images/point clouds and 7,518 test ones with three categories of Car, Pedestrian and Cyclist in the KITTI dataset. We evaluate our method on all three classes and use average precision (AP) metric to evaluate different methods. During evaluation, we follow the official KITTI evaluation protocol – that is, the IoU threshold is 0.7 for class Car and 0.5 for Pedestrian and Cyclist.

Implementation Details To align network input, we randomly choose 16k points from the entire point cloud per scene. The detail of backbone network is illustrated in Figure 3. The network is trained by ADAM [10] optimizer with an initial learning rate 0.002 and batch size 16 equally distributed on 4 GPU cards. The learning rate is decayed by 10 at 40 epochs. We train our model for 50 epochs.

We adopt 4 different data augmentation strategies on KITTI dataset in order to prevent overfitting. First, we use the mix-up strategy [28], which randomly adds foreground instances with their inner points from other scenes to current point cloud. For each bounding box, we also rotate it following a uniform distribution $\Delta\theta_1 \in [-\pi/4, +\pi/4]$ and add a random translation $(\Delta x, \Delta y, \Delta z)$. Finally, each point cloud is randomly flipped along x-axis. We randomly rotate each point cloud around z-axis (upper-direction) and rescale it.

Main Results In Table 3, we compare our method with state-of-the-art 3D detectors on KITTI test set. Since August 2019, KITTI changes the mAP calculation criterion to using 40 recall positions rather than the 11 recall positions applied in former KITTI test server. For papers published before that time, we cannot directly cite the results, and instead re-compute them using the new mAP calculation. So there may be misalignment between the results in Table 3 and in original papers.

As illustrated in Table 3, our method outperforms all state-of-the-art voxel-based single stage detectors by a large margin on all three classes. On the main metric, *i.e.*, AP on "moderate" instances in class Car, our method outperforms SECOND [28] and PointPillars [12] by 3.61% and 5.26% respectively. Still, it retains comparable performance to state-of-the-art point-based method STD [32] with more than $2\times$ faster inference time.

Our method outperforms the two-stage methods of part-A^2 net and PointRCNN by 1.08% and 3.93% respectively. Moreover, we prove its superiority by comparing with multi-sensors methods of MMF [14] and F-ConvNet [27] – our method intriguingly achieves 2.14% and 3.18% improvement respectively. On the other two classes Pedestrian and Cyclist, our 3DSSD even goes beyond these two-stage object detectors. It outperforms STD [32] on these two classes by 1.8% and 2.51% respectively. We present several qualitative results in Figure 4.

4.2. nuScenes

nuScenes is a more challenging dataset. It contains 1,000 scenes, gathered from Boston and Singapore considering heavy traffic and highly challenging driving situations. It provides 1.4M 3D objects in 10 classes, along with object attributes and velocity. There are about 40k points per frame.

In order to predict velocity and attribute, all former methods combine points from current frame and previous frames in 0.5s, gathering about 400k points. With such a large amount of points, all previous point-based two-stage methods perform less satisfyingly than voxel-based ones due to the GPU memory limitation.

In the benchmark, a new evaluation metric called nuScenes detection score (NDS) is also presented, which is a weighted sum between mean average precision (mAP), the mean average errors of location (mATE), size (mASE), orientation (mAOE), attribute (mAAE) and velocity (mAVE). We use TP to denote the set of the five mean average errors. NDS is calculated as

$$NDS = \frac{1}{10}[5mAP + \sum_{mTP \in TP}(1 - min(1, mTP))]. \quad (6)$$

Implementation Details For each key frame, we similarly combine its points with those in previous 0.5s frames to get richer point cloud input. Then, we apply voxelization for randomly sampling point clouds to align input and keep original distribution. We randomly choose 65,536 voxels, including 16,384 from the key frame and 49,152 from others. The voxel size is $[0.1, 0.1, 0.1]$. 1 interior point is randomly selected from each voxel. We feed these 65,536 points into our point-based network.

The backbone network is illustrated in Figure 3. The training schedule is the same as the one on KITTI dataset. We only apply flip augmentation during training.

Type	Method	Modality	Car (%)			Pedestrian (%)			Cyclist (%)		
			Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
2-stage	F-PointNet [20]	RGB + LiDAR	82.19	69.79	60.59	50.53	42.15	38.08	72.27	56.12	49.01
	AVOD-FPN [11]		83.07	71.76	65.73	50.46	42.27	39.04	63.76	50.55	44.93
	F-ConvNet [27]		87.36	76.39	66.69	52.16	43.38	38.80	81.98	65.07	56.54
	PointRCNN [23]	LiDAR	86.96	75.64	70.70	47.98	39.37	36.01	74.96	58.82	52.53
	MMLab-PartA^2 [24]		87.81	78.49	73.51	53.10	43.35	40.06	79.17	63.52	56.93
	STD [32]		87.95	79.71	75.09	53.29	42.47	38.35	78.69	61.59	55.30
1-stage	SECOND [28]	LiDAR	84.65	75.96	68.71	45.31	35.52	33.14	75.83	60.82	53.67
	PointPillars [12]		82.58	74.31	68.99	51.45	41.92	38.89	77.10	58.65	51.92
	Ours		**88.36**	**79.57**	**74.55**	**54.64**	**44.27**	**40.23**	**82.48**	**64.10**	**56.90**

Table 3. 3D AP Results on KITTI test set for class Car, Pedestrian and Cyclist drawn from official Benchmark [1].

	Car	Ped	Bus	Barrier	TC	Truck	Trailer	Moto	Cons. Veh.	Bicycle	mAP
SECOND [28]	75.53	59.86	29.04	32.21	22.49	21.88	12.96	16.89	0.36	0	27.12
PointPillars [12]	70.5	59.9	34.4	33.2	29.6	25.0	20.0	16.7	4.5	1.6	29.5
Ours	**81.20**	**70.17**	**61.41**	**47.94**	**31.06**	**47.15**	**30.45**	**35.96**	**12.64**	**8.63**	**42.66**

Table 4. AP on nuScenes dataset. The results of SECOND come from its official implementation [2].

	mAP	mATE	mASE	mAOE	mAVE	AAE	NDS
PP [12]	29.5	0.54	**0.29**	0.45	0.29	0.41	44.9
Ours	**42.6**	**0.39**	**0.29**	**0.44**	**0.22**	**0.12**	**56.4**

Table 5. NDS on nuScenes dataset. "PP" represents PointPillars.

Method	Easy	Moderate	Hard
VoxelNet [33]	81.97	65.46	62.85
SECOND [28]	87.43	76.48	69.10
PointPillars [12]	-	77.98	-
Ours	**89.71**	**79.45**	**78.67**

Table 6. 3D detection AP on KITTI val set of our model for "Car" compared to other state-of-the-art single-stage methods.

	D-FPS	F-FPS	FS
recall (%)	92.47	**98.45**	98.31
AP (%)	70.4	76.7	**79.4**

Table 7. Points recall and AP from different sampling methods.

	IoU	Mask	3D center-ness
without shifting (%)	70.4	76.1	43.0
with shifting (%)	78.1	77.3	**79.4**

Table 8. AP among different assignment strategies. "with shifting" means using shifts in the CG layer.

Main results We show NDSs and mAPs for different methods in Table 5, and compare their APs of each class in Table 4. As illustrated in Table 5, our method yields better performance compared to all voxel-based single-stage solutions by a large margin. It also outperforms these methods in terms of AP of each class, as illustrated in Table 4.

The results manifest that our model deals well with different objects with a large variance on scale. Even for a huge scene with many negative points, our fusion sampling strategy is still capable to gather enough positive points. In addition, better results on velocity and attribute prove that our model also better gather and separate information from different frames.

4.3. Ablation Studies

All ablation studies are conducted on KITTI dataset [6]. We follow VoxelNet [33] to split original training set to 3,717 images/scenes train set and 3,769 images/scenes val set. All "AP" results in ablation studies are calculated on "Moderate" difficulty level in class Car with 11 recall positions for fair comparison.

Results on Validation Set We report the performance on KITTI validation set and and compare it with other state-of-the-art voxel-based single-stage methods in Table 6. On the most important "moderate" difficulty level, our method outperforms by 1.47%, 2.97% and 13.99% compared to PointPillars, SECOND and VoxelNet respectively. This illustrates the vast effectiveness of our strategies.

Effect of Fusion Sampling Strategy Our fusion sampling strategy is composed of F-FPS and D-FPS. We compare points recall and AP among different sub-sampling methods in Table 7. Sampling strategy containing F-FPS yield higher points recall than that with D-FPS only.

In Figure 5, we also present visual examples to illustrate the benefit of F-FPS in fusion sampling. In addition, the fusion sampling strategy yields a much higher AP, *i.e.*, 2.7% better than the one with F-FPS only. The reason is that the fusion sampling method can gather enough negative points, which enlarge receptive fields and accomplish accurate classification results.

Effect of Shifting in CG Layer In Table 8, we compare performance when using (and not using) shifting representative points from F-FPS in CG layer. Under different assignment strategies, APs of models with shifting are all higher than those without these operations. It means if the candidate points are closer to the centers of instances, it is generally easier to retrieve their corresponding instances.

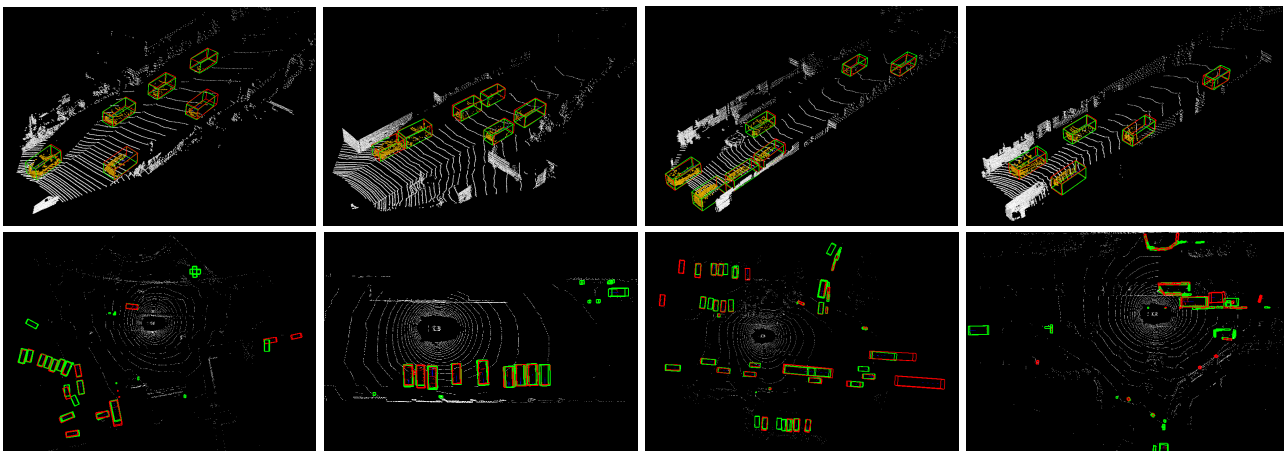

Figure 4. Visualizing results of 3DSSD on KITTI (top) and nuScenes (bottom) datasets. The ground truth and predictions are labeled in red and green respectively.

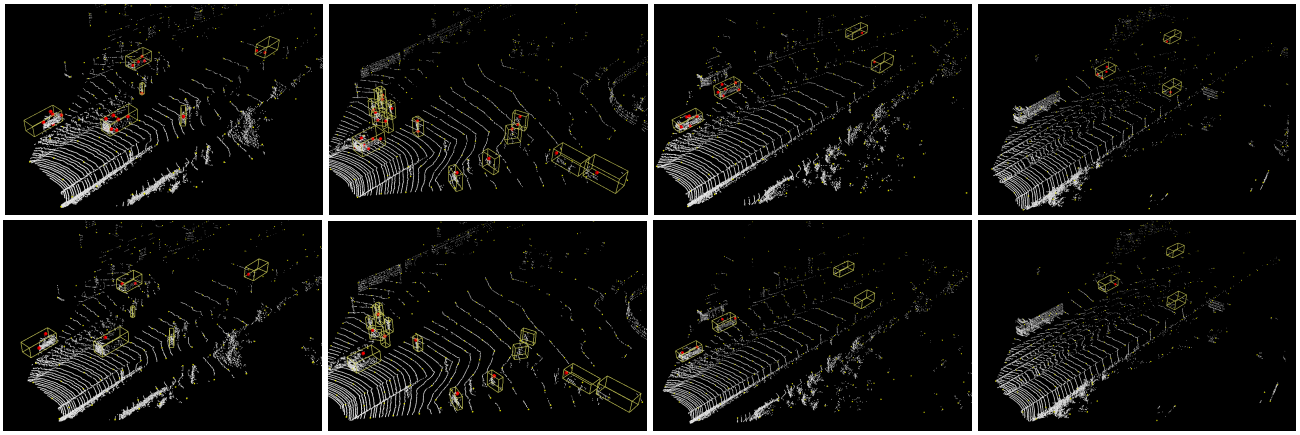

Figure 5. Comparison between representative points after fusion sampling (top) and D-FPS only (bottom). The whole point cloud and all representative points are colored in white and yellow respectively. Positive representative points are shown in red.

	F-PointNet [20]	PointRCNN [23]	STD[32]	Ours
time (ms)	170	100	80	**38**

Table 9. Inference time among different point-based methods.

Effect of 3D Center-ness Assignment We compare performance of different assignment strategies including IoU, mask and 3D center-ness label. As shown in Table 8, with the shifting operation, the model using center-ness label gains better performance than the other two strategies.

Inference Time The total inference time of 3DSSD is 38ms, tested on KITTI dataset with a Titan V GPU. We compare inference time between 3DSSD and all existing point-based methods in Table 9. As illustrated, our method is much faster than all these methods.

It is noteworthy that our method even keeps a level of similar inference speed compared to state-of-the-art voxel-based single-stage methods. For example, SECOND uses 40ms in inference while ours is 38ms. Among all existing methods, ours is only slower than PointPillars, which has been enhanced by several implementation optimization strategies, such as TensorRT, which however is not used so far in our implementation. Our method still have much potential to be further accelerated.

5. Conclusion

In this paper, as the first attempt, we have proposed a lightweight and efficient point-based 3D single-stage object detection framework. We introduced a novel fusion sampling strategy to remove the time-consuming FP layers and the refinement module, which were however needed in all existing point-based methods. In the prediction network, a candidate generation layer was designed to further reduce computation cost and utilize downsampled representative points. Our anchor-free regression head with 3D center-ness label boosted the final performance. All these effective designs enabled our model to work satisfyingly in terms of both performance and inference time.

References

[1] "kitti 3d object detection benchmark". http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d, 2019. 7

[2] "second github code". https://github.com/traveller59/second.pytorch, 2019. 7

[3] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019. 2, 3, 4, 6

[4] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, 2017. 1, 2

[5] Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *ICRA*, 2017. 1

[6] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *I. J. Robotics Res.*, 2013. 2, 6, 7

[7] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, 2012. 1, 6

[8] Alejandro González, Gabriel Villalonga, Jiaolong Xu, David Vázquez, Jaume Amores, and Antonio M. López. Multiview random forest of local experts combining RGB and LIDAR data for pedestrian detection. In *IV*, 2015. 1

[9] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, 2018. 2

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, 2014. 6

[11] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Lake Waslander. Joint 3d proposal generation and object detection from view aggregation. *CoRR*, 2017. 1, 2, 7

[12] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. *CVPR*, 2019. 1, 2, 6, 7

[13] Bo Li. 3d fully convolutional network for vehicle detection in point cloud. In *IROS*, 2017. 2

[14] Ming Liang*, Bin Yang*, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *CVPR*, 2019. 2, 6

[15] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. In *ECCV*, 2016. 5

[16] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS*, 2015. 1

[17] Youngmin Park, Vincent Lepetit, and Woontack Woo. Multiple 3d object tracking for augmented reality. In *ISMAR*, 2008. 1

[18] Cristiano Premebida, João Carreira, Jorge Batista, and Urbano Nunes. Pedestrian detection combining RGB and dense LIDAR data. In *ICoR*, 2014. 1

[19] Charles R. Qi, Or Litany, Kaiming He, and Leonidas J. Guibas. Deep hough voting for 3d object detection in point clouds. *ICCV*, 2019. 4

[20] Charles Ruizhongtai Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from RGB-D data. *CoRR*, 2017. 2, 5, 7, 8

[21] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 1, 2

[22] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, 2017. 1, 2

[23] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019. 1, 2, 3, 7, 8

[24] Shaoshuai Shi, Zhe Wang, Xiaogang Wang, and Hongsheng Li. Part-aˆ2 net: 3d part-aware and aggregation neural network for object detection from point cloud. *arXiv preprint arXiv:1907.03670*, 2019. 7

[25] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. FCOS: fully convolutional one-stage object detection. *ICCV*, 2019. 5

[26] Dominic Zeng Wang and Ingmar Posner. Voting for voting in online point cloud object detection. In *Robotics: Science and Systems XI*, 2015. 1, 2

[27] Zhixin Wang and Kui Jia. Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. In *IROS*. IEEE, 2019. 6, 7

[28] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 2018. 1, 2, 6, 7

[29] Bin Yang, Wenjie Luo, and Raquel Urtasun. PIXOR: real-time 3d object detection from point clouds. In *CVPR*, 2018. 1

[30] Ze Yang, Shaohui Liu, Han Hu, Liwei Wang, and Stephen Lin. Reppoints: Point set representation for object detection. *ICCV*, 2019. 5

[31] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. IPOD: intensive point-based object detector for point cloud. *CoRR*, 2018. 1, 2, 3

[32] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. STD: sparse-to-dense 3d object detector for point cloud. *ICCV*, 2019. 1, 2, 3, 6, 7, 8

[33] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *CoRR*, 2017. 1, 2, 7