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

Zetong Yang[†] Yanan Sun[‡] Shu Liu[†] Jiaya Jia[†]

[†]The Chinese University of Hong Kong [‡]The Hong Kong University of Science and Technology

{tomztyang, now.syn, liushuhust}@gmail.com leojia@cse.cuhk.edu.hk

Abstract

Currently, there have been many kinds of voxel-based 3D single stage detectors, while point-based single stage methods are still underexplored. In this paper, we first present a lightweight and effective point-based 3D single stage object detector, named 3DSSD, achieving a good balance between accuracy and efficiency. In this paradigm, all upsampling layers and refinement stage, which are indispensable in all existing point-based methods, are abandoned to reduce the large computation cost. We novelly 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, an anchor-free regression head with a 3D center-ness assignment strategy is designed to meet with our demand of accuracy and speed. Our paradigm is an elegant single stage anchor-free framework, showing great superiority to other existing methods. We evaluate 3DSSD 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 has comparable performance to two stage point-based methods as well, with inference speed more than 25 FPS, 2× faster than former state-of-the-art point-based methods.

1. Introduction

In recent years, 3D scene understanding has attracted more and more attention in computer vision since it benefits many real life applications, like autonomous driving [8] and augmented reality [20]. In this paper, we focus on one of the fundamental tasks in 3D scene recognition, *i.e.*, 3D object detection, which predicts 3D bounding boxes and class labels for each instance within a point cloud.

Although great breakthroughs have been made in 2D detection, it is inappropriate to translate these 2D methods to 3D directly because of the unique characteristics of point clouds. Compared to 2D images, point clouds are sparse, unordered and locality sensitive, making it impossible to use convolution neural networks (CNNs) to parse them. Therefore, how to convert and utilize raw point cloud data has become the primary problem in the detection task.

Some existing methods convert point clouds from sparse formation to compact representations by projecting them to images [4, 12, 9, 21, 6], or subdividing them to equally distributed voxels [19, 29, 36, 32, 31, 13]. We call these methods voxel-based methods, which conduct voxelization on the whole point cloud. Features in each voxel are generated by either PointNet-like backbones [24, 25] or hand-crafted features. Then many 2D detection paradigms can be applied on the compact voxel space without any extra efforts. Although these methods are straightforward and efficient, they suffer from information loss during voxelization and encounter performance bottleneck.

Another stream is point-based methods, like [34, 35, 26]. 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, they first utilize *set abstraction* (SA) layers 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 as the second stage to give final predictions. These methods achieve better performance, but their inference time is usually intolerable in many real-time systems.

Our Contributions Different from all previous methods, we first develop a lightweight and efficient point-based 3D single stage object detection framework. We observe that in point-based methods, FP layers and the refinement stage consume half of the inference time, motivating us to remove these two modules. However, it is non-trivial to abandon FP layers. Since under the current sampling strategy in SA, *i.e.*, furthest point sampling based on 3D Euclidean distance (D-FPS), foreground instances with few interior points may lose all points after sampling. Consequently, it is impossible for them to be detected, which leads to a huge performance drop. In STD [35], without upsampling, *i.e.*, conducting detection on remaining downsampled points, its performance drops by about 9%. That is the reason why FP layers must be adopted for points upsampling, although a large amount of extra computation is introduced. To deal with the dilemma, we first propose a novel sampling strategy based on feature distance, called F-FPS, which effectively preserves interior points of various instances. Our final sampling strategy is a fusion version of F-FPS and D-FPS.

To fully exploit the *representative points* retained after SA layers, we design a delicate 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 these representative points and the 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, so as to retrieve more precise localization predictions.

We eveluate our method on widely used KITTI [7] 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, achieving comparable performance to all two stage point-based methods at a much faster inference speed. In summary, our primary contribution is manifold.

- We first propose a lightweight and effective point-based 3D single stage object detector, named 3DSSD. In our paradigm, we remove FP layers and the refinement module, which are indispensible in all existing point-based methods, contributing to huge deduction on inference time of our framework.
- A novel fusion sampling strategy in SA layers is developed to keep adequate interior points of different foreground instances, which preserves rich information for regression and classification.
- We design a delicate box prediction network, making our framework both effective and efficient further. Experimental results on KITTI and nuScenes dataset show that our framework outperforms all single stage methods, and has comparable performance to state-of-the-art two stage methods with a much faster speed, which is 38ms per scene.

2. Related Work

3D Object Detection with Multiple Sensors There are several methods exploiting how 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 image, front view and BEV are then sent to the second stage to predict final bounding boxes. AVOD [12] extends MV3D by introducing image features in the proposal generation stage. MMF [16] 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 There are mainly two streams of methods dealing with 3D object detection only using LiDAR data. One is voxel-based, which applies voxelization on the entire point cloud. The difference among voxel-based methods lies on the initialization of voxel features. In [29], each non-empty voxel is encoded with 6 statistical quantities by the points within this voxel. Binary encoding is used in [15] for each voxel grid. VoxelNet [36] utilizes PointNet [24] to extract features of each voxel. Compared to [36], SECOND [31] applies sparse convolution layers [10] for parsing the compact representation. PointPillars [13] treats pseudo-images as the representation after voxelization.

Another one is point-based. They take raw point cloud as input, and generate predictions based on each point. F-PointNet [23] and IPOD [34] adopt 2D mechanisms like detection or segmentation to filter most useless points, and generate predictions from these kept useful points. PointR-CNN [26] utilizes a PointNet++ [25] 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, but their unbearable inference time makes it impossible to be applied in real-time autonomous driving system. STD [35] tries to take advantages of both point-based 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 representations and utilizes CNNs in the refinement module. Although it is faster than all former point-based methods, it is still much slower than voxel-based methods. As analyzed before, all point-based methods are composed of two stages, which are proposal generation module including SA layers and FP layers, and a refinement module as the second stage for accurate predictions. In this paper, we propose to remove FP layers and the refinement module so as to speed up point-based methods.

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

3.1. Fusion Sampling

Motivation Currently, 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. We observe that all current point-based methods [35, 26, 34] are composed of two stages including proposal generation stage and prediction refinement stage. 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 so as to recover all points. In the second stage, a refinement module optimizes proposals from RPN to get more accurate predictions. SA layers are necessary for extracting features of points, but FP layers and the refinement module indeed limit the efficiency of point-based methods, as illustrated in Table 1. Therefore, we are motivated to design a lightweight and effective point-based single stage detector.

Challenge However, it is non-trivial to remove FP layers. As mentioned before, 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 is 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 points, like ground points, due to its large amount. In other words, there are several foreground instances which are totally erased through this process, making them impossible to be detected.

With a limit of total representative points number N_m, for some remote instances, their inner points are not likely to be selected, because the amount of them is much smaller than that of background points. The situation becomes even worse on more complex datasets like nuScenes [3] dataset. 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 illustrated in the first row of Table 2, with 1024 or 512 representative points, their points recalls are only 65.9% or 51.8% respectively, which means nearly half of instances are totally erased, that is, cannot be detected. To avoid this circumstance, most of existing methods apply FP layers to recall those abandoned useful points during downsampling, but they have to pay the overhead of computation with longer inference time.

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

Table 1. Running time of difference components in our reproduced PointRCNN [26] model, which is composed of 4 SA layers and 4 FP layers for feature extraction, and a refinement module with 3 SA layers for prediction.

Methods	4096	1024	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. “4096”, “1024” and “512” represent the amount of representative points in the subset.

Feature-FPS In order to preserve *positive points* (interior points within any instance) and erase those meaningless *negative points* (points locating on background), we have to 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, many similar useless negative points will be mostly removed, like massive of ground points. Even for positive points of remote objects, they can also get survived, because semantic features of points from different objects are distinct from each other.

However, only taking the semantic feature distance as the sole criterion will preserve many points within a same instance, which introduces redundancy as well. For example, given a car, there is much difference between features of points around the windows and the wheels. As a result, points around the two parts will be sampled while any point in either part is informative for regression. To reduce the redundancy and increase the diversity, we apply both spatial distance and semantic feature distance as the criterion 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 XYZ distance and L2 feature distance between two points and λ is the balance factor. We call this sampling method as Feature-FPS (F-FPS). The comparison among different λ is shown in in Table 2, which demonstrates that combining two distances together as the criterion in the downsampling operation is more powerful than only using feature distance, *i.e.*, the special case where λ equals to 0. Moreover, as illustrated

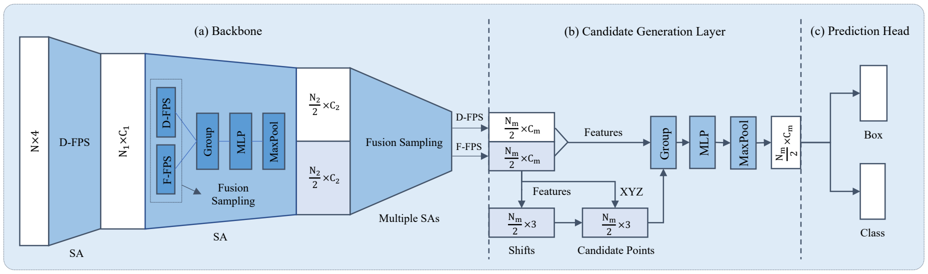

Figure 1. Illustration of the 3DSSD framework. On the whole, it is composed of backbone and box prediction network including a candidate generation layer and an anchor-free prediction head. (a) Backbone network. It takes the raw point cloud (x, y, z, r) as input, and generates global features for all 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.

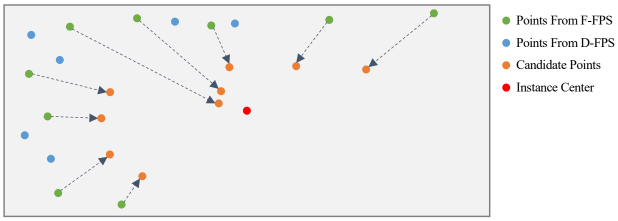

Figure 2. Illustration of the shifting operation in CG layer. The gray rectangle represents an instance with all positive representative points from F-FPS (green) and D-FPS (blue). The red dot represents instance center. We only shift points from F-FPS under the supervision of their distances to the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

in Table 2, using F-FPS with 1024 representative points and λ setting to 1 guarantees 89.2% instances can be preserved in nuScenes [3] dataset, which is 23.3% higher than D-FPS sampling strategy.

Fusion Sampling Large amount of positive points within different instances are preserved after SA layers thanks to F-FPS. However, with the limit of a fixed number of total representative points N_m, many negative points are discarded during the downsampling process, which benefits regression but hampers classification. That is, 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, the model finds difficulty in distinguishing positive and negative points, leading to a poor performance in classification. Our experiments also demonstrate this limitation in ablation study. Although the model with F-FPS has higher recall rate and better localization accuracy than the one with D-FPS, it prefers treating many negative points as positive ones, leading to a drop in classification accuracy.

As analyzed above, after a SA layer, not only positive points should be sampled as many as possible, but we also 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 keep 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 by several SA layers intertwined with fusion sampling, we gain a subset of points from both F-FPS and D-FPS, which are used for final predictions. In former point-based methods, another SA layer should be 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 advantages 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 points and useless in bounding box regression, we only use those from F-FPS as initial center points. These initial center points are shifted under the supervision of their relative locations to their corresponding instances as illustrated in Figure 2, same as VoteNet [22]. We call these new points after shifting operation as *candidate points*. Then we treat these candidate points as the center points in our CG layer. We use candidate points rather than original points as the center points 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, concatenate their normalized locations and semantic features as input, and apply MLP layers to extract features. These features will be sent to the prediction head for regression and classification. This entire process is illustrated in Figure 1.

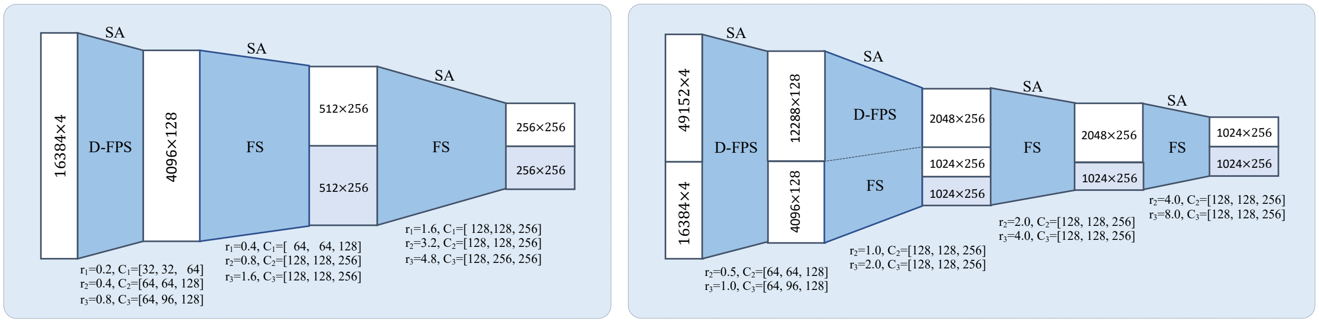
SA
SA
SA
16384×4
D-FPS
4096×128
FS
512×256
512×256
FS
256×256
256×256
r_1=0.2, C_1=[32, 32, 64]
r_2=0.4, C_2=[64, 64, 128]
r_3=0.8, C_3=[64, 96, 128]
r_1=0.4, C_1=[64, 64, 128]
r_2=0.8, C_2=[128, 128, 256]
r_3=1.6, C_3=[128, 128, 256]
r_1=1.6, C_1=[128,128, 256]
r_2=3.2, C_2=[128, 128, 256]
r_3=4.8, C_3=[128, 256, 256]
SA
SA
SA
SA
49152×4
16384×4
D-FPS
12288×128
4096×128
D-FPS
FS
2048×256
1024×256
1024×256
FS
2048×256
1024×256
FS
1024×256
1024×256
r_2=0.5, C_2=[64, 64, 128]
r_3=1.0, C_3=[64, 96, 128]
r_2=1.0, C_2=[128, 128, 256]
r_3=2.0, C_3=[128, 128, 256]
r_2=2.0, C_2=[128, 128, 256]
r_3=4.0, C_3=[128, 128, 256]
r_2=4.0, C_2=[128, 128, 256]
r_3=8.0, C_3=[128, 128, 256]

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

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 are faced with two options, anchor-based or anchor-free prediction network. If anchor-based head is adopted, we have to construct multi-scale and multi-orientation anchors so as to cover objects with variant sizes and orientations. Especially in complex scenes like those in the nuScenes dataset [3], where objects are from 10 different categories with a wide range of orientations, we need at least 20 anchors, including 10 different sizes and 2 different orientations $(0, \pi/2)$ in an anchor-based model. To avoid the cumbersome setting of multiple anchors and be consistent with our lightweight design, we utilize anchor-free regression head.

In the regression head, for each candidate point, we predict the distance (d_x, d_y, d_z) to its corresponding instance, as well as the size (d_l, d_w, d_h) and orientation of its corresponding instance. Since there is no prior orientation of each point, we apply a hybrid of classification and regression formulation following [23] in orientation angle regression. Specifically, we pre-define N_a equally split orientation angle bins and classify the proposal orientation angle into different 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, they usually use intersection-over-union (IoU) [18] threshold or mask [28, 33] to assign labels for pixels. FCOS [28] proposes a continuous center-ness label, replacing original binary classification label, to further distinguish pixels. It assigns higher center-ness scores to pixels closer to instance centers, leading to a relatively better performance compared to IoU- or mask-based assignment strategy. However, it is unsatisfying to directly apply center-ness label to 3D detection task. Given that all LiDAR points are located on surfaces of objects, their center-ness labels are all very small and similar, which makes it 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, and 3D center-ness labels are able to distinguish them easily. For each candidate point, we define its center-ness label through two steps. We first determine whether 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 number of total candidate points and positive candidate points, which are candidate points locating in foreground instance. 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}. Specifically, 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

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

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

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

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 the 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: the widely used KITTI Object Detection Benchmark [7, 8], and a larger and more complex nuScenes dataset [3].

4.1. KITTI

There are 7,481 training images/point clouds and 7,518 test images/point clouds with three categories of Car, Pedestrian and Cyclist in the KITTI dataset. We only evaluate our model on class Car, due to its large amount of data and complex scenarios. Moreover, most of state-of-the-art methods only test their models on this class. We use average precision (AP) metric to compare with different methods. During evaluation, we follow the official KITTI evaluation protocol – that is, the IoU threshold is 0.7 for class Car.

Implementation Details To align network input, we randomly choose 16k points from the entire point cloud per scene. The detail of backbone nework is illustrated in Figure 3. The network is trained by ADAM [11] optimizer with an initial learning rate of 0.002 and a batch size of 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 mix-up strategy same as [31], which randomly adds foreground instances with their inner points from other scenes to current point cloud. Then, for each bounding box, we rotate it following a uniform distribution $\Delta\theta_1 \in [-\pi/4, +\pi/4]$ and add a random translation $(\Delta x, \Delta y, \Delta z)$. Third, each point cloud is randomly flipped along x-axis. Finally, we randomly rotate each point cloud around z-axis (up axis) and rescale it.

Type	Method	Sens.	$AP_{3D}(\%)$		
			Easy	Mod	Hard
2-stage	MV3D [4]	R + L	71.09	62.35	55.12
	AVOD [12]		76.39	66.47	60.23
	F-PointNet [23]		82.19	69.79	60.59
	AVOD-FPN [12]		83.07	71.76	65.73
	IPOD [34]		80.30	73.04	68.73
	UberATG-MMF [16]		88.40	77.43	70.22
	F-ConvNet [30]		87.36	76.39	66.69
	PointRCNN [26]	L	86.96	75.64	70.70
	Fast Point-RCNN [5]		85.29	77.40	70.24
	Patches [14]		88.67	77.20	71.82
	MMLab-PartA^2 [27]		87.81	78.49	73.51
	STD [35]		87.95	79.71	75.09
1-stage	ContFuse [17]	R + L	83.68	68.78	61.67
	VoxelNet [36]	L	77.82	64.17	57.51
	SECOND [31]		84.65	75.96	68.71
	PointPillars [13]		82.58	74.31	68.99
	Ours		**88.36**	**79.57**	**74.55**

Table 3. Results on KITTI test set on class Car drawn from official Benchmark [1]. "Sens." means sensors used by the method. "L" and "R" represent using LiDAR and RGB images respectively.

Main Results As illustrated in Table 3, our method outperforms all state-of-the-art voxel-based single stage detectors by a large margin. On the main metric, *i.e.*, AP on "moderate" instances, our method outperforms SECOND [31] and PointPillars [13] by 3.61% and 5.26% respectively. Still, it retains comparable performance to the state-of-the-art point-based method STD [35] with a more than 2× faster inference time. Our method still outperforms other two stage methods like part-A^2 net and PointRCNN by 1.08% and 3.93% respectively. Moreover, it also shows its superiority when compared to multi-sensors methods, like MMF [16] and F-ConvNet [30], which achieves about 2.14% and 3.18% improvements respectively. We present several qualitative results in Figure 4.

4.2. nuScenes

The nuScenes dataset is a more challenging dataset. It contains 1000 scenes, gathered from Boston and Singapore due to their dense traffic and highly challenging driving situations. It provides us with 1.4M 3D objects on 10 different classes, as well as their attributes and velocities. There are about 40k points per frame, and in order to predict velocity and attribute, all former methods combine points from key frame and frames in last 0.5s, leading to about 400k points. Faced with such a large amount of points, all point-based two stage methods perform worse than voxel-based methods on this dataset, due to the GPU memory limitation. In

	Car	Ped	Bus	Barrier	TC	Truck	Trailer	Moto	Cons. Veh.	Bicycle	mAP
SECOND [31]	75.53	59.86	29.04	32.21	22.49	21.88	12.96	16.89	0.36	0	27.12
PointPillars [13]	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 [13]	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.

the benchmark, a new evaluation metric called nuScenes detection score (NDS) is 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, and NDS is calculated by

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

Implementation Details For each key frame, we combine its points with points in frames within last 0.5s so as to get a richer point cloud input, just the same as other methods. Then, we apply voxelization for randomly sampling point clouds, so as to align input as well as keep original distribution. We randomly choose 65536 voxels, including 16384 voxels from key frame and 49152 voxels from other frames. The voxel size is $[0.1, 0.1, 0.1]$, and 1 interior point is randomly selected from each voxel. We feed these 65536 points into our point-based network.

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

Main results We show NDS and mAP among different methods in Table 5, and compare their APs of each class in Table 4. As illustrated in Table 5, our method draws better performance compared to all voxel-based single stage methods by a large margin. Not only on mAP, it also outperforms those methods on AP of each class, as illustrated in Table 4. The results show that our model can deal with different objects with a large variance on scale. Even for a huge scene with many negative points, our fusion sampling strategy still has the ability to gather enough positive points out. In addition, better results on velocity and attribute illustrate that our model can also discriminate information from different frames.

Method	Easy	Moderate	Hard
VoxelNet [36]	81.97	65.46	62.85
SECOND [31]	87.43	76.48	69.10
PointPillars [13]	-	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.

4.3. Ablation Studies

All ablation studies are conducted on KITTI dataset [7]. We follow VoxelNet [36] 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.

Results on Validation Set We report and compare the performance of our method on KITTI validation set with other state-of-the-art voxel-based single stage methods in Table 6. As shown, 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 superiority of our method.

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 strategies containing F-FPS have higher points recall than D-FPS only. In Figure 5, we also present some visual examples to illustrate the benefits of F-FPS to fusion sampling. In addition, the fusion sampling strategy receives a much higher AP, *i.e.*, 2.7% better than the one with F-FPS only. The reason is that fusion sampling method can gather enough negative points, which enlarges receptive fields and achieve accurate classification results.

Effect of Shifting in CG Layer In Table 8, we compare performance between with or without shifting representa-

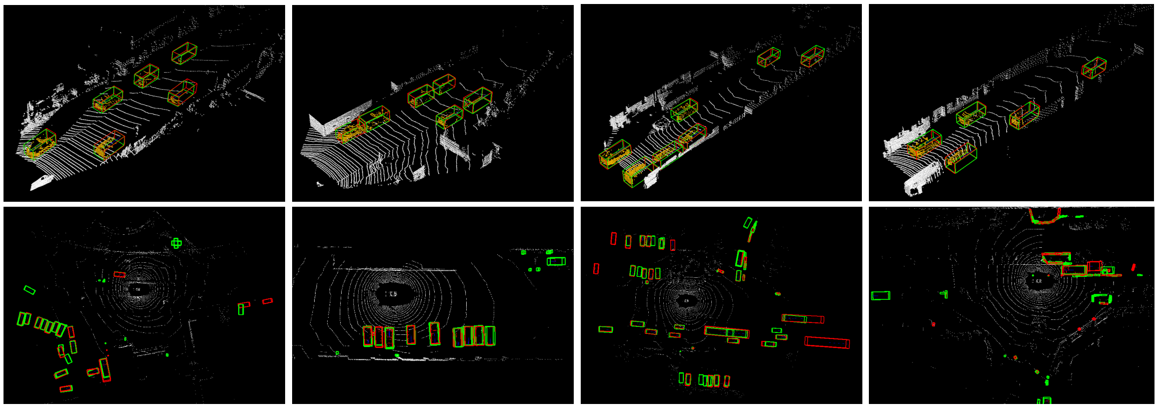

Figure 4. Qualitative results of 3DSSD on KITTI (top) and nuScenes (bottom) dataset. The groundtruths and predictions are labeled as red and green respectively.

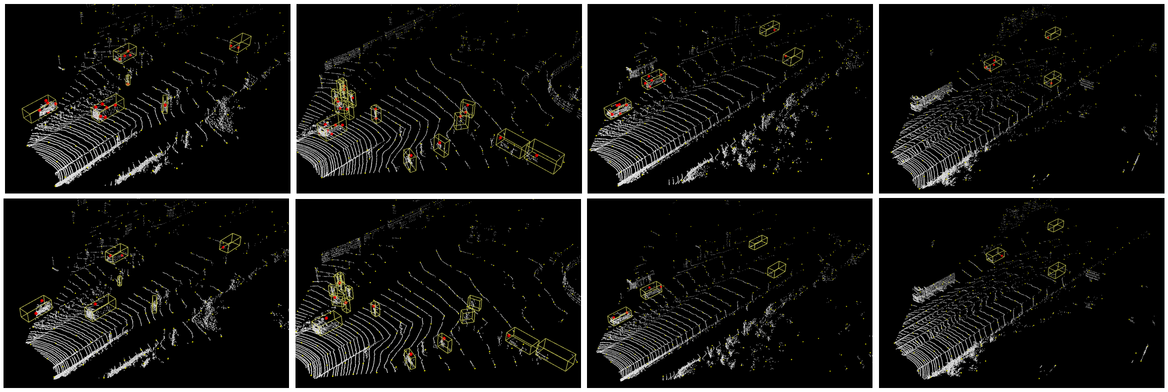

Figure 5. Comparision between representative points after fusion sampling (top) and D-FPS only (bottom). The whole point cloud and all representative points are colored in white and yellow respectively. Red points represent positive representative points.

	F-PointNet [23]	PointRCNN [26]	STD[35]	Ours
time (ms)	170	100	80	**38**

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

tive points from F-FPS in CG layer. Under different assignment strategies, APs of models with shifting are all higher than those without shifting operations. If the candidate points are closer to the centers of instances, it will be easier for them to retrieve their corresponding instances.

Effect of 3D Center-ness Assignment We compare the performance of different assignment strategies including IoU, mask and 3D center-ness label. As shown in Table 8, with 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 existing point-based methods. Moreover, our method maintains similar inference speed compared to state-of-the-art voxel-based single stage methods like SECOND which is 40ms. Among all existing methods, it is only slower than PointPillars, which is enhanced by several implementation optimizations such as TensorRT, which can also be utilized by our method for even faster inference speed.

5. Conclusion

In this paper, we first propose a lightweight and efficient point-based 3D single stage object detection framework. We introduce a novel fusion sampling strategy to remove the time-consuming FP layers and the refinement module in all existing point-based methods. In the prediction network, a candidate generation layer is designed to reduce computation cost further and fully utilize downsampled representative points, and an anchor-free regression head with 3D center-ness label is proposed in order to boost the performance of our model. All of above delicate designs enable our model to be superiority to all existing single stage 3D detectors in 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. 6

[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, 5, 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, 6

[5] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast point r-cnn. In *ICCV*, 2019. 6

[6] 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

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

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

[9] 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

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

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

[12] 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, 6

[13] 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

[14] Johannes Lehner, Andreas Mitterecker, Thomas Adler, Markus Hofmarcher, Bernhard Nessler, and Sepp Hochreiter. Patch refinement: Localized 3d object detection. *arXiv preprint arXiv:1910.04093*, 2019. 6

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

[16] 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

[17] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *ECCV*, 2018. 6

[18] 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

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

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

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

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

[23] 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, 6, 8

[24] 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

[25] 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

[26] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019. 1, 2, 3, 6, 8

[27] 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. 6

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

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

[30] 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

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

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

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

[34] 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, 6

[35] 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, 8

[36] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *CoRR*, 2017. 1, 2, 6, 7