SASA: Semantics-Augmented Set Abstraction for Point-based 3D Object Detection

Chen Chen[1], Zhe Chen[1], Jing Zhang[1], Dacheng Tao[2,1]

[1] The University of Sydney, Australia [2] JD Explore Academy, China
cche9000@uni.sydney.edu.au, {zhe.chen1, jing.zhang1}@sydney.edu.au, dacheng.tao@gmail.com

Abstract

Although point-based networks are demonstrated to be accurate for 3D point cloud modeling, they are still falling behind their voxel-based competitors in 3D detection. We observe that the prevailing set abstraction design for down-sampling points may maintain too much unimportant background information that can affect feature learning for detecting objects. To tackle this issue, we propose a novel set abstraction method named Semantics-Augmented Set Abstraction (SASA). Technically, we first add a binary segmentation module as the side output to help identify foreground points. Based on the estimated point-wise foreground scores, we then propose a semantics-guided point sampling algorithm to help retain more important foreground points during down-sampling. In practice, SASA shows to be effective in identifying valuable points related to foreground objects and improving feature learning for point-based 3D detection. Additionally, it is an easy-to-plug-in module and able to boost various point-based detectors, including single-stage and two-stage ones. Extensive experiments on the popular KITTI and nuScenes datasets validate the superiority of SASA, lifting point-based detection models to reach comparable performance to state-of-the-art voxel-based methods. Code will be available at https://github.com/blakechen97/SASA.

1 Introduction

3D object detection has attracted increasing interest from researchers because it plays an important role in various real-world scenarios like autonomous driving and the robotic system (Shi et al. 2020a; Yang et al. 2020). This task aims to identify and localize objects from 3D scenes. To properly detect objects from 3D space, LiDAR sensors are widely applied to capture 3D point clouds and represent the surrounding environments. Comparing to RGB images, point clouds provide rich and accurate 3D structure information, which is important for precise 3D object localization.

To exploit the representational power of deep learning (Zhang and Tao 2020), researchers have designed different neural networks to extract 3D features, including voxel-based ones (Zhou and Tuzel 2018; Yan, Mao, and Li 2018; Deng et al. 2020) that discretize sparse points into regular voxel grids and point-based ones (Shi, Wang, and Li 2019; Qi et al. 2019; Yang et al. 2020) that directly perform feature learning on 3D points. Benefiting from transformation-free point cloud processing and flexible receptive fields (Shi et al. 2020a), point-based methods have the potential of achieving compelling performance (Yang et al. 2020). However, compared with voxel-based detectors that show vast development, point-based 3D detection stagnates in recent years and fails to achieve top performance on related datasets.

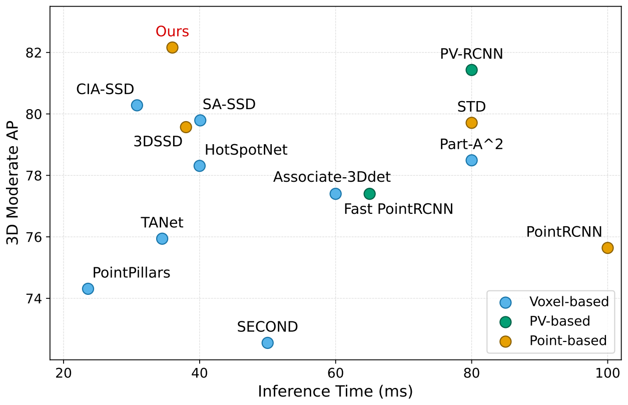


Figure 1: Our method reaches top performance (moderate AP: 82.16%) among both voxel-based and point-based detectors with a high inference speed for the car detection on the KITTI benchmark (Geiger, Lenz, and Urtasun 2012).

By investigating popular point-based methods, we find that an important problem is that the widely-used *set abstraction* (SA) is inefficient to describe scenes in the context of detection, especially with the problematic sampling strategy. In particular, the SA layer first selects a subset of input points as key points and then encodes context representations from nearby points for each sampled key point. However, when selecting key points, existing sampling strategies, which tend to choose distant points to better cover the entire scene, could make abstracted point sets involve excessive irrelevant background points like points on the ground, since the majority of the 3D space belongs to background especially in outdoor scenarios (Chen, Zhang, and Tao 2019). These irrelevant points usually deliver trivial information for detecting objects and, at the same time, plenty of beneficial foreground points could be inappropriately discarded.

For example, points on small objects like pedestrians may be completely neglected. Consequently, the point set given by SA may fail to provide sufficient foreground information or cover many foreground instances, thus the detection performance could be largely degraded. Although most point-based detectors (Qi et al. 2018; Yang et al. 2019b; Shi, Wang, and Li 2019) apply *feature propagation* (FP) layers to retrieve the foreground points dropped in the previous SA stage, these FP layers bring heavy memory usage and high computational cost (Yang et al. 2020) inevitably.

To solve the issue, we propose a *Semantics-Augmented Set Abstraction* (SASA) for point-based 3D detection. By incorporating point-wise semantic cues, we can help avoid including too many potentially irrelevant background points and focus on more informative foreground ones in the SA stage. Hence, abstracted point sets could then provide more object-related information for the succeeding box prediction network. To properly incorporate point semantics into SA, we have made the following two updates to the SA layer in Pointnet++ (Qi et al. 2017b). Firstly, we add a point binary segmentation module to identify foreground points from the input. Then, given point semantic maps, we adopt a novel sampling algorithm, *semantics-guided farthest point sampling* (S-FPS), to choose representative key points for SA layers. Comparing to the commonly used *farthest point sampling* (FPS), our proposed S-FPS gives more preference to positive points so more points from foreground are kept through down-sampling. With point-wise segmentation and advanced sampling strategy, SASA serves as a strong point feature learning technique for 3D detection.

In practice, our proposed SASA is an easy-to-plug-in module and can work seamlessly with various point-based detection frameworks. We have successfully implemented it in two popular point-based baselines, 3DSSD (Yang et al. 2020) and PointRCNN (Shi, Wang, and Li 2019). Though they use way different feature learning and box prediction schemes, SASA delivers consistent improvement. Experimental results (Sec. 4) show that SASA can boost the mean average precision (mAP) by around 2% for the most competitive car class on the KITTI dataset (Geiger, Lenz, and Urtasun 2012) and show notable improvement on the large-scale nuScenes dataset (Caesar et al. 2020).

In summary, the contribution of this work is derived from our novel point set abstraction design with semantics. For point-based 3D detection, we (a) attach a binary segmentation module to the SA layer to identify valuable points from foreground and; (b) propose a novel sampling algorithm S-FPS to make abstracted point sets focus on object areas. Our design is lightweight and can be easily adopted in manifold point-based detection models. Experimental results show that our method obtains highly boosted results on both single-stage and two-stage baselines on the KITTI (Geiger, Lenz, and Urtasun 2012) and nuScenes (Caesar et al. 2020) datasets and sets new state-of-the-art for point-based 3D object detection.

2 Related Work

3D Object Detection from Point Clouds. According to the 3D point processing schemes, recent 3D detection models can be mainly divided into grid-based and point-based methods. Grid-based methods (Chen et al. 2017; Ku et al. 2018; Song and Xiao 2016; Zhou and Tuzel 2018; Yan, Mao, and Li 2018; Chen et al. 2019; Lang et al. 2019; He et al. 2020; Shi et al. 2020b; Deng et al. 2020) firstly transform unordered 3D points into regular 2D pixels or 3D voxels where convolutional neural networks (CNN) can be applied for point cloud modeling. Some methods (Beltrán et al. 2018; Lang et al. 2019) process point clouds from projected 2D views (*e.g.* bird's eye view). VoxelNet (Zhou and Tuzel 2018) proposes to model 3D scenes via voxelization and 3D CNN. SECOND (Yan, Mao, and Li 2018) formulates an elegant 3D feature learning backbone with sparse convolutions (Liu et al. 2015) and makes a fast and effective one-stage detector. VoxelRCNN (Deng et al. 2020) proposes a novel voxel RoI pooling to efficiently aggregate RoI features from voxels in a Pointnet (Qi et al. 2017b) set abstraction style.

Another stream is point-based detection. Based on the prevailing point feature learning technique, Pointnet (Qi et al. 2017a,b), these methods model point clouds from raw points input. F-Pointnet (Qi et al. 2018) firstly introduces Pointnet (Qi et al. 2017a,b) to 3D detection for locating objects from cropped point clouds given by 2D detectors. To avoid leveraging RGB images, PointRCNN (Shi, Wang, and Li 2019) proposes a fully point-based detection paradigm, comprising a point-based region proposal network (RPN) to generate 3D proposals from point-wise features and a point-based refinement network to adjust 3D boxes with internal point features. VoteNet (Qi et al. 2019) replaces point-based RPN with a lightweight voting scheme and obtains an anchor-free point-based detector. 3DSSD (Yang et al. 2020) adopts a more advanced point sampling strategy to safely remove expensive FP layers without hurting the detection recall. Based on these popular point-based detection frameworks, we further explore how to upgrade the fundamental feature learning phase for boosting point-based detection.

Sampling Algorithms for Set Abstraction. In Pointnet-based feature learning paradigms (Qi et al. 2017b), SA layers firstly sample a subset of input points for dimension reduction, where most point-based models (Qi et al. 2018; Shi, Wang, and Li 2019; Qi et al. 2019) adopt the classic *farthest point sampling* (FPS) algorithm for key points sampling. Recent works (Yang et al. 2019a; Lang, Manor, and Avidan 2020; Yang et al. 2020; Nezhadarya et al. 2020) devise novel sampling algorithms to obtain better point modeling ability. For the representative point cloud classification task, (Yang et al. 2019a; Lang, Manor, and Avidan 2020) manage to make the sampling process differentiable so it can be optimized in an end-to-end manner. Besides, some methods choose to involve additional heuristic information into the sampling strategy. For example, Nezhadarya et al. (2020) tends to keep critical points that occupy a large proportion of channels in final representations. In 3D object detection, Yang et al. (2020) proposes the Feature-FPS (F-FPS) where the feature distance between points is also considered to increase the feature diversity of sampled points. In this paper, we use a more direct heuristic cue, point semantics, to help SA layers focus on more beneficial points from foreground.

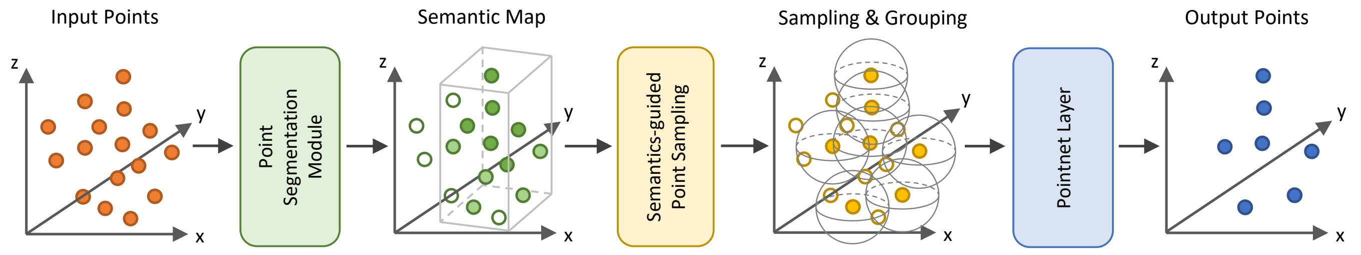


Figure 2: The structure of our proposed Semantics-Augmented Set Abstraction (SASA) layer. Based on the original SA layer design, we add a point segmentation module for mapping input point features to binary segmentation masks and update the point sampling algorithm by our semantics-guided farthest point sampling (S-FPS). Point semantic labels are derived from ground-truth boxes and all point segmentation modules are optimized with a segmentation loss function in an end-to-end manner.

3 Semantics-Augmented Set Abstraction

The overall structure of SASA is shown in Figure 2, which mainly comprises three components: a point segmentation module, a semantics-guided point sampling layer and a normal Pointnet++ SA layer.

Given the input point coordinates $\mathbf{X}$ and features $\mathbf{F}$, we first feed the point features to the point segmentation module to compute the point-wise foreground scores $\mathbf{P}$. Then, we employ our S-FPS to sample key point set $\mathbf{K}$ based on point coordinates $\mathbf{X}$ and foreground scores $\mathbf{P}$. For each point in the key point set $\mathbf{K}$, we apply the normal Pointnet++ SA layer (Qi et al. 2017b), including a point grouping operation, a multi-layer perceptron (MLP) and a max-pooling layer, to calculate high-level representations for sampled key points. The output key point coordinates and features are sent to the succeeding networks for further processing.

3.1 Point Segmentation Module

To help the Pointnet build the awareness of local semantics, we embed a lightweight point segmentation module in SASA. It is a simple 2-layer MLP and classifies input points into two categories, *i.e.* foreground and background. Specifically, denoting $\{f_1^{(l_k)}, f_2^{(l_k)}, \dots, f_{N_k}^{(l_k)}\}$ as the l_k-dimension point features fed to the k-th SA layer, the foreground score $p \in [0, 1]$ for each point is calculated as:

$$p_i = \sigma(\mathcal{M}_k(f_i^{(l_k)})) \ , \tag{1}$$

where $\mathcal{M}_k(\cdot)$ denotes the point segmentation module within the k-th SA layer, mapping input point-wise features f_i to foreground scores p_i. $\sigma(\cdot)$ is the sigmoid function.

For training the point segmentation module in each SASA layer, foreground segmentation labels for points can be naturally derived from box annotations. Similar to (Shi, Wang, and Li 2019), points inside any one of the ground-truth 3D bounding boxes are regarded as foreground points and the others as background ones. The total segmentation loss is computed with a cross entropy (CE) loss function:

$$\mathcal{L}_{seg} = \sum_{k=1}^{m} \frac{\lambda_k}{N_k} \cdot \sum_{i=1}^{N_k} \text{CE}(p_i^{[k]}, \hat{p}_i^{[k]}), \tag{2}$$

where $p_i^{[k]}$ and $\hat{p}_i^{[k]}$ denote the predicted foreground score and the ground-truth segmentation label (1 for points from foreground and 0 for ones from background) of the i-th point in the k-th SA layer. N_k and λ_k are the total number of input points and the segmentation loss weight for the k-th SA layer. The detailed parameter setting is deferred to Sec. 3.3.

3.2 Semantics-guided Farthest Point Sampling

Local semantic perception indicates hotspot locations where objects of interest may exist. Considering the goal of detecting objects, we need to pay more attention to these regions and sample more points from there. To exploit obtained point semantics in the sampling stage, a straightforward way could be directly choosing points with top-K foreground scores, but we have observed that this method selects too many points from easily identified objects, which usually have much higher foreground scores. The obtained key point set fails to cover the 3D scene and a great proportion of ground-truth objects are ignored. Thus, the overall detection performance is largely hurt.

Hence, we propose a novel point sampling algorithm, *i.e.* semantics-guided farthest point sampling (S-FPS), for incorporating the global scene awareness of FPS and the local object awareness induced by semantic heuristics. Given point-wise semantics yielded by the previous segmentation module as well as point coordinates from input, the process of our proposed S-FPS is described in Algorithm 1. The main idea is to select more foreground points by giving precedence to the points with higher foreground scores. Remaining the overall procedure of FPS unchanged, we rectify the sampling metric, distance to already-sampled points, with point foreground scores. Specifically, given 3D coordinates $\{x_1, x_2, \dots, x_N\}$ and foreground scores $\{p_1, p_2, \dots, p_N\}$ of input points, a distance array $\{d_1, d_2, \dots, d_N\}$ maintains the shortest distance from i-th point to already selected key points. In each round of selection, we add the point with highest *semantics-weighted distance* $\tilde{d}_i$ to the key point set and it is computed as:

$$\tilde{d}_i = p_i^{\gamma} \cdot d_i \ , \tag{3}$$

where γ is the balance factor controlling the importance of semantic information. It is worth noticing that S-FPS can reduce to vanilla FPS when $\gamma = 0$ and can also approximate to the aforementioned top-K selection if γ becomes extremely large.

Algorithm 1: Semantics-guided Farthest Point Sampling Algorithm. N is the number of input points and M is the number of output points sampled by the algorithm.

```
Input: coordinates $\mathbf{X} = \{x_1, \dots, x_N\} \in \mathbb{R}^{N \times 3}$;
       foreground scores $\mathbf{P} = \{p_1, \dots, p_N\} \in \mathbb{R}^N$
Output: sampled key point set $\mathbf{K} = \{k_1, \dots, k_M\}$
 1: initialize an empty sampling point set $\mathbf{K}$
 2: initialize a distance array $d$ of length $N$ with all $+\infty$
 3: initialize a visit array $v$ of length $N$ with all zeros
 4: for $i = 1$ to $M$ do
 5:     if $i = 1$ then
 6:         $k_i = \arg\max(\mathbf{P})$
 7:     else
 8:         $\mathbf{D} = \{p_k^\gamma \cdot d_k | v_k = 0\}$
 9:         $k_i = \arg\max(\mathbf{D})$
10:     end if
11:     add $k_i$ to $\mathbf{K}$, $v_{k_i} = 1$
12:     for $j = 1$ to $N$ do
13:         $d_j = \min(d_j, \|x_j - x_{k_i}\|)$
14:     end for
15: end for
16: return $\mathbf{P}$
```

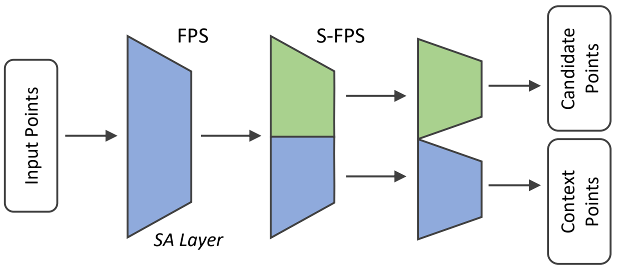
The benefit brought by S-FPS is manifold. Firstly, S-FPS can retain diverse points from foreground. Incorporated with semantic weights, positive points are more favored than negative ones during sampling, since they usually have a larger semantics-guided distance. Secondly, S-FPS enhances the density of key points in high-score areas, where foreground objects exist with higher probabilities. This could help provide more beneficial information for the follow-up box prediction network. Also, S-FPS is less sensitive to distant outliers (Yang et al. 2019a). Though outliers usually have a larger distance to other points, their low semantic weights prevent the sampling algorithm from choosing them. Lastly, S-FPS is permutation-irrelevant (Yang et al. 2019a). That is, previous sampling algorithms like FPS and F-FPS do not have a clear start point so different orders of input points may lead to different sampling outcomes. While S-FPS always starts with the point with the highest semantic score and all succeeding selections are unique. Key points sampled by S-FPS remain stable against different permutations.

3.3 Implementation Details

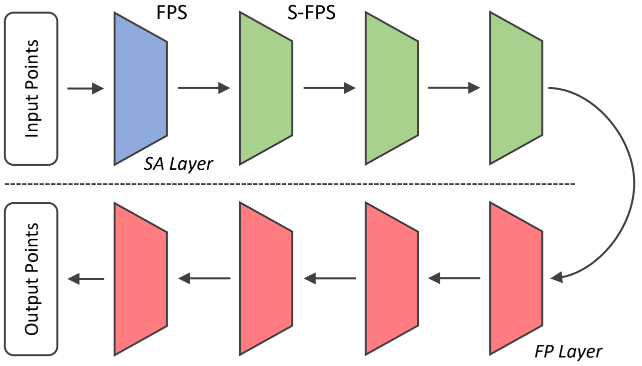
This section provides details about how we implement semantics-augmented set abstraction in 3DSSD (Yang et al. 2020) and PointRCNN (Shi, Wang, and Li 2019).

3DSSD. *3DSSD* (Yang et al. 2020) is a lightweight single-stage detector. The backbone is composed of three Pointnet SA layers only and the box prediction network is similar to VoteNet (Qi et al. 2019), where a vote point indicating the corresponding object centroid is firstly computed from candidate point features and then points in the vicinity of each vote point are aggregated to estimate the 3D box.

3DSSD introduces a *fusion sampling* strategy, where two different point sampling algorithms (namely FPS and F-



(a) 3DSSD backbone



(b) PointRCNN backbone

Figure 3: Illustration of 3DSSD and PointRCNN backbones with semantics-augmented set abstraction.

FPS) are adopted together to sample half of the total key points of the layer respectively. As shown in Figure 3a, we replace the F-FPS part with our proposed S-FPS and leave all other sampling settings (*e.g.* the number of sampled key points) unchanged. We start implementing our SASA from level 2 as the raw point input to level 1 cannot produce meaningful semantics. The segmentation loss weights for the level 2 and level 3 SA are set to 0.01 and 0.1.

In addition, we add a skip connection in each SA layer to enable low-level features better flow to the deep part of the network. Specifically, we concatenate point features before and after the Pointnet layer and aggregate hybrid features to output point representations.

PointRCNN. *PointRCNN* (Shi, Wang, and Li 2019) is a representative two-stage detection paradigm with Pointnet. The model comprises a Pointnet++ (Qi et al. 2017b) backbone, a point-based RPN and a refinement network. The backbone consists of four SA layers followed by four FP layers. Extracted point features are then fed to the RPN to filter background points and generate 3D regions of interest (RoIs) for foreground points. Finally, the refinement network gathers point features within each RoI and gives the final box estimation.

PointRCNN uses vanilla FPS to sample all key points in all SA layers. As shown in Figure 3b, we apply SASA from level 2 to level 4 and keep the backbone structure, including FP layers, the same as the original implementation. The segmentation loss weights for the three levels are set to 0.001, 0.01 and 0.1.

4 Experiments

4.1 Datasets

We validate our semantics-augmented set abstraction on the popular KITTI dataset (Geiger, Lenz, and Urtasun 2012) and the more challenging nuScenes dataset (Caesar et al. 2020).

KITTI Dataset. *KITTI dataset* (Geiger, Lenz, and Urtasun 2012) is a prevailing benchmark for 3D object detection in transportation scenarios. It contains $7,481$ LiDAR point clouds as well as finely calibrated 3D bounding boxes for training, and $7,518$ samples for testing.

Following the commonly applied setting (Zhou and Tuzel 2018), we divide all training examples into the *train* split ($3,712$ samples) and the *val* split ($3,769$ samples) and all experimental models are trained on the *train* split and tested on the *val* split. For the submission to KITTI *test* server, we follow the training protocol mentioned in (Shi et al. 2020a), where 80% labeled point cloud images are used for training and the rest 20% images are used for validation.

nuScenes Dataset. *nuScenes Dataset* (Caesar et al. 2020) is a more challenging dataset for autonomous driving with 380k LiDAR sweeps from $1,000$ scenes. It is annotated with up to 10 object categories, including 3D bounding boxes, object velocity and attributes, from the full $360°$ detection range (compared with $90°$ for KITTI). The evaluation metrics used in nuScenes dataset incorporate the commonly used mean average precision (mAP) and a novel nuScenes detection score (NDS), which reflects the overall prediction quality in multiple domains (*i.e.* detection, tracking and attribute estimation).

4.2 Experiment Settings

We have two different baselines, 3DSSD (Yang et al. 2020) and PointRCNN (Shi, Wang, and Li 2019), for evaluation. Our experimental models are all built with the OpenPCDet (Team 2020) toolbox, including our reproduced 3DSSD and the official implementation of PointRCNN.

3DSSD. We train the 3DSSD model with ADAM optimizer for 80 epochs. We apply the one-cycle learning rate schedule (Smith and Topin 2019) with the peak learning rate at 0.01. The total batch size is set to 16, equally distributed on four NVIDIA V100 GPUs.

During the training phase, manifold data augmentation strategies are employed to avoid over-fitting. We use the GT-AUG (Yan, Mao, and Li 2018; Shi, Wang, and Li 2019) to paste some instances along with their internal LiDAR points from other scenes to the current training scene. We also utilize global scene augmentations, such as random flipping along the X-axis, random scaling with a factor from $[0.9, 1.1]$ and random rotation with an angle from $[-\frac{\pi}{4}, \frac{\pi}{4}]$, as well as box-wise augmentations including random permutation, scaling and rotation. The augmentation settings are kept identical to (Yang et al. 2020). In the inference stage, we use 3D non-maximum-suppression (NMS) with the threshold of 0.01 to remove redundant predictions.

When transferring to the nuScenes dataset, we follow the official suggestion (Caesar et al. 2020) that combining LiDAR points from the current key frame as well as previous frames in 0.5s, which involves up to 400k LiDAR points in a single training sample. Then, we reduce the quantity of input LiDAR points in the same way as (Yang et al. 2020). In particular, we voxelize the point cloud from the key frame as well as that from piled previous frames with the voxel size of $(0.1\text{m}, 0.1\text{m}, 0.1\text{m})$, then randomly select $16,384$ and $49,152$ voxels from the key frame and previous frames and randomly choose one internal LiDAR point from each selected voxel. The total $65,536$ LiDAR points with 3D coordinates, reflectance and timestamp (Caesar et al. 2020) are fed to the model. The training phase runs for 20 epochs with a batch size of 16 on eight NVIDIA V100 GPUs.

PointRCNN. According to the model configuration provided in OpenPCDet (Team 2020), we train PointRCNN (Shi, Wang, and Li 2019) in an end-to-end manner with ADAM optimizer for 80 epochs. The learning rate schedule is one-cycle schedule (Smith and Topin 2019) with a peak learning rate at 0.01. We follow the original data augmentation strategies and inference settings. Please refer to (Shi, Wang, and Li 2019) and (Team 2020) for more details.

Method	Car (IoU=0.7)			Time
	Easy	Mod.	Hard	(ms)
RGB + LiDAR				
MV3D (Chen et al. 2017)	74.97	63.63	54.00	360
F-PointNet (Qi et al. 2018)	82.19	69.79	60.59	170
AVOD-FPN (Ku et al. 2018)	83.07	71.76	65.73	100
3D-CVF (Yoo et al. 2020)	89.20	80.05	73.11	75
LiDAR only				
Voxel-based:				
VoxelNet (Zhou et al. 2018)	77.47	65.11	57.73	220
SECOND (Yan et al. 2018)	83.34	72.55	65.82	50
PointPillars (Lang et al. 2019)	82.58	74.31	68.99	**23.6**
TANet (Liu et al. 2020)	83.81	75.38	67.66	34.5
Part-A^2 (Shi et al. 2020b)	87.81	78.49	73.51	80*
SA-SSD (He et al. 2020)	88.75	79.79	74.16	40.1
CIA-SSD (Zheng et al. 2020)	89.59	80.28	72.87	30.8
Voxel-RCNN (Deng et al. 2020)	**90.90**	81.62	77.06	25.2
PV-based:				
F-PointRCNN (Chen et al. 2019)	84.28	75.73	67.39	65
PV-RCNN (Shi et al. 2020a)	90.25	81.43	76.82	80*
Point-based:				
PointRCNN (Shi et al. 2019)	86.96	75.64	70.70	100*
STD (Yang et al. 2019b)	87.95	79.71	75.09	80
3DSSD (Yang et al. 2020)	88.36	79.57	74.55	38
Ours (3DSSD + **SASA**)	88.76	**82.16**	**77.16**	36

Table 1: Results on the car class of KITTI *test* set. Our model is 3DSSD with SASA. The evaluation metric is the AP calculated on 40 recall points. Inference time data with "*" is pasted from the official KITTI benchmark website.

4.3 Main Results.

Our main evaluation compared with state-of-the-art models is performed on the 3DSSD model with our proposed SASA.

Results on KITTI Dataset. Table 1 shows the 3D object detection performance on the KITTI *test* set evaluated on the official server. For the most competitive car detection race track, our method surpasses all existing point-based detectors by a great margin and obtains comparable results to state-of-the-art voxel-based models. Comparing

Method	NDS	mAP	Car	Truck	Bus	Trailer	C.V.	Ped.	Motor	Bicycle	T.C.	Barrier
PointPillars (Lang et al. 2019)	46.8	28.2	75.5	31.6	44.9	23.7	4.0	49.6	14.6	0.4	8.0	30.0
3D-CVF (Yoo et al. 2020)	49.8	42.2	79.7	37.9	55.0	36.3	-	71.3	37.2	-	40.8	47.1
3DSSD (Yang et al. 2020)	56.4	42.6	81.2	47.2	61.4	30.5	12.6	70.2	36.0	8.6	31.1	47.9
Ours (3DSSD + **SASA**)	**61.0**	**45.0**	76.8	45.0	66.2	36.5	16.1	69.1	39.6	16.9	29.9	53.6

Table 2: Results on the nuScenes *validation* set. Our model is 3DSSD with SASA. Evaluation metrics include NDS, mAP and AP on 10 classes. Abbreviations: Pedestrian (Ped.), Traffic cone (T.C.), Construction vehicle (C.V.).

Sampling Method	PS	FS	Easy	Mod.	Hard	Recall
FPS	✗	✗	91.08	82.75	79.93	92.10
FPS	✓	✗	91.17	82.83	81.97	92.01
F-FPS	✗	✓	91.54	83.46	82.18	96.65
S-FPS ($\gamma = 0.1$)	✓	✓	91.53	83.16	81.92	95.79
S-FPS ($\gamma = 1$)	✓	✓	**92.19**	**85.76**	**83.11**	**97.65**
S-FPS ($\gamma = 10$)	✓	✓	92.17	83.41	80.61	95.02
S-FPS ($\gamma = 100$)	✓	✓	91.72	82.35	78.24	91.19

Table 3: Performance comparison between FPS, F-FPS and S-FPS with different balance factor settings. "PS" represents point segmentation modules and "FS" represents the fusion sampling strategy. Point recall is calculated according to the 256 candidate points that are used to generate votes.

with the baseline model 3DSSD, our method boosts the AP by $0.40\%, 2.59\%, 2.61\%$ for the three difficulty levels respectively. It is worth noting that our method acquires significant improvements on the moderate and hard levels, demonstrating our proposed semantics-augmented operation can retain sufficient points from hardly visible instances so as to make more robust object estimations, which is of great significance in building safe autonomous driving systems.

Results on nuScenes Dataset. We report the nuScenes detection score (NDS) and the mean average precision (mAP) as well as the average precision (AP) for the 10 object categories in Table 2. Our method obtains much higher NDS and mAP compared with the baseline method 3DSSD (4.6% on NDS and 2.4% on mAP). We believe our proposed SASA efficiently chooses plenty of key points from multiple frames so as to enhance the detection accuracy as well as the tracking accuracy. Especially for bicycles that are commonly regarded as difficult detection targets, our method still produces satisfactory results.

Inference Speed. We have also observed a slightly faster inference speed (around 36ms per sample) with our proposed SASA compared with the 3DSSD baseline (38ms per sample). An important reason is that SASA bypasses the time-consuming calculation of the feature distance matrix of F-FPS. Especially when the number of points and feature dimensions go large, our point segmentation module and S-FPS require less time than the F-FPS adopted in 3DSSD baseline.

4.4 Ablation Study

We conduct ablation studies to validate each part of SASA. All results provided in this section are trained on the KITTI *train* split and evaluated on the *val* split of the car class.

Effects of Semantics-guided Point Sampling. Table 3 compares the detection performance as well as the point recall, which means that the proportion of GT boxes that have at least one internal sample point comparing to the total number of GT boxes (Yang et al. 2020), among different sampling algorithms, based on the 3DSSD baseline. We only adjust the point sampling strategy and keep other model settings identical. Results show that our S-FPS outperforms the F-FPS used in the 3DSSD baseline in all three difficulty levels, especially by up to 2.30% in the moderate level. Also, candidate points sampled by our method can "hit" 1% more ground-truth boxes comparing to F-FPS.

Visualization results in Figure 4 also prove our method effective. Comparing to F-FPS, S-FPS can keep more key points within a single instance, even for those severely occluded or tiny objects. Thus, hard examples are more likely to be detected with our proposed S-FPS sampling algorithm.

Effects of Point Segmentation Layer. The 1^{st} row and the 2^{nd} row of Table 3 compare the detection performance with and without point segmentation modules. Stand-alone segmentation layers show limited effects on the detection accuracy. The improvement is mainly derived from the point sampling algorithm.

Effects of Semantics Balance Factor. We also compare S-FPS with different balance factor γ from the 4^{th} to 7^{th} row in Table 3. Results indicate that an overly large or small γ could not appropriately boost the detection accuracy. As aforementioned, S-FPS will approximate to the top-K selection on foreground scores if the γ becomes extremely large. Sampled key points could crowd in a minority of easily identified instances and fail to cover distant or occluded ones. When $\gamma = 100$, the point recall drops sharply to 91.19%, even worse than that for vanilla FPS. Also, the box prediction network would encounter the imbalance training problem as the quantity of internal sampled points shows a great disparity among objects. Therefore, the overall detection performance is largely hurt. From the other aspect, S-FPS will degrade to vanilla FPS if γ is close to 0, making limited improvement. A suitable γ could significantly improve the performance. When $\gamma = 1$, the three difficulty levels reach satisfactory performance simultaneously.

4.5 Compatibility Study

Our SASA is an easy-to-plug-in design and can serve multiple point-based detection paradigms. As SASA already obtains notable enhancement on the one-stage model 3DSSD (as shown in Table 4), here we test its compatibility in a two-stage point-based detector, PointRCNN.

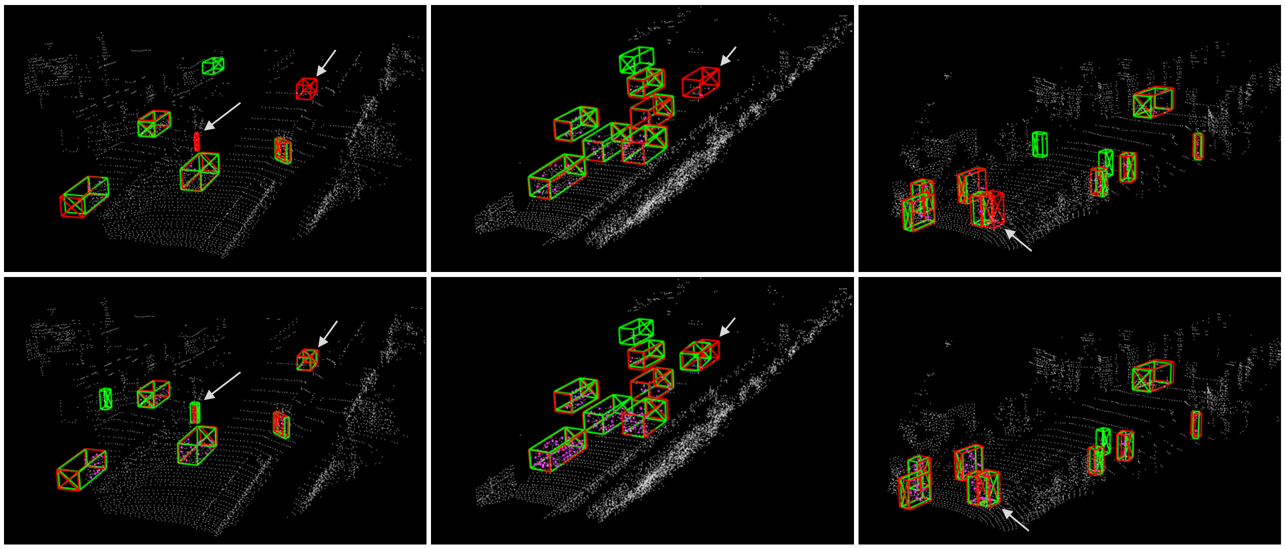


Figure 4: Visualizing detection results between F-FPS (*top*) and S-FPS (*bottom*) on the KITTI *val* split. Predicted and ground-truth boxes are labeled in green and red, respectively. Pink points mark the key points sampled in the last SA layer and white arrows indicate some false negative examples for F-FPS but successfully recovered by our method.

Method	Easy	Mod.	Hard	mAP
3DSSD	91.54	83.46	82.18	85.73
3DSSD + **SASA**	92.19	85.76	83.11	87.02
SASA Improvement	*+0.65*	*+2.30*	*+0.93*	*+1.29*
PointRCNN	91.57	82.24	80.45	84.75
PointRCNN + **SASA**	92.13	82.64	82.40	85.72
SASA Improvement	*+0.56*	*+0.40*	*+1.95*	*+0.97*

Table 4: Effects of SASA in different point-based detection paradigms evaluated on the car class of KITTI *val* split.

Results on KITTI Dataset. Although the PointRCNN backbone contains four FP layers to recover points discarded in the SA stage, there is no concern about the point recall rate. Nevertheless, SASA can still enhance the detection performance. As shown in Table 4, it improves the detection performance for all difficulty levels, especially with a 1.95% AP leading in the hard mode. Harder instances hold fewer LiDAR points, which are usually difficult to survive in deep SA layers with vanilla FPS. Their features could not be deeply and sufficiently encoded by the Pointnet backbone, while S-FPS can better focus on these point samples and their feature quality is largely improved.

Quantitative Analysis. Here we further analyze the quality of produced point features by comparing the accuracy of 3D proposals generated by RPN. From Figure 5, point features extracted by our semantics-augmented backbone can yield more accurate 3D RoIs compared to the original one. When we use the top-100 RoIs as PointRCNN suggests, our method can cover nearly 2% more ground-truth boxes at the IoU level of 0.7. This gap widens to almost 10% when the number of RoIs becomes lower, demonstrating the superiority of the our feature learning scheme with SASA.

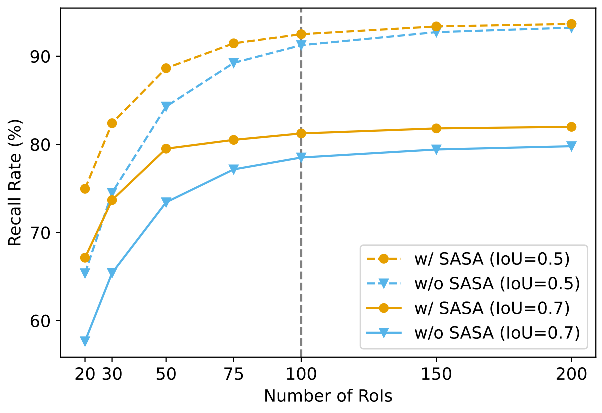


Figure 5: Proposal recall rate under different numbers of RoIs with and without SASA on PointRCNN.

5 Conclusion

In this paper, we present the Semantics-Augmented Set Abstraction (SASA) for point-based 3D detection. Our main concept is to incorporate semantic information into the Pointnet SA stage for guiding the point-based backbone to better model potential objects. Experimental results on the KITTI and nuScenes datasets indicate that our strategy can help access a higher point recall during the point down-sampling stage so as to obtain a better detection outcome for manifold point-based detectors. Our proposed method provides a promising direction for point-based detection. Not only can it be implemented in Pointnet-based models, but it is also compatible with, for example, transformer-based networks and graph neural networks for model reduction. We hope this study could inspire the research community to further break the sampling bottleneck in point-based detection.

6 Acknowledgements

Dr Zhe Chen is supported by Australian Research Council Project IH-180100002 and Dr Jing Zhang is supported by Australian Research Council Project FL-170100117.

References

Beltrán, J.; Guindel, C.; Moreno, F. M.; Cruzado, D.; Garcia, F.; and De La Escalera, A. 2018. Birdnet: a 3d object detection framework from lidar information. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 3517–3523. IEEE.

Caesar, H.; Bankiti, V.; Lang, A. H.; Vora, S.; Liong, V. E.; Xu, Q.; Krishnan, A.; Pan, Y.; Baldan, G.; and Beijbom, O. 2020. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 11621–11631.

Chen, X.; Ma, H.; Wan, J.; Li, B.; and Xia, T. 2017. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1907–1915.

Chen, Y.; Liu, S.; Shen, X.; and Jia, J. 2019. Fast point r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9775–9784.

Chen, Z.; Zhang, J.; and Tao, D. 2019. Progressive lidar adaptation for road detection. *IEEE/CAA Journal of Automatica Sinica*, 6(3): 693–702.

Deng, J.; Shi, S.; Li, P.; Zhou, W.; Zhang, Y.; and Li, H. 2020. Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection. *arXiv preprint arXiv:2012.15712*.

Geiger, A.; Lenz, P.; and Urtasun, R. 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 3354–3361. IEEE.

He, C.; Zeng, H.; Huang, J.; Hua, X.-S.; and Zhang, L. 2020. Structure aware single-stage 3d object detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11873–11882.

Ku, J.; Mozifian, M.; Lee, J.; Harakeh, A.; and Waslander, S. L. 2018. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1–8. IEEE.

Lang, A. H.; Vora, S.; Caesar, H.; Zhou, L.; Yang, J.; and Beijbom, O. 2019. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12697–12705.

Lang, I.; Manor, A.; and Avidan, S. 2020. SampleNet: differentiable point cloud sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7578–7588.

Liu, B.; Wang, M.; Foroosh, H.; Tappen, M.; and Pensky, M. 2015. Sparse convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 806–814.

Liu, Z.; Zhao, X.; Huang, T.; Hu, R.; Zhou, Y.; and Bai, X. 2020. TANet: Robust 3D Object Detection from Point Clouds with Triple Attention. *ArXiv*, abs/1912.05163.

Nezhadarya, E.; Taghavi, E.; Razani, R.; Liu, B.; and Luo, J. 2020. Adaptive hierarchical down-sampling for point cloud classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12956–12964.

Qi, C. R.; Litany, O.; He, K.; and Guibas, L. J. 2019. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, 9277–9286.

Qi, C. R.; Liu, W.; Wu, C.; Su, H.; and Guibas, L. J. 2018. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 918–927.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 652–660.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, 5099–5108.

Shi, S.; Guo, C.; Jiang, L.; Wang, Z.; Shi, J.; Wang, X.; and Li, H. 2020a. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10529–10538.

Shi, S.; Wang, X.; and Li, H. 2019. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 770–779.

Shi, S.; Wang, Z.; Shi, J.; Wang, X.; and Li, H. 2020b. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *IEEE transactions on pattern analysis and machine intelligence*.

Smith, L. N.; and Topin, N. 2019. Super-convergence: Very fast training of neural networks using large learning rates. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006, 1100612. International Society for Optics and Photonics.

Song, S.; and Xiao, J. 2016. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 808–816.

Team, O. D. 2020. OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds. https://github.com/open-mmlab/OpenPCDet.

Yan, Y.; Mao, Y.; and Li, B. 2018. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10): 3337.

Yang, J.; Zhang, Q.; Ni, B.; Li, L.; Liu, J.; Zhou, M.; and Tian, Q. 2019a. Modeling point clouds with self-attention and gumbel subset sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3323–3332.

Yang, Z.; Sun, Y.; Liu, S.; and Jia, J. 2020. 3dssd: Point-based 3d single stage object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11040–11048.

Yang, Z.; Sun, Y.; Liu, S.; Shen, X.; and Jia, J. 2019b. Std: Sparse-to-dense 3d object detector for point cloud. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1951–1960.

Yoo, J. H.; Kim, Y.; Kim, J. S.; and Choi, J. W. 2020. 3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object detection. *arXiv preprint arXiv:2004.12636*, 3.

Zhang, J.; and Tao, D. 2020. Empowering things with intelligence: a survey of the progress, challenges, and opportunities in artificial intelligence of things. *IEEE Internet of Things Journal*, 8(10): 7789–7817.

Zheng, W.; Tang, W.; Chen, S.; Jiang, L.; and Fu, C.-W. 2020. CIA-SSD: Confident IoU-Aware Single-Stage Object Detector From Point Cloud. *arXiv preprint arXiv:2012.03015*.

Zhou, Y.; and Tuzel, O. 2018. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4490–4499.