3DIoUMatch: Leveraging IoU Prediction for Semi-Supervised 3D Object Detection Supplementary Material

He Wang[1*] Yezhen Cong[2*] Or Litany[3] Yue Gao[2] Leonidas J. Guibas[1]
[1]Stanford University [2]Tsinghua University [3]NVIDIA

Abstract

In this supplementary material, we provide more implementation details of our 3DIoUMatch method coupled with VoteNet and PV-RCNN, per-category results on ScanNet and SUN-RGBD, and result visualizations.

1. 3D IoU Estimation Module for VoteNet

To facilitate the rejection of poorly localized proposals, as well as guiding deduplication and test-time refinement, we devise a new 3D IoU estimation module differentiable w.r.t bounding box parameters for point-cloud-based IoU-unaware detectors like VoteNet.

In detail, for each predicted bounding box $b^{(k)}$, we wish to estimate its 3D IoU $v^{(k)} \in [0,1]$ with respect to its corresponding ground-truth box $\{o^k \in \{o^{(j)}\} | k = \text{argmax}_j(\text{IoU}(b^{(k)}, o^{(j)}))\}$. VoteNet does not have intermediate region proposals and only output bounding box parameters at the end stage. Features used for bounding box parameter regression are gathered from vote points in a fixed-radius ball vicinity around each vote cluster, which are unaware of the final bounding box prediction. So, different from implementation in IoU-Net [2] that parallel the bounding refinement and IoU estimation, we need to do it serially by pooling features again specifically for 3D IoU estimation using the final predicted bounding box.

This feature pooling layer takes a bounding box as input and should generate continuous features with respect to the change in bounding box parameters. Existing RoI pooling methods proposed in GSPN [7] and PointRCNN [5] and 3D IoU module proposed in STD [6] simply set a hard cropping boundary at the bounding box surface, taking the point features inside the proposal and ignoring any points outside. These designs have poor differentiability and cause discontinuities whenever a change in the box parameters modifies the point population inside the box, thus are not suitable for 3D IoU optimization (see Table 4).

Here, for the first time, we devise a 3D pooling layer, 3D Grid Pooling, that is differentiable with respect to the change in all bounding box parameters. Inspired by RoIAlign [1] in 2D object detection, we propose to construct virtual grid points spanning the space of the bounding box and their features are obtained by distance-weighted interpolation from real points not restricted inside the box.

Network architecture for IoU-aware VoteNet Taking as inputs the seed points $\{z_i\}_{i=1}^M$, predicted bounding box $b = \{\mathbf{c}, \mathbf{d}, \theta\}$, and a predicted label l, our 3D IoU module estimates the largest 3D IoU between B and all ground truth bounding boxes. Following IoU-Net [2], the IoU estimation is class-aware.

To build a differentiable 3D IoU module, we first construct a $D \times D \times D$ grid $\{g_m \in \mathbb{R}^3 \mid m \in [0, D^3-1]\}$ that exactly span over the space of b and evenly divide its width, length, and height. For each grid point g_m, we find its k nearest neighbours among all seed points and interpolate their features to get $f_m = \frac{\Sigma_{i=1}^k w_i f_i}{\Sigma_{i=1}^k w_i}$, where $w_i = \frac{1}{d(g_m, g_i)^2}$ and d is the L2 distance. Ideally, if k is equal to the number of all seed points, then the IoU module is continuously differentiable. Due to the computational cost, we empirically find $k = 3$ is sufficient for accurate 3D IoU estimation with smooth gradients. We then concatenate g_m and f_m for each grid point and form a grid feature set $\{[g_m; f_m]\}$. The feature set will be pushed towards a PointNet to predict class-aware 3D IoU. A final 3D IoU selection will be performed using the input class label.

Our IoU-aware VoteNet shares the same structure with VoteNet[3] except for the IoU estimation module. We provide a more detailed description of the IoU estimation module here. The IoU estimation module is appended after the proposal generation module of VoteNet and takes the bounding box proposals as input. For each bounding box proposal, we create $4 \times 4 \times 4$ virtual grid points. We obtain the relative coordinates of the grid points by subtracting the coordinates of the bounding box center. For every grid point we find its k nearest neighbours among all seed points and interpolate their features to get $f_m = \frac{\Sigma_{i=1}^k w_i f_i}{\Sigma_{i=1}^k w_i}$, where

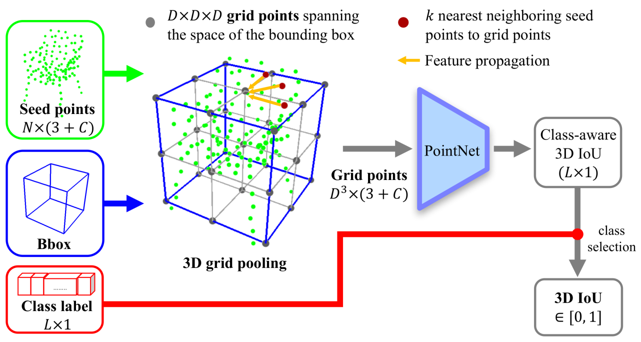
$D \times D \times D$ grid points spanning the space of the bounding box
k nearest neighboring seed points to grid points
Feature propagation
Seed points $N \times (3+C)$
Bbox
Class label $L \times 1$
3D grid pooling
Grid points $D^3 \times (3+C)$
PointNet
Class-aware 3D IoU $(L \times 1)$
class selection
3D IoU $\in [0,1]$

Figure 1. **3D IoU module** takes inputs seed feature points and a bounding box along with its predicted class label, and estimates the 3D IoU between the box and its maximum overlapping ground truth. The module constructs a 3D regular grid with D^3 virtual grid points spanning over the bounding box. We then perform a 3D grid feature pooling that applies a distance-weighted interpolation for feature propagation from the seed points to the grid points. Then the local coordinates of these grid points along with their features are pushed through a PointNet to regress class-aware 3D IoU. Finally, we use the input class label for output selection.

$w_i = \frac{1}{d(g_m, g_i)^2}$ and d is the L2 distance. The interpolated features of every grid point is then concatenated with the relative coordinates and forwarded into an MLP with channel dimensions of [256+3, 128, 128, 128] to learn a new feature. Then the features of all grid points go through a global max pooling, after which go through another MLP with channels [128, 128, 128, C], where C is the number of classes, to make the IoU prediction class-aware. Finally, we select the per box IoU estimation by using the class label (during training) or class prediction (during inference).

Training IoU Estimation Module To train the 3D IoU estimation branch in both stages, we generate on-the-fly training data via jittering the bounding box predictions, i.e. adding Gaussian noise to the box center and size. As a way of data augmentation, this jittering is essential for the generalization of IoU estimation to unlabeled data. We use an L1 loss to supervise the IoU estimation module.

2. More Implementation Details for VoteNet-based 3DIoUMatch

Training For the pre-training stage, we find that the network does not converge using the same protocol as fully-supervised VoteNet. We instead use a new protocol, where the network is trained for 900 epochs, optimized by an ADAM optimizer with an initial learning rate of 0.001, and the learning rate is decayed by 0.1, 0.1, 0.1 at the 400th, 600th and 800th, respectively. We observe convergence using this protocol on all ratios of labeled data.

Inspired by IoU-Net[2], for both stages, we generate on-the-fly training data via jittering the bounding box predictions for the IoU estimation module. Specifically, we add $\epsilon_{size} \sim N(0, (0.3\mathbf{d})^2)$ to each bounding box size prediction $\mathbf{d}$ and add $\epsilon_{center} \sim N(0, (0.3\mathbf{d})^2)$ to each bounding box center prediction $\mathbf{c}$ to obtain $N_{proposal}$ more training samples. The final IoU estimation loss is the L1 loss averaged over all IoU trainig samples, original predictions or jitters. The IoU estimation loss weight is 1.

Inference As IoU-Net[2] did not release code, we implemented a simple version of test-time IoU optimization.

1. We obtain the original bounding box proposals.
2. We calculate the gradients of the IoU estimation w.r.t. to bounding box size and center, $grad_{size}, grad_{center}$, and update the bounding box size and center by adding $grad_{size} * \lambda, grad_{center} * \lambda$ to the box size and center, respectively, where λ is the optimization step size.
3. We repeat the second step for T times.

We find setting T to 10 yields noticeable improvement while not slowing inference speed too much. Choosing λ from the range of $[1e^{-4}, 5e^{-4}]$ has similar performance.

3. More Implementation Details for PV-RCNN-based 3DIoUMatch

We basically follow the training protocols and settings of PV-RCNN [4]. For pre-training on small amounts of labeled data (1% and 2%), we train ten times the original number of epochs for the model to converge. For SSL training, we set the batch size to 16 (8 labeled + 8 unlabeled, 8 GPUs) and train five times the original number of epochs.

4. Overhead of the IoU module

Our light-weighted IoU estimation module brings moderate overhead to the network, as shown in Table 1. The memory reported in the second column refers to the memory consumed by training with batch size 8 on a single GTX 1080Ti GPU. The last two columns mean the time consumed by a full pass (forwarding and backwarding) of a batch of 8 on a single GTX 1080Ti GPU, training ScanNet and SUNRGB-D respectively. Note that regardless of the network design, there is overhead introduced by calculating the ground truth IoU for supervision.

Method	Mem. (GB)	ScanNet (s)	SUNRGB-D (s)
VoteNet	6.56	0.282	0.316
Ours	6.60	0.325	0.377

Table 1. Memory and time overhead of the IoU module.

	cab	bed	chair	sofa	table	door	wind	bkshf	pic	cntr	desk	curt	fridg	showr	toil	sink	bath	ofurn
VoteNet mAP@0.25	17.9	74.7	74.5	75.3	45.6	18.3	11.7	21.7	0.7	28.4	49.4	21.5	23.2	18.5	79.6	25.7	66.3	11.7
SESS mAP@0.25	20.5	75.1	76.2	76.4	48.1	20.0	14.4	19.4	1.2	30.0	51.8	25.0	30.0	26.4	82.2	29.2	72.3	14.1
Without IoU mAP@0.25	22.6	79.5	77.8	77.8	49.6	25.4	18.6	27.7	3.3	41.4	56.2	27.4	30.4	**53.6**	81.3	28.5	74.5	18.8
3DIoUMatch mAP@0.25	**26.6**	**82.6**	**80.9**	**83.3**	**52.1**	**28.0**	**19.9**	**29.4**	**3.7**	**45.0**	**61.9**	**29.2**	**34.1**	51.2	**85.7**	**32.3**	**82.8**	**21.5**
VoteNet mAP@0.5	3.2	64.6	43.4	49.3	25.1	2.8	1.1	8.7	0.0	2.4	14.7	3.9	7.6	1.1	46.8	11.9	39.4	1.5
SESS mAP@0.5	3.7	61.2	48.0	44.8	29.5	3.2	2.8	8.4	0.2	7.5	19.2	5.0	12.2	1.8	48.7	**15.3**	40.8	2.2
Without IoU mAP@0.5	3.9	66.1	52.7	50.7	35.1	7.9	5.0	13.1	**0.9**	14.5	26.1	**10.3**	17.5	**7.0**	63.9	11.7	62.1	4.9
3DIoUMatch mAP@0.5	**5.9**	**72.0**	**60.5**	**56.6**	**39.7**	**10.3**	**5.2**	**18.1**	0.7	**16.0**	**35.3**	8.3	**21.4**	6.2	**67.5**	13.2	**67.6**	**5.2**

Table 2. Per class mAP@0.25 and mAP@0.5 on ScanNet val set, with 10% labeled data.

	bathtub	bed	bookshelf	chair	desk	dresser	nightstand	sofa	table	toilet
VoteNet mAP@0.25	67.8	32.2	39.4	58.5	53.5	8.0	1.9	14.7	3.2	20.3
SESS mAP@0.25	70.8	34.7	41.9	60.4	63.0	9.8	3.7	25.2	4.0	28.0
Without IoU mAP@0.25	75.1	33.5	43.0	59.5	76.9	**6.8**	5.1	33.0	3.5	34.8
3DIoUMatch mAP@0.25	**75.4**	**37.7**	**45.2**	**64.2**	**77.0**	6.0	**5.7**	**34.6**	**4.5**	**39.4**
VoteNet mAP@0.5	31.2	6.2	15.5	29.6	14.6	0.5	0.2	2.0	0.3	5.2
SESS mAP@0.5	36.7	7.2	19.2	31.8	20.4	0.7	0.5	7.0	0.4	7.1
Without IoU mAP@0.5	41.5	9.7	25.7	34.5	40.8	**0.8**	0.8	8.3	**0.8**	11.4
3DIoUMatch mAP@0.5	**45.2**	**14.4**	**27.8**	**43.6**	**47.2**	**0.8**	**1.9**	**15.7**	0.6	**13.4**

Table 3. Per class mAP@0.25 and mAP@0.5 on SUNRGB-D val set, with 5% labeled data.

5. IoU Module Comparison

As mentioned in 1, some region-based 3D detectors, e.g. STD [6], crop the features inside a predicted bounding box and regress the offset. To capture their core characteristics under IoU optimization, we build a simple IoU estimation module which only queries points inside the predicted box and passes the queried feature points through a PointNet to predict the 3D IoU, namely box query. In principle, the differentiability of this module is the same as that in STD, which doesn't release their code and misses the IoU optimization step in their paper. For a fair comparison, we train another IoU-aware VoteNet with box query as the IoU estimation module and show the comparison between it and our proposed method on the full set of ScanNet and SUN RGB-D. From the results in Table 4, we prove that our method is more effective on both IoU-guided NMS and IoU optimization than box query.

We provide more explanation on why an IoU estimation network design like that in STD[6] is less effective in IoU estimation and is not differentiable. Given a bounding box proposal, STD concatenates the canonized coordinates and features of the points inside the bounding box to form new features of the points. Therefore, the new feature of a point f' can be denoted as a function of the point coordinates $\mathbf{p}$, the original point feature f, the bounding box center $\mathbf{c}$ and the bounding box heading angle θ. Then STD voxelizes the bounding box and sample points in each voxel to produce the voxel feature f_v. The process of producing the voxel feature from points in voxels consist of no other parameters except from the point features $\{f'_i\}$ and point coordinates $\mathbf{p}_i$, so f_v is still a function of $\{f_i\}, \{\mathbf{p}_i\}, \mathbf{c}, \theta$. As all voxel features are flattened and fed to an MLP, which outputs the final IoU, we can conclude that the IoU estimation is not differentiable w.r.t. bounding box size.

We also argue that for VoteNet, since the number of seed points with features are small (1024), box query methods may have difficulty querying points inside a bounding box, especially if a bounding box is too small. Our method, instead won't suffer from this as we are not confined to points inside the bounding box.

Although STD didn't release code, we still implemented an IoU estimation module according to the paper for better comparison. However, some issues need to be stated. First, since the backbone of STD is very different from VoteNet, the comparison between IoU estimation module alone is inherently problematic. Second, STD aims at outdoor object detection, where the task is slightly different. Third, we adopted most of the parameters of STD in the paper, but changed number of voxels (to 27) and number of points sampled per voxel (to 6) due to memory concerns and the small number of seed points in VoteNet. The results in Table 4 show the better performance of our IoU module. We also observe serious overfitting using the STD IoU module, suggesting that it may not be suitable for our problem.

6. Why not Supervise Votes and Objectness in VoteNet?

As we mentioned, we supervise all VoteNet loss terms on unlabeled data except for vote regression loss and objectness binary classification loss. As we observe, supervising votes or objectness with pseudo labels leads to degrading performance. The main reason is that by rigorous filtering

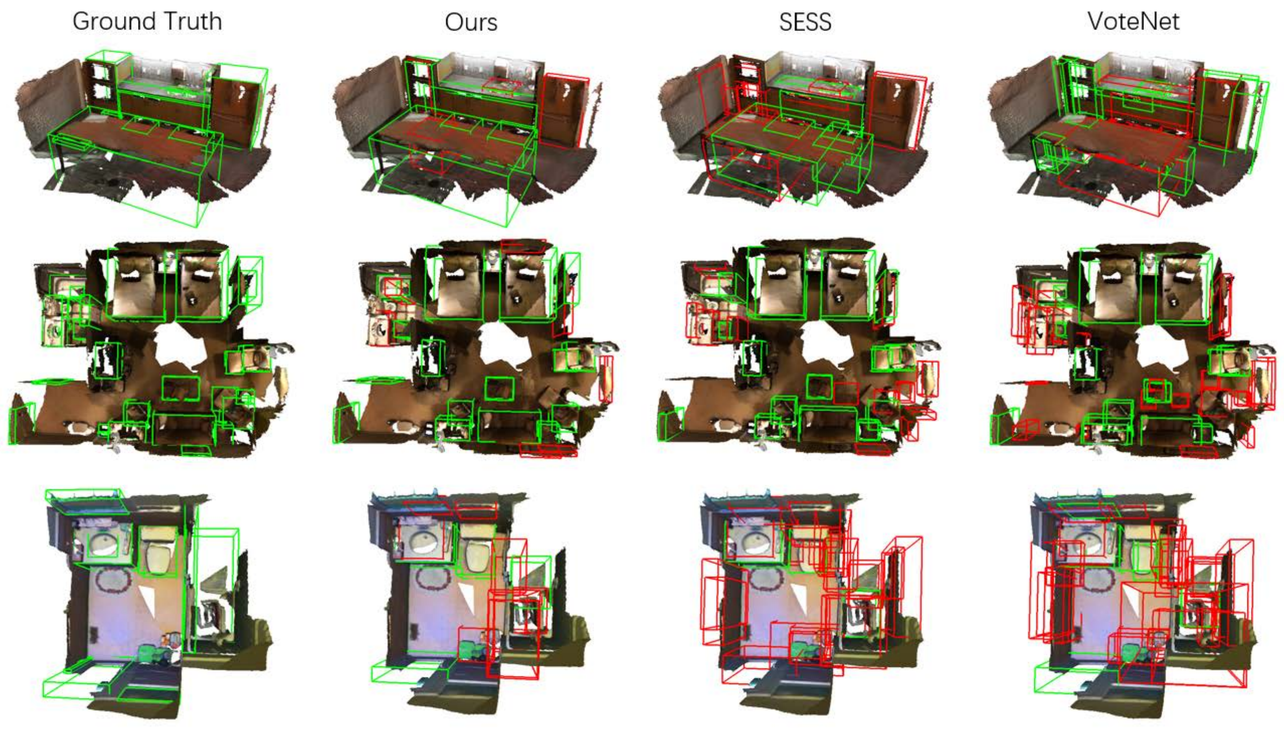

Figure 2. **Qualitative results on ScanNet, with 10% labeled data.** Here green bounding boxes have an IoU ≥ 0.25 while red bounding boxes are with an IoU < 0.25.

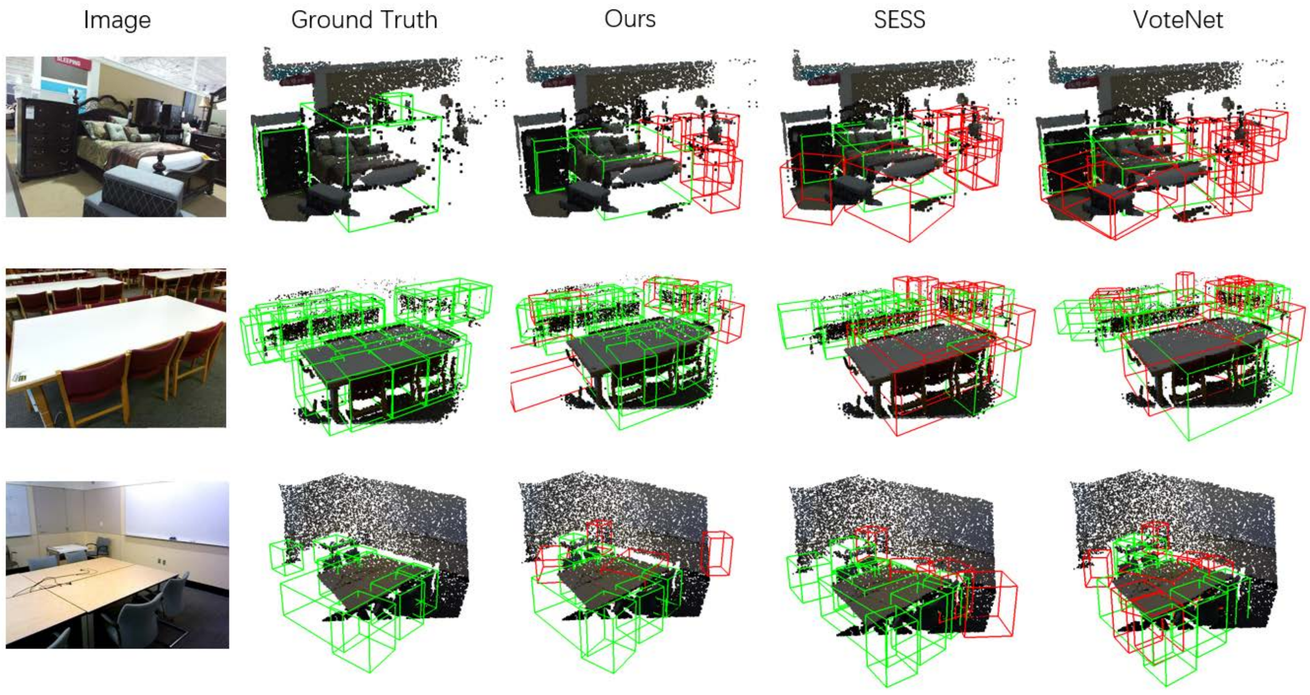

Figure 3. Qualitative results on SUNRGB-D, with 5% labeled data.

Method	ScanNet		SUN RGB-D	
	mAP @0.25	mAP @0.5	mAP @0.25	mAP @0.5
Obj-NMS [3]	57.84	35.99	58.01	33.44
Box query IoU-NMS only	57.56	37.07	58.16	34.81
Box query IoU-NMS + Optim.	57.62	37.17	58.19	34.90
STD IoU-NMS only	57.81	36.21	58.21	34.76
STD IoU-NMS + Optim.	57.85	36.21	58.21	34.75
Ours IoU-NMS only	57.92	37.01	58.82	36.22
Ours IoU-NMS + Optim.	**58.46**	**37.43**	**59.11**	**36.71**

Table 4. Comparison of our IoU module with box-query on ScanNet 100% and SUN RGB-D 100% .

and deduplication we can only be highly confident of a true object being close to a pseudo bounding box, but we are not sure whether or not there is a true object where there are no pseudo bounding boxes nearby. If we supervise objectness on unlabeled data with the pseudo labels the same way as VoteNet, it's not difficult to imagine the network would be more and more biased on detecting objects. In Table 5, our experiments on ScanNet 10% and SUNRGB-D 5% show that the performance suffers a drop after supervising objectness on unlabeled data.

Vote prediction is an unique component of VoteNet. For a point, the label for its vote is the center of the object it belongs to. To generate pseudo vote labels, the straightforward way is to count every point inside a pseudo bounding box as a vote. However, since this pseudo vote label set is also far from complete, we face a similar problem supervising with it. In Table 5, our experiments on ScanNet 10% and SUNRGB-D 5% also show that the performance drops after supervising vote prediction on unlabeled data.

Method	ScanNet 10%		SUNRGB-D 5%	
	mAP @0.25	mAP @0.5	mAP @0.25	mAP @0.5
3DIoUMatch	47.2	28.3	39.0	21.1
+vote sup. on unlabeled	45.4	28.3	37.9	20.9
+obj. sup. on unlabeled	40.1	26.0	38.2	20.4

Table 5. Objectness & vote supervision on unlabeled data using pseudo-labels.

7. Per-class Evaluation

We report per-class average precision on ScanNet with 10% labeled data and SUNRGB-D with 5% labeled data, respectively. The bold numbers are the highest per class. The results in Table 2, 3 show that our method improves the average precision on nearly all classes over SESS. Our 3DIoUMatch also has better performance on most classes than the without-IoU version.

8. Qualitative Results

We show the qualitative results on ScanNet val set with 10% labeled training data, Figure 2 and on SUNRGB-D val set with 5% labeled training data, Figure 3. For the results of our method, SESS and VoteNet, green bounding boxes are the predicted bounding boxes whose IoU ≥ 0.25, and the red bounding boxes are those with an IoU < 0.25. As can be seen in both figures, our method give more accurate predictions and significantly reduces the number of false positives.

References

[1] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[2] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–799, 2018.

[3] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9277–9286, 2019.

[4] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020.

[5] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019.

[6] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std: Sparse-to-dense 3d object detector for point cloud. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1951–1960, 2019.

[7] Li Yi, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas J Guibas. Gspn: Generative shape proposal network for 3d instance segmentation in point cloud. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3947–3956, 2019.