3D Object Detection in Self-driving Cars

Yiheng Xiong
Technical University of Munich
Boltzmannstr. 3, 85748 Garching
yiheng.xiong@tum.de

Abstract

VoteNet, an end-to-end 3D object detection network based on a synergy of deep point set networks and Hough voting, is a novel approach to 3D point cloud object detection problems. It achieves state-of-the-art results for multiple 3D object detection tasks in indoor scenes. However, there are few works using VoteNet for tasks in outdoor scenes. Thus, in this work, we test VoteNet in outdoor self-driving scenarios. And it turns out that VoteNet can achieve modestly good results in outdoor scenes with some modifications.

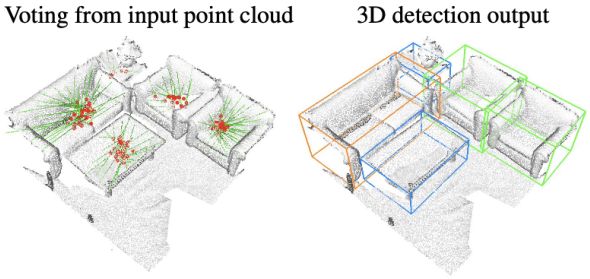


Figure 1. **3D object detection in point clouds with a deep Hough voting model**. Taken from [15]

1. Introduction

As a huge progress of deep learning has been made in recent years, multiple computer vision tasks such as semantic segmentation, instance segmentation as well as object detection have been improved significantly with the help of deep neural networks. More specifically, the goal of 3D object detection is to localize and recognize objects in 3D scene. In self-driving scenarios, detecting objects on roads is the foundation for autonomous cars before they make any instructions such as turning left or turning right. In particular, majority of autonomous driving systems heavily rely on LiDAR sensors, which produce 3D point clouds on object surfaces.

On the one hand, compared to images, 3D point clouds produced by LiDAR sensors provide accurate geometry and robustness to illumination changes. On the other hand, however, point clouds are irregular, unordered and sparse, which makes direct application of typical convolution-based methods difficult. Therefore, most proposed methods either voxelize the irregular point clouds to regular 3D grids and apply 3D CNN detectors [11] [20] or project points to regular 2D bird's eye view images and then apply 2D detectors to localize objects [6] [24]. These methods either fail to leverage sparsity in the data and suffer from high computation cost due to expensive 3D convolutions or sacrifice geometric details. More recently, [12] [16]proposed a cascaded two-step pipeline by firstly detecting objects in front-view images and then localizing objects in frustum point clouds extruded from the 2D boxes, which however is strictly dependent on the 2D detector and will miss an object entirely if it is not detected in 2D.

In this project, we analyze VoteNet - a novel end-to-end deep neural network that leverages the Hough voting to detect 3D objects directly from the raw point cloud data (shown in Fig. 1). The model achieved state-of-the-art results in 3D object detection tasks on two large indoor datasets - ScanNet [7] and SUN RGB-D [19], relying solely on point cloud data. But to our best knowledge, VoteNet has seldom been tested on outdoor point cloud datasets. One piece of work [1] found that VoteNet performs less well in outdoor scenes without modifications. To this end, we first test two point-cloud-based backbones PointNet [17] and PointNet++ [18], an essential part of the whole VoteNet, on SemanticKITTI [2] dataset. Then we adapt the tuned backbone to VoteNet and test it on outdoor LiDAR point cloud data of KITTI [8] 3D object detection dataset. Our study shows that VoteNet with some modifications can also perform relatively well in outdoor scenes.

In summary, the contributions of our work are two parts: first test and tune VoteNet on KITTI 3D object detection benchmark; then show the feasibility as well as the limitation of using VoteNet in outdoor scenes.

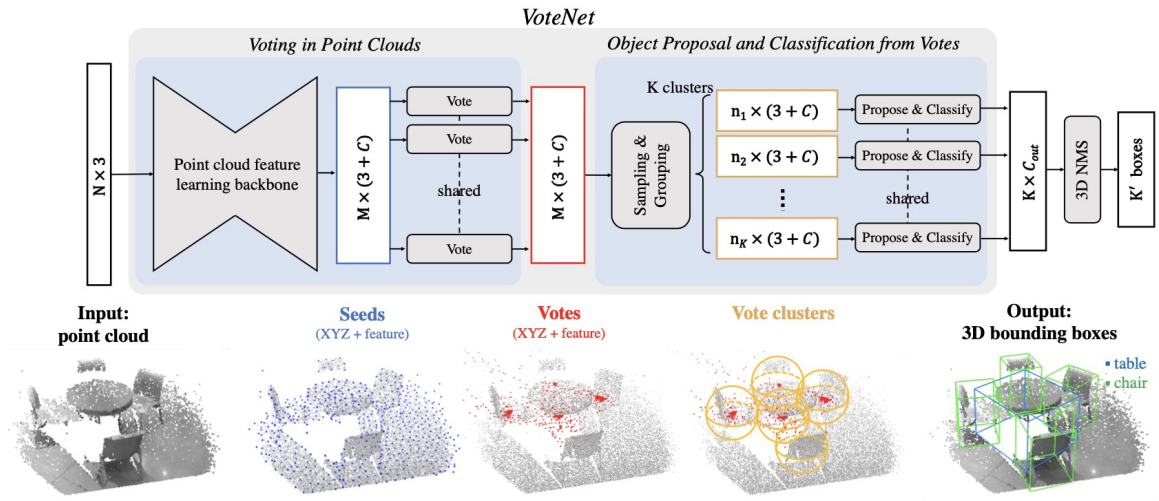


Figure 2. **VoteNet Architecture**. Taken from [15]

2. VoteNet Architecture

Fig. 2 illustrates the overall architecture of VoteNet. The entire work comprises of two parts: voting part and object proposal and classification part. In the bottom of Fig. 2, data transformation process is clearly shown: from point cloud to seeds via backbone - from seeds to votes by voting mechanism - from votes to vote clusters - finally it outputs multiple proposals.

2.1. Vote Generation Network

The initial input point cloud has the shape of $N \times 3$, with a 3D coordinate for each of the N points. These points are then processed through the VGN to generate M votes and each vote has a 3D coordinate and a high dimensional feature vector. Point cloud feature learning via a backbone network and deep Hough voting from seed points are the major parts of the VGN.

Point cloud feature learning. The initial points with the shape of $N \times 3$ will be put into the backbone network. Although there is no restriction on the choices of backbone, in the paper of VoteNet, PointNet++ [18] is adopted due to its simplicity and demonstrated success on multiple 3D computer vision problems which is essentially composed of two parts: set abstraction(SA) and feature propagation(FP). In general, PointNet++ contains several SA layers and FP layers.

In the first two sub-layers of SA layer, the initial N points are sampled via farthest point sampling (FPS) to M centroids, based on which it outputs M groups of points where each group contains K nearest neighbor points with respect to the corresponding centroid. Thus, the output has the shape of $M \times K \times 3$. The third sub-layer of SA is PointNet [17] layer, through which the coordinates of each group of points will be first normalized with respect to its centroid. And each group of normalized points will be put into several shared MLP layers and one max pooling layer to produce a C-dimensional feature vector in each group as is shown in Eq. 1, where h function is a shared MLP layer and g function is a max pooling layer.

$$f(\{x_1, ..., x_n\}) \approx g(h(x_1), ..., h(x_n)) \tag{1}$$

FP layer aims to propagate features of current subsampled points to original unsampled points. It adopts a hierarchical propagation strategy with distance based interpolation and across level skip links. Given point features from $N_l \times (d + C)$ points, FP layer will propagate them to N_{l-1} original unsampled points based on inverse distance average of k nearest neighbors of each original unsampled point, where N_{l-1} and N_l (with $N_l \leq N_{l-1}$) are point set size of input and output of SA level l [18]. Then the concatenated features are passed through a "unit pointnet" where a few fully connected and ReLU layers are applied to update each point's feature vector [18]. In summary, the output of the PointNet++ backbone is called seeds with the shape of $M \times (3 + C)$.

Hough voting with deep networks. Given seeds with the shape of $M \times (3+C)$, the input to the Hough voting module only contains the C-dimensional feature vector and it will generate an intermediate output $[\Delta x_i; \Delta y_i]$ which is composed of euclidean space offset $\Delta x_i \in \mathbb{R}^3$ and feature offset $\Delta y_i \in \mathbb{R}^C$. And specifically, predicted euclidean space off-

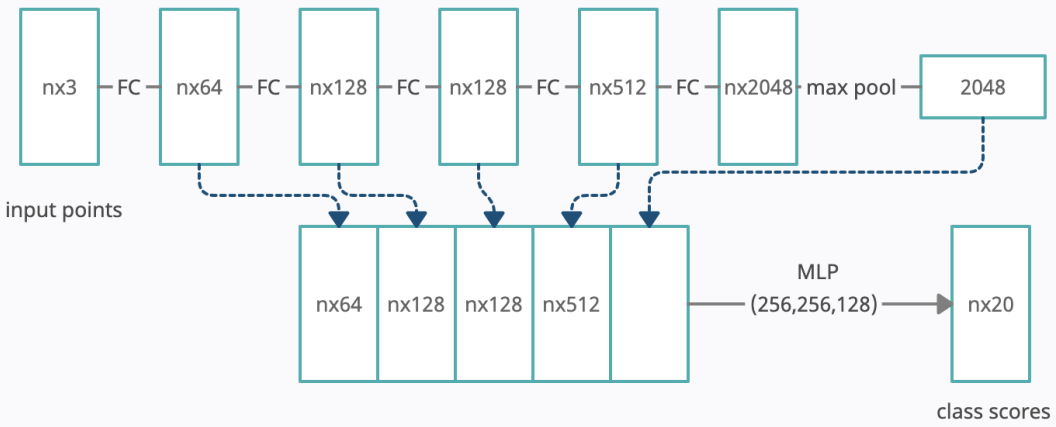


Figure 3. **PointNet Architecture**. FC is fully connected layer operating on each point. MLP is multi-layer perceptron on each point (second last MLP is followed by dropout layer with drop ratio 0.5).

set is supervised by a regression loss:

$$L_{vote-reg} = \frac{1}{M_{pos}} \sum \|\Delta x_i - \Delta x_i^*\| \, I[s_i \; on \; object] \tag{2}$$

where $I[s_i \; on \; object]$ indicates whether a seed point s_i is on an object surface and M_{pos} is the count of total number of seeds on object surface. Δx_i^* is the ground truth displacement from the seed position x_i to the bounding box center of the object it belongs to.

For the final output, Hough voting module generates multiple votes with the shape of $[x_i + \Delta x_i; f_i + \Delta f_i]$. In essence, votes are the same as seeds in tensor representation (i.e. the dimension is the same) but unlike seeds, votes are no longer grounded on object surfaces. More specifically, votes generated from seeds on the same object are now closer to each other than the seeds are, which makes it easier to combine cues from different parts of the object. In summary, the output here is called votes with the same shape as seeds.

2.2. Object Proposal and Classification from Votes

The votes have more semantic-aware locality which can be aggregated better for generating object proposals and classifying them [15].

Vote clustering through sampling and grouping. Given $M \times (3 + C)$ votes, FPS is used again to sample K votes, based on which it will form K clusters by finding neighboring votes.

Proposal and classification from vote clusters. The input here is normalized votes, where each coordinate of vote is normalized with repect to the centroid coordinate of its cluster. Then there is a shared PointNet layer - votes from each cluster are independently processed by a MLP_1 then they are processed through a max pooling layer, which generates a single feature vector every cluster.

Feature vectors above are put into a MLP_2 and generate a multidimensional vector p for each cluster. It contains objectness score, bounding box parameters and semantic classification scores:

$$p = MLP_2 \left\{ max \left\{ MLP_1([z_i; h_i]) \right\} \right\} \tag{3}$$

where z_i with the shape of 3 is normalized coordinate as vote location and h_i with the shape of C is the vote feature.

Loss Function. The loss functions in the proposal and classification stage consist of objectness, bounding box estimation, and semantic classification losses:

1. Objectness: cross entropy loss normalized by the number of non-ignored proposals (either positive proposals - close to a ground truth object center, or negative proposals - far from any centers).

2. Bounding box estimation and class prediction (only for positive proposals): decouple the box loss to center regression, heading angle estimation and box size estimation following [16]:

$$\begin{aligned} L_{box} = \lambda_1 L_{center} + \lambda_2 L_{size-cls} + \lambda_3 L_{size-reg} \\ + \lambda_4 L_{heading-cls} + \lambda_5 L_{heading-reg} \end{aligned} \tag{4}$$

3. Semantic classification: standard cross entropy loss.

4. All regression loss: Huber loss (i.e. smooth-L1).

3. Experiments

In this section, we first test two point-cloud-based backbones, PointNet [17] and PointNet++ [18], and compare the results with state-of-the-art methods on SemanticKITTI dataset [2]. After that, we adapt the tuned backbone to the whole VoteNet then test it on KITTI 3D object detection dataset [8], and again compare the results with one previous

layer name	input layer	type	output size	layer params
sa1	raw point cloud	SA	(1024, 3+64)	(1024,2,[32,32,64])
sa2	sa1	SA	(512, 3+128)	(512,4,[64,64,128])
sa3	sa2	SA	(256, 3+256)	(256,6,[128,128,256])
sa4	sa3	SA	(128, 3+512)	(128,8,[256,256,512])
fp1	sa3, sa4	FP	(256, 3+256)	[256,256]
fp2	sa2, sa3	FP	(512, 3+256)	[256,256]
fp3	sa1, sa2	FP	(1024, 3+128)	[256,128]
fp4	raw point cloud, sa1	FP	(N, K)	[128,128,128,128,K]

Table 1. **PointNet++ architecture**: layer specifications (second last fully connected layer in fp4 is followed by dropout layer with drop ratio 0.5).

work which also tested VoteNet on KITTI and other state-of-the-art methods on the same dataset.

3.1. Testing Backbones in Outdoor Scenes

3.1.1 Dataset

We test our backbones in semantic segmentation tasks on SemanticKITTI dataset, the largest dataset for autonomous vehicle LiDAR segmentation with point-wise 19 annotated classes. This dataset is based on the KITTI dataset, containing 41000 total frames which captured in 21 sequences, in which sequence 00 to 10 are training set with sequence 08 for validation.

3.1.2 Implement Details

Input and data augmentation. Input to our backbones is a point cloud of around $100k$ points. To augment the training data, we randomly sub-sample $50k$ points from the total scene points on-the-fly. We also randomly flip the point cloud in both horizontal direction, randomly rotate the scene points by Uniform$[-5°, 5°]$ around the upright-axis, and randomly scale the points by Uniform$[0.9, 1.1]$.

Network architecture details. For PointNet, we use the architecture shown in Fig. 3, which is similar to the one for part segmentation. The reason for using such an architecture is that it concatenates multiple intermediate feature vectors as well as the global feature vector, which performs better in the more complex outdoor scenes.

For PointNet++, we use four SA layers and four FP layers. Since outdoor scenes are much larger than indoor scenes, we adjust receptive radius of SA layers from 0.2, 0.4, 0.8 and 1.2 to 2, 4, 6 and 8 in meters while they sub-sample the input to 1024, 512, 256, 128 points repectively. The four FP layers up-sample the 4th SA layer's output back to original points with 20-dim class scores including the non-object class. (more details in the Table. 1).

Training the network. For PointNet, we train the entrie network with an Adam optimizer, batch size 2 (due to limited memory resource), an initial learning rate of 0.001 and weight decay 0.001. Because of the small batch size, we don't use any batch normalization layers while using Kaiming initialization [10] to get stable gradients. The learning rate is decreased by half every epoch. Due to the large number of classes and imbalance in the dataset, we add weight to each class in the crossentropy loss, which indicates the double square root of inverse proportion of each class. Training the model to convergence with PyTorch and on GTX 1080 GPU takes around 24 hours.

For PointNet++, we set batch size to 3, initial learning rate to 0.01 and others remain the same as PointNet settings. Training the model to convergence with PyTorch and on GTX 1080 GPU takes around 36 hours.

3.1.3 Results and Comparisons

To access the labeling performance, we rely on the commonly applied mean intersection-over-union(mIoU) for each class in the validtaion set. We compare our results with published results in the paper of SemanticKITTI shown in Table. 2. The paper results are evaluated on sequences 11 to 21 (test set). It is clear that we produce better results on our tuned PointNet++ than both PointNet(ours and paper version) and PointNet++(paper version). Based on that, we decide to use the tuned PoinNet++ as our backbone of VoteNet. On the other hand, however, compared with other state-of-the-art methods, the results are still very modest.

3.2. Adapting VoteNet to Outdoor Scenes

After testing above two backbones and achieving relatively good results with the tuned PointNet++, we adopt it to VoteNet and test the whole network altogether in outdoor scenes..

3.2.1 Dataset

We test our model on the KITTI 3D object detection dataset [8] which contains 7481 training images/point clouds and 7518 test images/point clouds. We only use the point cloud in our approach. We utilize the frequently used train/validation split [9] to divide the annotated part of the dataset into a training split of 3712 scenes and a validation

Approach	mIoU	road	sidewalk	parking	other-ground	building	car	truck	bicycle	motorcycle	other-vehicle	vegetation	trunk	terrain	person	bicyclist	motorcyclist	fence	pole	traffic sign
PointNet [17]	14.6	61.6	35.7	15.8	1.4	41.4	46.3	0.1	1.3	0.3	0.8	31.0	4.6	17.6	0.2	0.2	0.0	12.9	2.4	3.7
SPGraph [13]	17.4	45.0	28.5	0.6	0.6	64.3	49.3	0.1	0.2	0.2	0.8	48.9	27.2	24.6	0.3	2.7	0.1	20.8	15.9	0.8
SPLATNet [21]	18.4	64.6	39.1	0.4	0.0	58.3	58.2	0.0	0.0	0.0	0.0	71.1	9.9	19.3	0.0	0.0	0.0	23.1	5.6	0.0
PointNet++ [18]	20.1	72.0	41.8	18.7	5.6	62.3	53.7	0.9	1.9	0.2	0.2	46.5	13.8	30.0	0.9	1.0	0.0	16.9	6.0	8.9
SqueezeSeg [22]	29.5	85.4	54.3	26.9	4.5	57.4	68.8	3.3	16.0	4.1	3.6	60.0	24.3	53.7	12.9	13.1	0.9	29.0	17.5	24.5
SqueezeSegV2 [23]	39.7	88.6	67.6	45.8	17.7	73.7	81.8	13.4	18.5	17.9	14.0	71.8	35.8	60.2	20.1	25.1	3.9	41.1	20.2	36.3
DarkNet53Seg [3]	49.9	91.8	74.6	64.8	27.9	84.1	86.4	25.5	24.5	32.7	22.6	78.3	50.1	64.0	36.2	33.6	4.7	55.0	38.9	52.2
PointNet(Ours)	10.3	59.7	30.9	0.0	0.3	28.6	8.8	0.0	0.0	0.0	**2.0**	**40.9**	1.4	13.4	0.0	0.0	0.0	7.1	1.0	1.9
PointNet++(Ours)	**26.1**	70.3	**44.4**	8.5	0.0	**71.7**	**64.5**	0.3	0.0	0.0	**15.1**	**70.8**	**26.6**	**51.4**	0.1	**22.1**	0.0	**18.1**	**23.7**	8.9

Table 2. **Semantic segmantation results of backbones.** Our methods are evaluated on sequence 08 (validation set) while other methods are evaluated on sequences 11 to 21 (test set).

split of 3769 scenes. In addition, we perform our training on three most abundant classes of the KITTI dataset: *car*, *pedestrian* and *cyclist*, with mean sizes [width, height, length] of [1.63, 1.52, 3.88], [0.66, 1.76, 0.84] and [0.60, 1.74, 1.76] in meters and heading angle ranging from [-π, π].

3.2.2 Implement Details

Input and data augmentation. As the 3D bounding box ground truth annotations are only available only in the view field region of the front facing camera, we consider point clouds within the range of $[0, 70.4] \times [-40, 40] \times [-3, 1]$ meters along X, Y, Z axis respectively [24]. Thus, the number of points of each scene is reduced from around $100k$ to around $50k$. To augment the training data, we randomly sub-sample $18.75k$ points each scene on-the-fly. We also randomly flip the point cloud in both horizontal direction, randomly rotate the scene points by Uniform$[-5^\circ, 5^\circ]$ around the upright-axis, and randomly scale the points by Uniform$[0.9, 1.1]$.

Network architecture details. For point cloud feature learning backbone, we adopt our tuned PointNet++ except the last FP layer (fp4 shown in Table. 1), which has four SA layers and three FP layers. The three FP layers up-sample the 4th SA layer's output back to 1024 points with 256-dim features and 3D coordinates.

The voting layer is implemented by a multi-layer perceptron with FC output sizes of 256, 256, 259, where the last FC layer outputs XYZ offset and feature residuals.

The proposal module is realized through a SA layer with a post processing MLP_2 to generate proposals after the max-pooling. The SA uses radius 1 instead of original 0.3 in meters and MLP_1 with output sizes of 128, 128, 128. The max pooled features are further processes by MLP_2 with output sizes of 128, 128, 2 + 3 + $2NH$ + $4NS$ + NC consisting of 2 objectness scores, 3 center regression values, $2NH$ numbers for heading regression and $4NS$ numbers for box size regression and NC numbers for semantic classfication. In particular, the number of size templates (NS) is 3 while the number of heading bins (NH) is 12 (divide $[-\pi, \pi]$ into 12 parts equally).

Training the network. We train the entire network including the backbone end-to-end with an Adam optimizer, batch size 8 and an initial learning rate of 0.001. We also use batch normalization for all FC layers except the last one of voting module and proposal module. The learning rate is decreased by half after every 8 epochs. Besides, due to larger scenes and objects, we set near threshold to 1 meter and far threshold to 2 meters for objectness.

Inference. The output proposals of our model are post-processed by a 3D NMS module and we use mean average pricision (mAP) as evaluating metrics.

3.2.3 Results and Comparisons

Firstly, we compare our results with one previous work which also adapts VoteNet on KITTI 3D object dataset as shown in Table. 4. The results are produced by the 3D NMS module with an IoU threshold of 0.25. Our model performs better than the original and tuned version of the previous work.

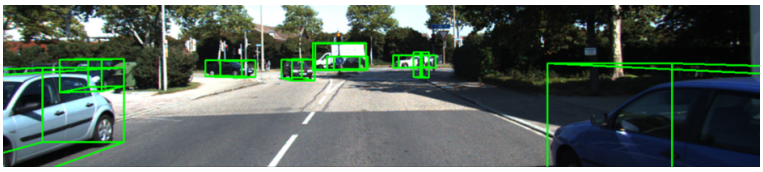
Then, to get a better evaluation of our model, we compare our results with state-of-the-art methods as shown in Table. 3. The results are produced by the 3D NMS module with an IoU threshold of 0.7 for *car* and 0.5 for *pedestrian* and *cyclist*. And for each class, detection outcomes are evaluated based on three difficulty levels: $easy$, $moderate$ and $hard$. Compared to the current state-of-the-art results of KITTI 3D object detection benchmark, AP scores are still relatively modest.

4. Limitation

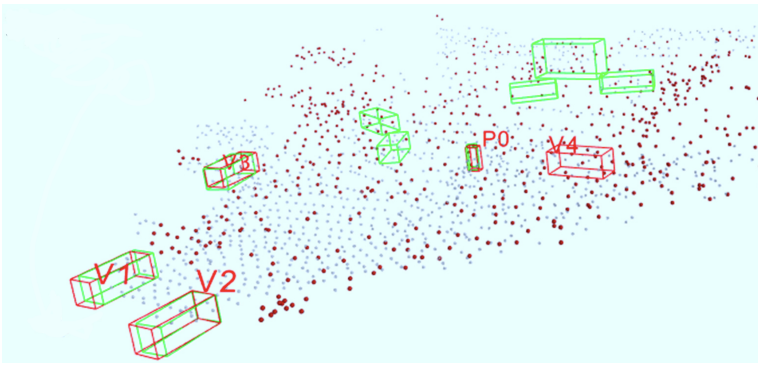
Point clouds produced by LiDAR have strongly varying density and are generally more sparse than the RGB-D imagery point clouds. In particular, if an object is far from the LiDAR, which can be a common case in outdoor scenes, the number of foreground points is quite small. Also, LiDAR returns from pedestrians and cyclists are even more

Method	Modality	Car			Pedestrian			Cyclist		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
Mono3D [4]	Mono	2.53	2.31	2.31	N/A	N/A	N/A	N/A	N/A	N/A
3DOP [5]	Stereo	6.55	5.07	4.10	N/A	N/A	N/A	N/A	N/A	N/A
VeloFCN [14]	LiDAR	15.20	13.66	15.98	N/A	N/A	N/A	N/A	N/A	N/A
MV(BV+FV) [6]	LiDAR	71.19	56.60	55.30	N/A	N/A	N/A	N/A	N/A	N/A
MV(BV+FV+RGB) [6]	LiDAR+Mono	71.29	62.68	56.56	N/A	N/A	N/A	N/A	N/A	N/A
Ours	LiDAR	43.67	31.42	19.86	11.83	10.79	6.84	24.85	19.34	10.34

Table 3. **Performance comparison in 3D detection: average pecision on KITTI validation split.**



(a) Ground truth bounding boxes projected onto image plane.



(b) 3D bounding boxes: V - car, P - pedestrian, C - cyclist. Green boxes are ground truth labels and red ones are predicting boxes. Blue points are seed points and red ones are votes.

	mAP	Car	Pedestrian	Cyclist
Original [1]	10.9	21.0	11.3	0.5
Tuned [1]	24.4	31.2	27.9	14.1
Ours	**33.3**	**52.6**	14.6	**32.7**

Table 4. **Final AP scores on KITTI validation split, IoU at 0.25.** Specifically, "Orginal" version is the one used in indoor scenes without any modifications and "Tuned" version is the one with modifications by the author.

sparse [24]. The small number of points results in very few vote points (seed points within a bounding box). Statistically, the average ratio of vote points is only around 1.5%.

Besides, vote points do not generally tend to cluster up near centroids of the object as expected shown in Fig. 4(b), possibly due to more sparsity in outdoor point clouds, which makes ratio of positive points (within objectness threshold) to negative points (beyond objectness threshold) quite small - around 5%. Thus, it increases the difficulty for object proposal and classification network to predict accurately.

5. Conclusion

To conclude, we test and tune VoteNet in 3D outdoor scenes and get relatively good results with some modifications to the original VoteNet. However, we also notice that compared with state-of-the-art methods, the results are still very limited. To this end, two major problems are about to be addressed: voting does not make enough sense in much more sparse outdoor scenes due to small number of vote points; the large amount of negative points causes difficulty for object proposal and classification network. One idea could be implementing "attention" mechanism which only focuses on the positive/meaningful areas. In addition, one could also possibly filter the negative areas well in advance before putting it into the neural network..

References

[1] Alexander Arzhanov. 3d object detection from point cloud.

[2] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Juergen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences, 2019.

[3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020.

[4] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2147–2156, 2016.

[5] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals for accurate object class detection. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[6] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. *CoRR*, abs/1611.07759, 2016.

[7] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas A. Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. *CoRR*, abs/1702.04405, 2017.

[8] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2018.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015.

[11] Ji Hou, Angela Dai, and Matthias Nießner. 3d-sis: 3d semantic instance segmentation of RGB-D scans. *CoRR*, abs/1812.07003, 2018.

[12] Jean Lahoud and Bernard Ghanem. 2d-driven 3d object detection in rgb-d images. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4632–4640, 2017.

[13] Loïc Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. *CoRR*, abs/1711.09869, 2017.

[14] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle detection from 3d lidar using fully convolutional network, 2016.

[15] Charles R. Qi, Or Litany, Kaiming He, and Leonidas J. Guibas. Deep hough voting for 3d object detection in point clouds. *CoRR*, abs/1904.09664, 2019.

[16] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from rgb-d data, 2018.

[17] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016.

[18] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *CoRR*, abs/1706.02413, 2017.

[19] Song Shuran, Lichtenberg Samuel P., and Xiao Jianxiong. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 567–576, 2015.

[20] Shuran Song and Jianxiong Xiao. Deep sliding shapes for amodal 3d object detection in RGB-D images. *CoRR*, abs/1511.02300, 2015.

[21] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. *CoRR*, abs/1802.08275, 2018.

[22] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud, 2017.

[23] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud, 2018.

[24] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *CoRR*, abs/1711.06396, 2017.