3D Object Detection in Point Clouds

Yen-Yu Chang Tzu-Sheng Kuo Shao-Yuan Ho
Department of Electrical Engineering
Stanford University
{yenyu, tskuo, yuan1995}@stanford.edu

Abstract

In this project, we study VoteNet, the state-of-the-art neural network that detects 3D objects in point clouds. We first survey related literature to understand what makes VoteNet better than other existing models. We then conduct an error analysis to identify where VoteNet could be improved. We finally propose three testing probes on VoteNet as an attempt for improvement. We report our evaluation results and suggest future research directions for 3D object detection in point clouds.

1 Introduction

Object detection is a task that aims to localize and classify objects in a given scene. Over the past decades, deep learning algorithms such as R-CNN [3] and YOLO [12] have achieved unprecedented success in 2D object detection. Recently, the rising interest in augmented reality and autonomous driving has further fueled the exploration of 3D object detection. Various deep learning algorithms are proposed to solve the task of object detection in 3D scenes [11] [10] [9] [4].

Object detection in 3D scenes is different from its 2D counterpart because of the difference in its data properties. 3D data representation, such as point clouds, is inherently irregular and sparse, making typical 2D deep learning networks an unsuitable and inefficient solution. As an alternative, researchers have either applied extended 2D CNNs to voxelized point clouds [14] or projected point clouds onto 2D surfaces [1] [6]. These approaches, however, sacrifice the valuable geometric information (which provides additional clues for object detection) and sparsity (which offers an opportunity for computational acceleration) in 3D point clouds.

In this project, we aim to detect 3D object in point clouds directly. We build upon a recently proposed state-of-the-art 3D detection network, VoteNet [11], and hope to improve its performance. We first conduct error analysis on VoteNet based on what we have learned in the class. With the clues from the analysis, we further propose three testing probes trying to improve its performance. Finally, we report our findings and suggest future research directions on 3D object detection in point clouds.

2 Related Work

Researchers have been working on 3D object detection before the rise of deep learning. For example, a conditional random field has been applied to integrate information of 2D segmentation and 3D geometry for 3D cuboid classification with RGB-D data [8]. A template-based method detects 3D objects in RGB-D data by retrieving aligned and scaled 3D shapes from a database [7]. Support Vector Machine (SVM) is also adopted to compare objects in 3D sliding windows with CAD models rendered from hundreds of viewpoints for 3D object detection [13].

As deep neural networks achieve unprecedented success on 2D object detection, researchers extend 2D object detection networks for 3D detection tasks. For example, researchers apply 3D convolutional layers to voxelized RGB-D data for object detection [15]. Similarly, the R-CNN framework [3] is extended into a 3D version for detection on 3D voxels [5].

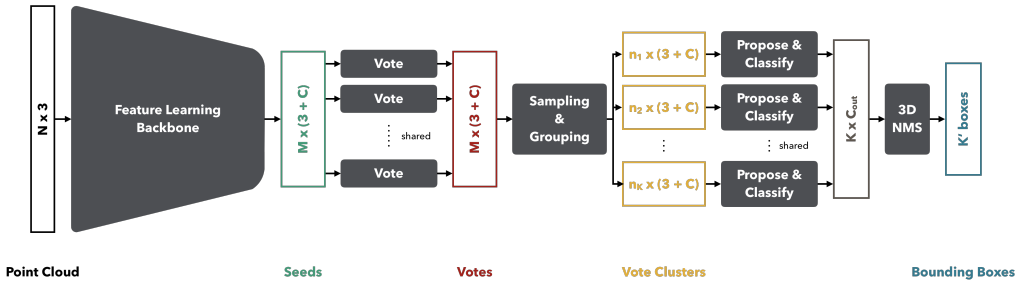


Figure 1: An illustration of the architecture of VoteNet. The input of VoteNet is raw point clouds and the output is bounding boxes of 3D objects. Modules within VoteNet are denoted using background color in grey. The remainings are the input and output of these modules.

On the contrary to these approaches, another set of deep learning research projects 3D data onto 2D planes and then apply 2D object detectors for detection. For example, the MV3D network fuses the information of a 2D RGB image, a 2D LIDAR bird's view data, and a 2D LIDAR front view data to generate 3D object proposals [1]. Another approach detects objects from 2D front-view images directly and generates 3D bounding based on the front-view depth image with a regression model [6].

All the above mentioned deep learning approaches either voxelize or project 3D data, reducing the rich information available in raw 3D point clouds. In this project, we study VoteNet [11], which is an end-to-end deep learning framework that directly takes 3D point clouds as input for 3D object detection.

3 VoteNet

VoteNet is a recently proposed end-to-end framework that achieves state-of-the-art performance on 3D object detection. An illustration of the architecture of VoteNet is shown in Figure 1.

The architecture of VoteNet is divided into five stages. First, a feature learning backbone takes N raw point clouds with XYZ coordinates as input and outputs M (< N) seeds with additional learned spatial features. Seeds are fed into a shared voting module that generates votes, which are the expected centers of objects. Grouped and sampled votes form clusters, which become the input of the next shared module that generates bounding boxes of 3D objects. Finally, an NMS module prunes and reduces the number of proposed bounding boxes as final results.

The network is trained end-to-end with the loss function L:

$$L = L_{vote-reg} + aL_{obj-cls} + bL_{box} + cL_{sem-cls},$$

where $L_{vote-reg}$ denotes the regression loss for the displacement from seeds to their corresponding object centers; $L_{obj-cls}$ denotes objectness scores that estimate votes are either close to or far from object centers; L_{box} denotes regression loss for the center, orientation, and size of bounding boxes; $L_{sem-cls}$ denotes cross entropy loss for semantic classification. The weights a, b, and c are 0.5, 1, and 0.1, respectively in the original implementation.

4 Dataset

The authors of VoteNet evaluated their network on two datasets – SUN RGB-D [16] and ScanNet [2]. In this project, we evaluate results only on the SUN RGB-D dataset because we do not have access to the ScanNet dataset.

SUN RGB-D is a benchmark dataset with about 10K RGB-D images for 3D scene understanding. It contains 3D bounding boxes, orientations, and categories of 37 objects in different room layouts. Following Votenet, we adopt the same pre-processing pipeline and evaluate our result on the same object categories.

Table 1: The original and reproduced quantitative results. The metric is average precision with 3D Intersection over Union threshold = 0.25 for each object category. The last column is mean average precision for all categories.

model	bathtub	bed	bookshelf	chair	desk	dresser	nightstand	sofa	table	toilet	mAP
paper	74.4	83.0	28.8	75.3	22.0	29.8	62.2	64.0	47.3	90.1	57.7
reproduction 1	77.8	82.7	28.1	74.2	24.2	24.4	61.7	62.2	49.0	89.4	57.4
reproduction 2	77.1	83.1	28.4	74.7	25.1	25.2	61.7	61.6	49.4	87.7	57.4
reproduction 3	76.8	83.7	27.9	74.3	24.3	26.4	60.9	63.4	48.6	88.6	57.5

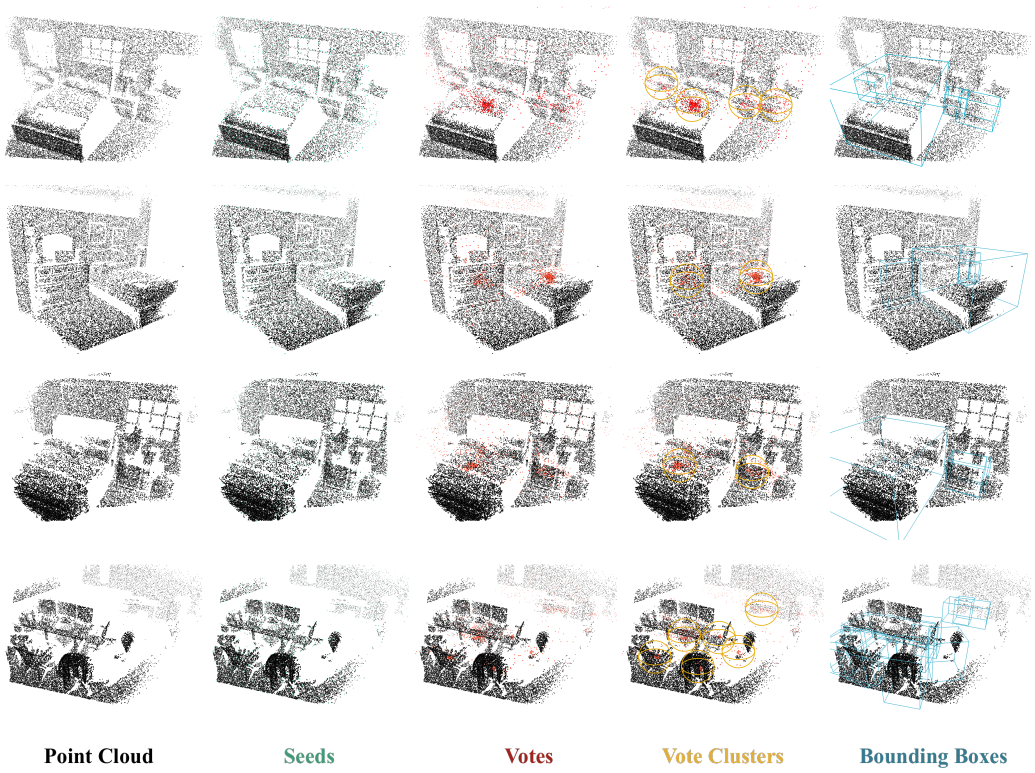


Figure 2: Reproduced qualitative results.

5 Reproduction Experiment

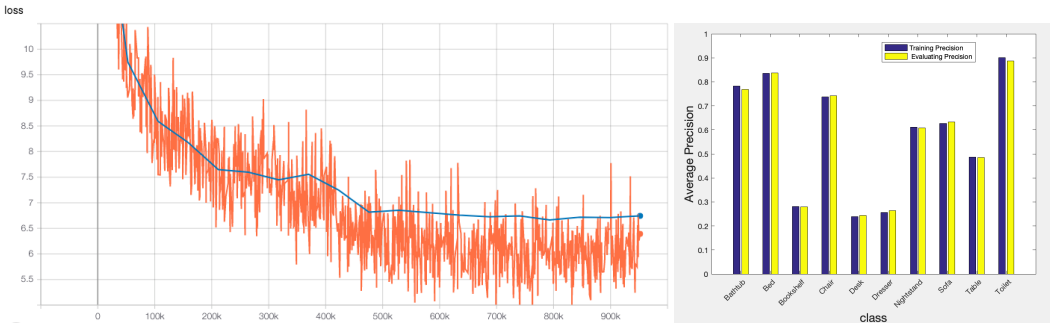
We reproduce VoteNet and run it three times for experiment. As shown in Table 1, the reproduced quantitative results are comparable to the original result. Example qualitative reproduction results are also shown in Figure 2, where colors are selected to match corresponding components in Figure 1.

6 Error Analysis

After reproducing comparable results, we are interested in understanding how we may further improve VoteNet. We conduct an error analysis based on what we have learned in the class, focusing mainly on tracking training loss and decoupling results for each object category.

6.1 Training Loss

According to the authors of VoteNet, learning rate scheduling is crucial for the performance. This finding is consistent with what we have learned in class during the section for hyperparameter tuning. To identify possible alternatives for the value of the learning rate, we visualize the training loss as shown in Figure 3a, where the vertical axis denotes training loss and the horizontal axis denotes



(a) **(b)**

Figure 3: Visualization of error analysis: (a) Variation of the loss value during the training process, and (b) visualization of average precision for different categories.

Table 2: Results for different improvement strategies.

model	bathtub	bed	bookshelf	chair	desk	dresser	nightstand	sofa	table	toilet	mAP
paper	74.4	83.0	28.8	75.3	22.0	29.8	62.2	64.0	47.3	90.1	57.7
decay at 100,130,160 epochs	76.4	82.5	30.2	74.5	23.9	27.2	59.2	62.9	48.3	88.8	57.4
decay at 80,110,140 epochs	77.3	83.8	28.5	74.5	23.3	26.1	63.1	62.3	49.5	90.6	57.9
increase a to 0.8	79.5	83.3	28.5	74.3	23.7	26.9	60.0	63.3	49.4	89.0	57.8
increase b to 1.3	74.6	82.0	26.3	74.7	23.3	26.5	58.4	60.9	49.8	88.3	56.5
increase c to 0.5	71.6	83.3	26.1	74.4	22.4	26.7	61.1	58.9	46.5	89.3	56.0
include RGB features	72.4	84.3	27.0	73.4	24.6	24.2	58.3	62.7	49.7	87.7	56.4
voxel feature encoding layer	4.75	23.2	2.95	36.9	2.00	0.68	79.1	14.2	11.6	47.8	14.6

training epochs (note that the numbers in the horizontal axis do not reflect the actual numbers of epoch). According to Figure 3a, the training loss stops decreasing twice during the entire training process. We believe that tuning the learning rate with a better strategy at these moments might improve the performance of VoteNet.

6.2 Decouple Results

We compare the average precision of each object category to understand if the network performs particularly terrible on any of them. As shown in Figure 3b, the blue and yellow line denote the training and testing results, respectively. Firstly, since the training and evaluating results align closely with each other, we may infer that the network does not overfit to the training set. Secondly, we could observe the results of the bookshelf, desk, and dresser are worse than other categories, implying that improvements in these categories would increase the overall performance significantly.

7 Improvement Strategies and Results

We tried three strategies to improve the performance of VoteNet. We describe them separately in the following sections and show all the results together for comparison in Table 2. Note that given the limited amount of time and computational resource available (it takes more than 2 days to train VoteNet once on the provided AWS machine), we did not have the opportunity to run experiment multiple times for rigorous verification. Therefore, even though some strategies seem promising, more experiments are undoubtedly needed for further confirmation.

7.1 Hyperparameter tuning

We first tune the decay schedule for the learning rate. According to the original paper, they reduce the learning rate by 10x at the 80, 120, and 160 epochs. We either forward or postpone the decay schedule comparing to the original strategy. According to the second and the third rows in Table 2, decreasing the learning rate earlier may lead to better performance. We suspect that training the

network longer with the smallest learning rate contributes to this improvement (all the experiments stop training at the 180 epoch).

We also tune the weight of each element in the loss function. Our hypothesis is that: if the performance improves as we increase the weight of a loss element, it may imply that enhancing the function that the loss element is responsible for could be a direction for network improvement. We run experiments three times by increasing a, b, or c and keeping the other weights the same. As shown in the fourth to the sixth rows in Table 2, increasing the weight of $L_{obj-cls}$ contributes to a slightly higher mAP. As mentioned above, since we only run the experiment once, more investigations are needed to verify whether increasing the weight of $L_{obj-cls}$ leads to consistent improvement.

7.2 Include RGB features

In the original paper, VoteNet takes only the XYZ coordinates of point clouds as input. One straightforward idea for improvement is to include RGB channels for more information. This idea is similar to our daily experience, where colors help us identify objects easier. Thus, we include the RGB feature as input and keep its dimension throughout the network along with XYZ coordinates. Surprisingly, this strategy does not improve but harm performance. We do not have a conclusion for this result yet and argue that more experiments are needed.

7.3 Backbone Modification

The original paper adopts PointNet++ [10] as their feature learning backbone. With an aim for learning better feature representations, we add a Voxel Feature Encoding (VFE) layer [17] after the grouping layers of PointNet++. The VFE layer takes the output groups from PointNet++ as input voxels, transforms voxels into a feature space using a fully connected network (FCN), and outputs encoded features by aggregation in the feature space. The FCN includes a linear layer, a batch normalization layer, and a rectified linear unit (ReLU) layer. After FCN, we apply element-wise max-pooling across all points in the feature space to obtain aggregated local features, which are concatenated with the aforementioned features to form the final output of the VFE layer. We believe this backbone with aggregated local information may provide better feature representation for point clouds. However, according to the results shown in the last row of Table 2, the mAP drops drastically to 14.6. We need more experiments to understand the reason behind this performance drop.

8 Conclusion

In this project, we study VoteNet for 3D object detection in point clouds. We first survey related literature to understand key ideas that make VoteNet the state-of-the-art network. We then reproduce VoteNet to achieve comparable quantitative and qualitative results. We further conduct an error analysis of the possible spaces for improvements on VoteNet. Finally, we propose three improvement strategies and report our findings for future research directions.

For future research, with more time and computational resources available, we are interested in rigorously validate current experimental results. We also hope to investigate the underlying reason that causes performance drop when we modify the backbone. Last but not least, we find the recently proposed Sparse Convolutional Network [4] has a better design in capturing spatial information strategically. We believe that adopting this network as the feature learning backbone may lead to significantly better performance.

Code

Our code is publically available at https://github.com/yuyuchang/CS230_project. The source code of VoteNet is available at https://github.com/facebookresearch/votenet

Contributions

- **Yen-Yu Chang**: idea brainstorming, environmental setting, project proposal, reproduction, include RGB features, backbone modification
- **Tzu-Sheng Kuo**: idea brainstorming, project proposal, reproduction, project milestone, visualization, final report, final poster
- **Shao-Yuan Ho**: error analysis, hyperparameter tuning, final report

Acknowledgement

We thank Dr. Or Litany for valuable guidance and feedback.

References

[1] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-View 3d Object Detection Network for Autonomous Driving. *arXiv:1611.07759 [cs]*, Nov. 2016. URL `http://arxiv.org/abs/1611.07759`. arXiv: 1611.07759.

[2] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, June 2014. doi: 10.1109/CVPR.2014.81.

[4] B. Graham and L. van der Maaten. Submanifold Sparse Convolutional Networks. *arXiv:1706.01307 [cs]*, June 2017. URL `http://arxiv.org/abs/1706.01307`. arXiv: 1706.01307.

[5] H. Ji, A. Dai, and M. Nießner. 3d-sis: 3d semantic instance segmentation of rgb-d scans. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019.

[6] J. Lahoud and B. Ghanem. 2d-driven 3d object detection in rgb-d images. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4632–4640, Oct 2017. doi: 10.1109/ICCV.2017.495.

[7] Y. Li, A. Dai, L. Guibas, and M. NieBner. Database-assisted object retrieval for real-time 3d reconstruction. *Comput. Graph. Forum*, 34(2):435–446, May 2015. ISSN 0167-7055. doi: 10.1111/cgf.12573. URL `https://doi.org/10.1111/cgf.12573`.

[8] D. Lin, S. Fidler, and R. Urtasun. Holistic scene understanding for 3d object detection with rgbd cameras. In *2013 IEEE International Conference on Computer Vision*, pages 1417–1424, Dec 2013. doi: 10.1109/ICCV.2013.179.

[9] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep Learning on Point Sets for 3d Classification and Segmentation. *arXiv:1612.00593 [cs]*, Dec. 2016. URL `http://arxiv.org/abs/1612.00593`. arXiv: 1612.00593.

[10] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *arXiv:1706.02413 [cs]*, June 2017. URL `http://arxiv.org/abs/1706.02413`. arXiv: 1706.02413.

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

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, June 2016. doi: 10.1109/CVPR.2016.91.

[13] S. Song and J. Xiao. Sliding shapes for 3d object detection in depth images. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 634–651, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10599-4.

[14] S. Song and J. Xiao. Deep Sliding Shapes for Amodal 3d Object Detection in RGB-D Images. *arXiv:1511.02300 [cs]*, Nov. 2015. URL `http://arxiv.org/abs/1511.02300`. arXiv: 1511.02300.

[15] S. Song and J. Xiao. Deep Sliding Shapes for amodal 3D object detection in RGB-D images. 2016.

[16] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 567–576, June 2015. doi: 10.1109/CVPR.2015.7298655.

[17] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, June 2018. doi: 10.1109/CVPR.2018.00472.