Effective Backbone Network for 3D Object Detection in Point Cloud

Jun Xu¹, Yanxin Ma², Yanxin Ma², and Songhua He¹, c

¹ College of Information Science and Engineering, Hunan University, Changsha, China
² College of Meteorology and Oceanography, National University of Defense Technology, Changsha, China

¹15738335219@139.com, bmayanxin@nudt.edu.cn, c13973132618@139.com

Abstract. Three-dimensional (3D) object detection is composed of object classification and object localization, and has been used in many applications, such as autonomous driving and mobile robot. However, the accuracy of classification and localization is greatly affected by the depth of the network. Shallow networks tend to cause poor classification, but as the depth of the network increases, the network degradation will become more obvious, which is not conducive to the training of network. To solve this problem, a novel Backbone Network is proposed in this paper for 3D object detection in point cloud, which consists of multiple residual modules. Experimental results on the KITTI 3D object detection benchmarks show that Backbone Network proposed can effectively improve the accuracy of 3D object detection.

1. Introduction

Three-dimensional (3D) object detection is a challenging subject in the field of computer vision, and is widely used in various practical applications, such as autonomous driving [1, 2], video surveillance [3], mobile robot [4] and so on. It can be divided into two sub-tasks: object classification and object localization. Object classification is to accurately determine the category of the object, object localization is to give an accurate 3D bounding box of the object in a given scene. In recent years, deep learning has become an important research direction of computer vision with its powerful feature learning ability, and has made substantial progress in many fields, including 3D object detection.

Sufficient depth plays a key role in the success of deep learning model in various tasks. The main module of deep learning network structure is a standard non-linear transformation module, which consists of convolution layer, pooling layer and activation layer. The deeper the model is, the stronger its non-linear expression ability is, allowing it to learn more complex transformations and fit more complex feature inputs [5]. Although deepening the network can improve the performance of the model to a certain extent, it does not mean that the performance of the network is directly proportional to the depth, which is mainly reflected in the improvement of performance and optimization. First of all, when the depth of the network reaches a certain level, the performance of network tends to be saturated and will not increase with the increase of depth. In this case, the deepening of the network can only bring expensive time cost. Secondly, the deepening of the model may also lead to the decline in the learning ability of some shallow levels, which limits the learning of some deep levels. Last but not least, the problem of network degradation caused by deep network always exists. Although it can be alleviated, it cannot be eliminated [6]. Therefore, it is possible that with the deepening of the network, performance will begin to decline.
Network degradation means that in the process of increasing network depth, the training accuracy tends to be saturated gradually, and the training accuracy will decline if we continue to increase the depth of network. Moreover, this reduction in accuracy is not caused by overfitting. To solve this issue in 3D object detection network, we use the residual learning model proposed by He et al. [6], and its basic structure is shown in Fig. 1. The experimental results show that the use of residual module ensures good network performance when training deeper networks.

2. Related Work

3D Object Detection. The existing 3D object detection methods can be divided into three categories by the data, i.e. monocular image based, 3D point cloud based and multi-model fusion based methods.

Monocular Image Based Detection. Nowadays, monocular image based two-dimensional (2D) object detection has achieved superior performance in both speed and accuracy. However, due to the lack of spatial structure information, 3D object detection still faces great challenges. Even so, some studies [7, 8] are still devoted to the research of 3D object detection in monocular image, since it is a basic problem and the equipment for obtaining image is much cheaper. DS3D [8] first uses 2D object detection technology to obtain 2D candidate boxes from the image. Then each candidate box is inferred into 3D space by combining the prior information of the object. Next, three view features of the 3D candidate box are extracted from the Bird's-Eye View (BEV), the Front View (FV) and the side view. Finally, these features are merged and the merged feature is used to refine the 3D bounding box to improve the performance of detection. Unfortunately, the accuracy of 3D object detection in monocular image is far from meeting the needs of applications such as autonomous driving.

3D Point Cloud Based Detection. The 3D point cloud is irregular data. To apply Convolutional Neural Network (CNN) whose input must be regular, there are mainly two ways to process point cloud: projection [1, 9] and voxelization [2, 10]. Although projecting point cloud onto one or more 2D planes can improve computational efficiency, it is easy to lose spatial structure information. Therefore, many technologies [2, 10] convert point cloud into regular data by rasterizing it into 3D voxel grids, and then the regular data is input to the CNN.

In addition, some studies have proposed direct processing of point cloud [11, 12]. PointRCNN [12] is a two-stage object detector. In the first stage, each point is classified using PointNet [11] and a large number of candidate boxes are generated. In the second stage, redundant candidate boxes are discarded according to the category information and the remaining candidate boxes are refined. Since all input points of PointRCNN [12] to be classified and produce corresponding candidate boxes, the detection is slow, although PointRCNN [12] achieves good performance on the KITTI dataset [13]. Considering the requirements of accuracy and speed, voxelization is used to convert point cloud into regular data in this paper.

Multi-model Fusion Based Detection. Some techniques [14 - 17] have achieved good detection performance by fusing RGB image with point cloud. These algorithms use CNN to extract the appearance information of RGB image and structure information of point cloud respectively, and then use the fused feature to determine the final detection results. There are also some studies to improve the accuracy of 3D object detection by fusing RGB image and depth image [18, 19]. Lukas et al. [18] designed an object detection framework, which consists of Network-in-Network (NIN) [20] and GoogLeNet [21]. In detail, NIN [20] uses Multi-Layer Perceptron (MLP) instead of traditional convolution layer, the nonlinearity of MLP makes NIN [20] more generalization. Combining multiple models tends to improve detection accuracy, however, the speed is relatively slow, since the large number of features to be processed.

Residual Network. The depth of the network is critical for the performance of the network. However, simply stacking networks will lead to inevitable network degradation. To solve this issue, He et al. proposed the residual network [6] which consisting of several residual blocks. The architecture of the residual block is shown in Fig. 1. The network is designed as $H(X) = F(X) + X$ through a short connection, so a residual function $F(X) = H(X) - X$ to be learned by the network. This method will not cause additional parameters and calculations. The most important is that when the network is deepened,
the residual block can solve the problem of network degradation very well. Consequently, it has been applied to various tasks, such as object classification [6], object detection [18] and so on.

3. Our Network

The architecture proposed will be introduced in this section. As shown in Fig. 2, it consists of four parts: (1) Data preprocessing; (2) Feature Learning Network; (3) Backbone Network; (4) Header Network.

**Data Preprocessing.** To apply CNN, it is necessary to convert the point cloud into regular data. Firstly, we crop point cloud using a fixed dimension $L \times W \times H$ m$^3$ in the directions of X, Y and Z axes. Then the cropped point cloud is discretized into an evenly spaced grid in the x-y plane with size $D_x$ and $D_y$. Consequently, a total $L_1 \times W_1$ pillars are generated, where $L_1 = L / D_x$, $W_1 = W / D_y$. In addition, to reduce the sampling deviation between the pillars, we use random down-sampling to fix the number of points in each non-empty pillar to a number $N$.

**Feature Learning Network.** To obtain a more expressive feature, any point in a non-empty pillar is represented as a vector $p_i = \{x, y, z, r, x-\triangle x, y-\triangle y, z-\triangle z, x-x_c, y-y_c\}$ ($i \leq N$), where $(x, y, z, r)$ represents the coordinates and reflectance of the point itself, $(\triangle x, \triangle y, \triangle z)$ represents the mean coordinates of all points in each pillar, and $(x_c, y_c)$ represents the x, y center of the pillar. Then, a simplified version PointNet is used in each non-empty pillar to learn pillar-wise feature. In detail, the simplified PointNet consists of a linear layer, a Batch Normalization (BN) layer and Rectified Linear Unit (ReLU) layer. After that, each non-empty pillar is represented as a 128D vector. And the empty pillars are filled with a 128D zero vector. As a result, the point cloud can be represented by a pseudo-image of size $L_1 \times W_1 \times 128$.

**Backbone Network.** Conv2d is the 2D convolutional neural network, and K and S are the convolutional kernel and stride respectively. Deconv2d is the 2D deconvolutional neural network.
Backbone Network. Some studies have used multiple convolutional layers in the Backbone Network to obtain better classification results [2, 10, 22]. However, the deepening of the network is likely to cause network degradation. In order to end the problem, we propose a novel Backbone Network for 3D object detection in the point cloud, and its structure is shown in Fig. 3. It is the unidirectional feed-forward computation of 2D CNN, which computes the pyramidal 2D feature hierarchy. Every network layer consists of a convolution layer and a residual block. In particular, the structure and parameters of residual block in each network layer are the same, as shown in Fig. 3. Each residual block is composed of three convolutional layers, and each layer is followed by a BN layer and a ReLU layer. In addition, to aggregate high-level features and high-level features, three deconvolution layers are applied to the output of three convolution blocks. After that, three feature maps with different levels but the same size are obtained, and then these feature maps are inputted to the Header Network for classification and regression.

Header Network. The aim of Header Network is to detection objects from feature maps generated by the Backbone Network. All the 2D feature map will be fused to concatenated together to form a stronger semantic feature map for classification and regression tasks.

Figure 4. Header Network: DCNN is the 2D deconvolutional neural network, Conv2d is the 2D convolutional neural network, and K and S are the convolutional kernel and stride respectively.

4. Experiment and Results
All experiments are performed on the challenging KITTI benchmarks [13], which contains 7481 training point clouds and corresponding RGB images, covering three categories: Car, Cyclist and Pedestrian. Because the label of the testing dataset is not available, the training dataset is divided into training set (3712) and validation (3769). Moreover, the loss function and the parameters of network will be described in this section.

Loss function. The loss of network consists of two parts: classes classification and 3D bounding box regression. As shown in Eq. 1, different weights are used to balance relative importance, where $\rho=1$ and $\mu=2$.

$$L_{all} = \rho L_{cls} + \mu L_{reg}. \quad (1)$$

Classes Classification $L_{cls}$. For the object classification loss, focal loss proposed by Lin et al. [23] is used, which is defined as follows:

$$L_{cls} = -\alpha(1 - p)^{\gamma} \log p. \quad (2)$$

3D Bounding Box Regression $L_{reg}$. For the 3D bounding box regression task, we use the same loss functions defined in PointPillars [22]. Anchors and ground truths are represented by 7 parameters, i.e. central coordinates, dimensions and rotation angles around the Z axis. Consequently, the regression targets are defined as a vector, $(\Delta x, \Delta y, \Delta z, \Delta l, \Delta w, \Delta h, \Delta \theta)$, and details are as follows:
\[ \Delta x = (x^g - x^a)/d^a. \quad \Delta y = (y^g - y^a)/d^a. \quad \Delta z = (z^g - z^a)/h^a. \]
\[ \Delta l = \log(l^g/l^a). \quad \Delta w = \log(w^g/w^a). \quad \Delta h = \log(h^g/h^a). \quad \Delta \theta = \sin(\theta^g - \theta^a). \]

(3)

\[ L_{\text{reg}} = \sum_{r \in \{x,y,z,l,w,h,\theta\}} \text{SmoothL}_1(\Delta r). \]

(4)

where \(x^g\) and \(x^a\) are the ground truth and anchor respectively, and \((d^a)^2 = (l^a)^2 + (w^a)^2\).

**Setting.** Considering that the dimensions of Car, Cyclist and Pedestrian are quite different, different sizes are used for cropping and rasterization of point clouds in data preprocessing.

**Car Detection.** The \(x, y, z\) range of point cloud is \(L \times W \times H = [0, 70.4] \times [-40, 40] \times [-3, 1]\)m\(^3\), and the pillar size is set to be \(D_x = D_y = 0.16m\). Following the PointPillars [22], the dimension of the anchor is set to \((3.9m, 1.6m, 1.5m)\) with a \(z\) center of \(z = -1m\).

**Cyclist and Pedestrian Detection.** The \(x, y, z\) range of point cloud is \(L \times W \times H = [0, 47.36] \times [-19.84, 19.84] \times [-2.5, 0.5]\)m\(^3\), and the pillar size is set to be \(D_x = D_y = 0.16m\). The dimension of the cyclist anchor is set to \((1.76m, 0.6m, 1.73m)\) while the pedestrian anchor is \((0.8m, 0.6m, 1.73m)\). Moreover, all the anchors are centered at \(z = -0.6m\).

**Results and Qualitative Analysis.** The proposed network is tested on the KITTI verification set, and we compare it with other 3D detection methods. In addition, to better prove the effectiveness of our Backbone Network, some ablation experiments are conducted with the basic network, i.e. PointPillars [22]. The 3D and BEV detection results are given in Tab. 1 and Tab. 2, respectively. The number in parentheses indicates the number of convolutional layers with a stride of 1 in each convolution block. For better visualization, (a), (b) and (c) in Fig. 5 give several examples of three categories respectively.

![Visualization of 3D object detection on KITTI validation dataset. All the 2D and 3D bounding boxes in RGB images are obtained from the result of detection. In the point clouds, red 3D boxes represent ground trues while teal 3D boxes indicated that have been successfully detected.](image)

**Figure 5.** Visualization of 3D object detection on KITTI validation dataset. All the 2D and 3D bounding boxes in RGB images are obtained from the result of detection. In the point clouds, red 3D boxes represent ground trues while teal 3D boxes indicated that have been successfully detected.

**Table 1.** 3D detection performance: Average precision (AP) (in %) for 3D boxes in the KITTI validation set.

| Method   | Modality      | Car Easy | Car Mod. | Car Hard | Cyclist Easy | Cyclist Mod. | Cyclist Hard | Pedestrian Easy | Pedestrian Mod. | Pedestrian Hard |
|----------|---------------|----------|----------|----------|--------------|--------------|--------------|-----------------|-----------------|-----------------|
| MV3D     | Img & Lidar   | 71.09    | 62.35    | 55.12    | N/A          | N/A          | N/A          | N/A             | N/A             | N/A             |
| AVOD     | Img & Lidar   | 81.94    | 71.88    | 66.38    | 64.00        | 52.18        | 46.61        | 50.80           | 42.81           | 40.88           |
| F-PointNet| Img & Lidar   | 81.20    | 70.39    | 62.19    | 71.96        | 56.77        | 50.39        | 51.21           | 44.89           | 51.21           |
Table 2. BEV detection performance: Average precision (AP) (in %) for BEV boxes in the KITTI validation set.

| Method          | Modality       | Car          | Cyclist      | Pedestrian   |
|-----------------|----------------|--------------|--------------|--------------|
|                 |                | Easy  | Mod.  | Hard  | Easy  | Mod.  | Hard  | Easy  | Mod.  | Hard  |
| MV3D            | Img.&Lidar     | 86.02 | 76.90 | 68.48 | N/A   | N/A   | N/A   | N/A   | N/A   | N/A   |
| AVOD            | Img.&Lidar     | 88.53 | 83.79 | 77.90 | 68.09 | 57.48 | 50.77 | 58.75 | 51.05 | 47.54 |
| F-PointNet      | Img.&Lidar     | 88.70 | 84.00 | 75.33 | 75.38 | 61.96 | 54.68 | 58.09 | 50.22 | 47.02 |
| VoxelNet        | Lidar          | 89.35 | 79.26 | 77.39 | 66.07 | 54.76 | 50.55 | 46.13 | 40.74 | 38.11 |
| SECOND V1       | Lidar          | 89.23 | 86.25 | 78.95 | 73.67 | 65.04 | 48.78 | 55.10 | 46.27 | 44.76 |
| SECOND V1.5     | Lidar          | 89.61 | 85.73 | 78.51 | 73.61 | 63.58 | 56.16 | 57.62 | 49.35 | 46.57 |
| PointPillars (3)| Lidar          | 89.87 | 87.36 | 84.98 | 81.76 | 61.07 | 58.20 | 68.94 | 64.72 | 60.18 |
| Ours (1×3)      | Lidar          | 90.10 | 87.40 | 85.04 | 83.54 | 64.57 | 60.98 | 73.85 | 68.87 | 63.47 |
| PointPillars (6)| Lidar          | 90.11 | 87.16 | 84.98 | 83.30 | 62.84 | 59.55 | 66.45 | 63.35 | 58.71 |
| Ours (2×3)      | Lidar          | 90.27 | 88.03 | 86.49 | 84.96 | 64.83 | 61.76 | 72.19 | 67.47 | 61.79 |

**Car Detection.** For car detection task, as shown in Tab. 1 and Tab. 2, the detection performance of the proposed Backbone Network on KITTI validation set is significantly better than other algorithms listed, both of 3D and BEV. Compared with the basic network PointPillars [22], our network achieves better results with respect to average precision (AP).

**Cyclist and Pedestrian Detection.** Compared with the task of car detection, the accuracy of cyclist and pedestrian detection is slightly poor. There are two main reasons: firstly, the network training is insufficient due to the lack of cyclists and pedestrians. In the training set, there were 10520 cars, while only 594 cyclists and 2104 pedestrians. Another reason is that the sizes of cyclists and pedestrians are small and their distribution is dense, which makes detection difficult. This is more evident in pedestrian detection task. However, the accuracy of our method is much higher than other algorithms, especially for the pedestrian, the 3D detection performance of our method is 3.74% higher than that of PointPillars [22] for the Hard level. In addition, we can see from the results of the ablation experiment that shallow network is more suitable for pedestrian detection.

5. **Summary**

To solve the problem of degradation caused by network deepening, a novel Backbone Network is proposed in this paper, which consists of several residual modules. The results of the ablation experiments on the challenging KITTI validation set show that the performance our network is better than that of PointPillars [22] under the same network depth. In addition, the results of comparison with other methods also fully demonstrate that the proposed Backbone Network is more suitable for 3D object detection.

**References**

[1] L. Bo, Z. Tianlei, X. Tian. Vehicle detection from 3D lidar using fully convolutional network. arXiv 2016, arXiv:1608.07916, (2016).

[2] Z. Yin, O. Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4490–4499, (2018).
[3] J. Chengbin, L. Shengzhe, T. D. Do, H. Kim. Real-Time Human Action Recognition Using CNN Over Temporal Images for Static Video Surveillance Cameras. Advances in Multimedia Information Processing – PCM 2015, pp. 330-339, (2015).
[4] F. Pomerleau, F. Colas, R. Siegwart. A Review of Point Cloud Registration Algorithms for Mobile Robotics. Foundations and Trends® in Robotics, (2015).
[5] M. Bianchini, F. Scarselli. On the complexity of neural network classifiers: A comparison between shallow and deep architectures[J]. IEEE transactions on neural networks and learning systems, pp. 1553-1565, (2014).
[6] H. Kaiming, Z. Xiangyu, R. Shaoqing, S. Jian. Deep Residual Learning for Image Recognition. The IEEE Conference on Computer Vision and Pattern Recognition, pp. 770-778, (2016).
[7] C. Xiaozhi, K. Kundu, Z. Ziyu, M. Huimin, S. Fidler, R. Urtasun. Monocular 3d object detection for autonomous driving. In the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2147–2156, (2016).
[8] L. Buyu, O. Y. Wanli, L. Sheng, Z. Xingyu, W. Xiaogang. GS3D: An Efficient 3D Object Detection Framework for Autonomous Driving. The IEEE Conference on Computer Vision and Pattern Recognition, pp. 1019-1028, (2019).
[9] Y. Bin, L. Wenjie, R. Urtasun. PIXOR: Real-time 3D Object Detection from Point Clouds. The IEEE Conference on Computer Vision and Pattern Recognition, pp. 7652–7660, (2018).
[10] Y. Yan, M. Yuxing, L. Bo. SECOND: Sparsely Embedded Convolutional Detection. Sensors (2018).
[11] C. R. Qi, S. Hao, K. Mo, Guibas, L. J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 77–85, (2017).
[12] S. Shaoshuai, W. Xiaogang, L. Hongsheng. PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud. The IEEE Conference on Computer Vision and Pattern Recognition, pp. 770-779, (2019).
[13] Kitti 3D Object Detection Benchmark Leader Board. Available online: http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d (2018).
[14] C. Xiaozhi, M. Huimin, W. Ji, L. Bo, X. Tian. Multi-view 3D object detection network for autonomous driving. In Proceedings of the IEEE Computer Vision and Pattern Recognition, Honolulu, Volume 1, pp. 1907-1915, (2017).
[15] C. R. Qi, L. Wei, W. Chenxia, S. Hao, L. J. Guibas. Frustum PointNets for 3D Object Detection from RGB-D Data. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, pp. 918–927, (2018).
[16] R. Huitl, G. Schroth, S. Hilsenbeck, F. Schweiger, E. Steinbach. TUMindoor: An extensive image and point cloud dataset for visual indoor localization and mapping. IEEE International Conference on Image Processing, (2012).
[17] J. Ku, M. Mozifian, J. Lee, A. Harakeh, S. L. Waslander. Joint 3D Proposal Generation and Object Detection from View Aggregation. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1–8, (2018).
[18] L. Schneider, M. Jasch, B. Frohlich, T. Weber, U. Franke, M. Pollefeys, M. Rätsch. Multimodal Neural Networks: RGB-D for Semantic Segmentation and Object Detection, pp 98-109, (2017).
[19] D. Zhuo, L. J. Latecki. Amodal Detection of 3D Objects: Inferring 3D Bounding Boxes from 2D Ones in RGB-Depth Images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5762-5770, (2017).
[20] L. Min, C. Qiang, Y. Shuicheng. Network in Network. arXiv 2013, arXiv:1312.4400. (2013)
[21] C. Szegedy, L. Wei, J. Yangqing, P. Sermanet, S. Reed, A. Dragomir, D. Erhan, V. Vanhoucke, A. Rabinovich. Going Deeper with Convolutions. The IEEE Conference on Computer Vision and Pattern Recognition, pp. 1-9, (2015).
[22] A.H. Lang, S. Vora, H. Caesar, Z. Lubing, Y. Jiong, O. Beijbom. PointPillars: Fast Encoders for Object Detection from Point Clouds. The IEEE Conference on Computer Vision and Pattern Recognition, pp. 12697-12705, (2019).

[23] T.-Y. Lin, P. Dollar, R. Girshick, H. Kaiming, B. Hariharan, S. Belongie. Feature Pyramid Networks for Object Detection. CVPR, pp. 936–944, (2017).