JIMDT, Vol. 6, 2024
RESEARCH ARTICLE
https://doi.org/10.5564/jimdt.v6i1.3609

3d object detection from point cloud

Minjinsor Myagmarsuren[1*]

[1]*Institute of Mathematics and Digital Technology, Mongolian Academy of Sciences, Ulaanbaatar 13330, Mongolia*

**Corresponding author: mmsuren0909@gmail.com; ORCID:0009-0004-5153-7390*

Article Info: Received: 2024.09.15; Accepted: 2024.10.01; Published: 2024.12.26

Abstract: Here we present a comparison of two different deep learning architectures' effectiveness along with two distinct detection head approaches for detecting point cloud ball objects. Two backbones that are explored are: VoxelNet, suited for organized point clouds, and PointNet, which handles unorganized point clouds. We modified and implemented SSD and Faster R-CNN detection heads for both backbones. It turns out that the PointNet backbone integrated with a customized Faster R-CNN detection head achieved higher accuracy compared to other combination of models.

Key words: computer vision, deep learning, object detection, point cloud

1. Introduction

Detecting objects from point cloud data accurately is one of the essential and concerning tasks in various applications, such as autonomous driving, medical imaging, augmented reality, and so on. Point-cloud data that are captured by depth cameras or lidar, provide a rich representation of the 3D environment. But analyzing and processing point clouds become a bit challenging when it comes to their unstructured nature and the presence of noise. Classic approaches like VoxelNet, which converts point cloud data into a structured voxel grid [1], are effective, but small object detection and its details can be performed poorly. However, PointNet, which is known for its ability to process unstructured data, has potential pros and cons, including sensitivity to input density. This study aims to compare the effectiveness of VoxelNet and PointNet architectures as backbone and exchanging a customized SSD and Faster R-CNN detection heads for detecting ball objects from point cloud data, evaluating their performance in terms of Average Precision (AP).

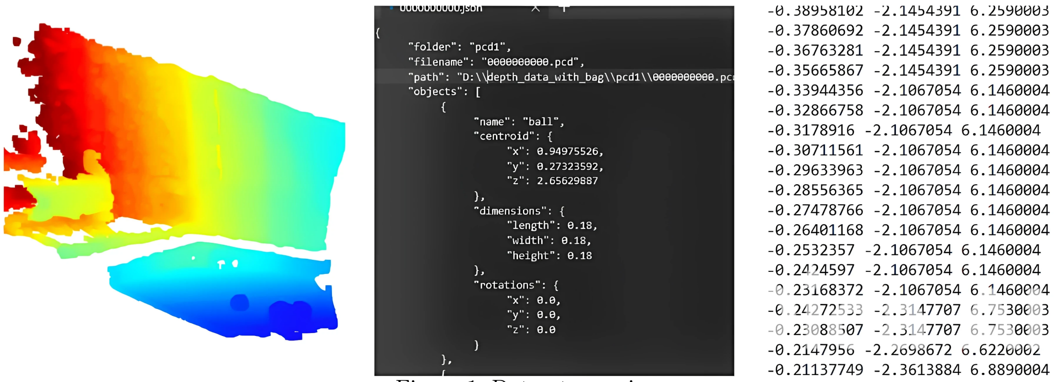
As mentioned before, 3D understanding is an extensively researched topic. Earlier approaches ([1], [2], [5], [6]) achieved satisfactory results for their own needs. Some of the techniques ([1], [6]) represent 3D point cloud data using the voxel occupancy grid representation, followed by the use of 3D convolutions to compute the 3D bounding boxes.([15]) created voxels containing feature vectors as well. Due to the high computational and memory costs, several approaches based on BEV representation were developed ([1]). ([9]) converted 3D data into a 2D BEV image to propose a detector called SECOND along with a voxel-wise feature extractor. Furthermore, a specific height-encoded BEV input is used to design a fast single-stage detector([12]). ComplexYOLO([13]) uses a YOLO([14]) network and specific encoding method to increase the 3D bounding box detection speed. In typical image-based methods ([10]), hand-crafted approaches are used to generate feature maps. Generally, a common modern approach for detecting 3D objects is to combine different backbone architectures with specialized detection heads([3], [4]) to leverage their respective strengths([7]). ([8], [11]) implemented a fully convolutional one-stage 3D object detector for the LiDAR point cloud, for ([8]) built model is called FCOS-LiDAR. ([16]) is a comprehensive review of recent progress in deep learning methods for point clouds, covering three major tasks. This work aims to investigate how different approaches can be combined.

2. Research materials and data

2.1. Dataset

Point cloud data is captured using an ASUS X-tion depth camera with no RGB, sizes for ball objects were annotated with the free open-source tool LabelCloud. A total of 20 frames of point cloud data with a .pcd extension were captured, each containing an average of 3-4 balls or no balls. The maximum and minimum number of points are 307188 and 125429 respectively while the average number of points in the data is 213983.

Each .pcd file in the dataset is formatted as an each row in the file represents one point in 3d space and 3 columns represent the coordinates of the point. The label files are in JSON format and contain the centroid, dimensions, and rotations of the bounding box for the ball object. Splitted the dataset into training, validation, and test sets in an 80:10:10 ratio.



```
{
    "folder": "pcd1",
    "filename": "0000000000.pcd",
    "path": "D:\\depth_data_with_bag\\pcd1\\0000000000.pc
    "objects": [
        {
            "name": "ball",
            "centroid": {
                "x": 0.94975526,
                "y": 0.27323592,
                "z": 2.65629887
            },
            "dimensions": {
                "length": 0.18,
                "width": 0.18,
                "height": 0.18
            },
            "rotations": {
                "x": 0.0,
                "y": 0.0,
                "z": 0.0
            }
        },
```

```
-0.38958102 -2.1454391 6.2590003
-0.37860692 -2.1454391 6.2590003
-0.36763281 -2.1454391 6.2590003
-0.35665867 -2.1454391 6.2590003
-0.33944356 -2.1067054 6.1460004
-0.32866758 -2.1067054 6.1460004
-0.3178916 -2.1067054 6.1460004
-0.30711561 -2.1067054 6.1460004
-0.29633963 -2.1067054 6.1460004
-0.28556365 -2.1067054 6.1460004
-0.27478766 -2.1067054 6.1460004
-0.26401168 -2.1067054 6.1460004
-0.2532357 -2.1067054 6.1460004
-0.2424597 -2.1067054 6.1460004
-0.23168372 -2.1067054 6.1460004
-0.24272533 -2.3147707 6.7530003
-0.23088507 -2.3147707 6.7530003
-0.2147956 -2.2698672 6.6220002
-0.21137749 -2.3613884 6.8890004
```

Figure 1: Dataset overview

2.2. Data augmentation and preprocessing

Applied the average downsampling approach to reduce point cloud density. It maintains a representative subset of the original data and reduces noise and computational cost. Handled varying point densities by padding with zero values. Also changed NaN values in the point cloud data to zero values due to problems that arose during training with PyTorch.

Implemented rotation and translation techniques both globally and locally to enhance the diversity of the dataset and improve model robustness. By augmenting the dataset, increased the number of instances in the dataset fivefold, resulting in a total of 100 instances.

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

3. Methodology

3.1. Model Implementation

We experimented with 2 different feature extractors, VoxelNet and PointNet, as backbone combining with 2 different detection heads, Faster R-CNN and SSD. All models are trained with batch size 4, learning rate 0.01, 100 epochs, SGD optimizer.

The PointNet backbone takes raw, unstructured nx3 input, extracts a 3x3 transformation matrix using T-Net section, multiplies the input with the matrix and passes it through a shared Multi-Layer Perceptron (MLP) which contains Conv1D, ReLU and BatchNorm techniques [2]. After that obtains another 64x64 transform matrix to use as a local feature map. By taking the below sequence of operations, max pooling is applied on the extracted nx1024 feature.

Both detection heads utilize context called anchor boxes, which are predefined bounding boxes of various sizes and aspect ratios that act as reference boxes across the point cloud. By analyzing the points in a dataset, we found that our data points are lying in an interval $x \in [-2.91, 0.18]$, $y \in [0.13, 0.5]$, $z \in [0.8, 1.5]$. Also interpreted that bounding box sizes were $minsizes : [0.13, 0.13, 0.13]$ and $maxsizes : [0.18, 0.18, 0.18]$ for each dimension. So, by this information, we used 10x10x10 grids to create 4 different ratio anchor boxes for each grid.

The Faster RCNN part is composed of 2 main components called Region Proposal Network (RPN) and Region of Interest (ROI) pooling [4]. The RPN takes the feature map that was extracted from the backbone layer as an input and applies a 3x3 convolutional layer with 512 output channels and then 1x1 convolutional layer with 36 output channels. Rather than directly predicting absolute bounding box coordinates, the network learns to predict offsets or adjustments from these anchor boxes [4].The figure below shows PointNet with Faster R-CNN architecture.

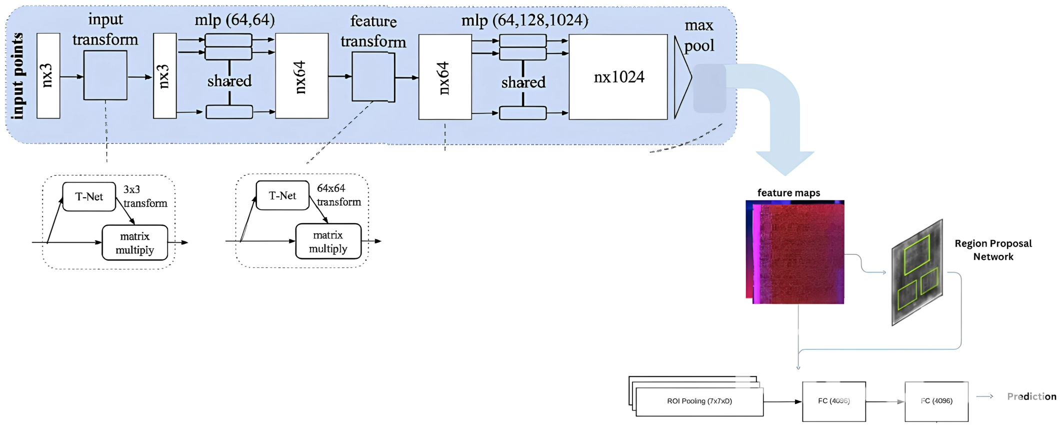


Figure 2: PointNet with faster RCNN head

VoxelNet is an end-to-end approach for detecting 3d objects. Originally, VoxelNet consisted of 3 functional blocks: Feature learning network (FLN), Convolutional middle layers (CML), and Region proposal network [6], but in this work, we dropped the region proposal network to customize it. In FLN component, point clouds are divided into equally spaced voxels and randomly sampled in each voxel due to data sparsity [1]. We flattened the input x, making it 2d data and passed it through the linear layer, batchNorm, ReLU activation, then reshaped the computed feature back into 3D. In CML, a sequence of three Conv3D layers is used with kernel size=3 and different stride, padding, and channel values. Each of them is followed by batch normalization and ReLU activation techniques.

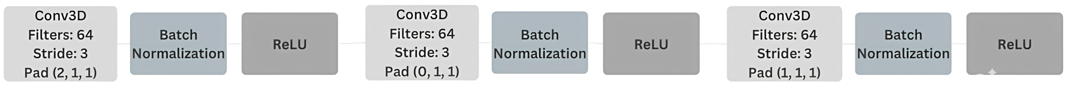

Conv3D Filters: 64 Stride: 3 Pad (2, 1, 1)
Batch Normalization
ReLU
Conv3D Filters: 64 Stride: 3 Pad (0, 1, 1)
Batch Normalization
ReLU
Conv3D Filters: 64 Stride: 3 Pad (1, 1, 1)
Batch Normalization
ReLU

Figure 3: Convolutional Middle Layer

Unlike Faster RCNN, Single Shot Detection (SSD) network is a single-stage detector that utilizes small convolutional filters, such as Conv4_3, applied to feature maps [3] to regress bounding boxes, even though it uses anchor boxes. In our case, features pass over the Conv1D layer that expects 4 input channels and outputs 6x3 channels (6 for a point representation and 3 for a default number of boxes to predict) with kernel size 1. After generating 15000 anchor boxes, we calculated offsets to find the best fits for ground truth boxes. The below equations show how prediction boxes are estimated by anchor boxes coordinates and predictions from Conv1D,

$$x' = x_a + t_x \cdot w_a, \qquad y' = y_a + t_y \cdot h_a, \qquad w' = w_a \cdot e^{t_w}, \qquad h' = h_a \cdot e^{t_h} \tag{3.1}$$

3.2. Loss calculation

In each of the four cases, the same loss function is calculated while only using bounding box regression loss to get the optimal result.
The localization loss between the predicted box l and the ground truth box g is defined as the smooth L1 loss with cx and cy as the offset to the default bounding box d, aka the anchor bounding box of width w and height h.

$$L_{\text{loc}}(x,l,g) = \sum_{i \in \text{pos}} \sum_{m \in \{cx,cy,w,h\}} x_{ij}^{k} \text{smooth}_{L1}(l_i^m - \hat{g}_j^m). \tag{3.2}$$

$$\hat{g}_j^{cx} = \frac{(g_j^{cx} - d_i^{cx})}{d_i^w}, \quad \hat{g}_j^{cy} = \frac{(g_j^{cy} - d_i^{cy})}{d_i^h}, \quad \hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right), \quad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right), \tag{3.3}$$

$$x_{ij}^p = \begin{cases} 1 & \text{if IoU} > 0.5 \text{ between default box } i \text{ and ground truth box } j \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

IoU in 3D is calculated as

$$\text{IoU}_{3D} = \frac{\text{Volume}_{\text{overlap}}}{\text{Volume}_g + \text{Volume}_d - \text{Volume}_{\text{overlap}}}. \tag{3.5}$$

Smooth L1 loss that mentioned above can be interpreted as

$$\text{SmoothL1}(x,y) = \begin{cases} 0.5(x-y)^2 & \text{if } |x-y| < 1 \\ |x-y| - 0.5 & \text{otherwise} \end{cases} \tag{3.6}$$

4. The results

Here we present a comparison of two different deep learning architectures' effectiveness alongside two distinct detection head approaches for detecting point cloud ball objects. Two backbones that are explored, are VoxelNet, suited for organized point clouds, and PointNet, which handles unorganized point clouds. We modified and implemented SSD and Faster R-CNN detection heads for both backbones. It turns out that PointNet backbone integrated with a customized Faster R-CNN detection head achieved higher accuracy compared to other model combinations.

The PointNet processes irregular and sparse point cloud data directly, unlike voxel-based methods (e.g., VoxelNet) which allows it to preserve fine geometric details and avoid quantization artifacts. Along with that, Faster R-CNN, is known for its two-stage detection process (an RPN and a classifier) that leads to better localizations' accuracy.

Evaluation: Performance was assessed using Average Precision (AP) metrics to compare the accuracy of bounding box detection between VoxelNet and PointNet. The result is yet lower than the other experimented state-of-the-art learning models that are conducted in the Deep Learning for 3D point clouds survey ([16]), such as 3P-RNN, PointCNN, and ConvPoint. Those surveyed methods are evaluated on the KITTI test 3D detection benchmark and resulted in relatively higher accuracies. Even though a comparison gap is high, our experiment is one of the examples of different approaches for 3D object detection and can enable accurate point cloud detection in sparse data.

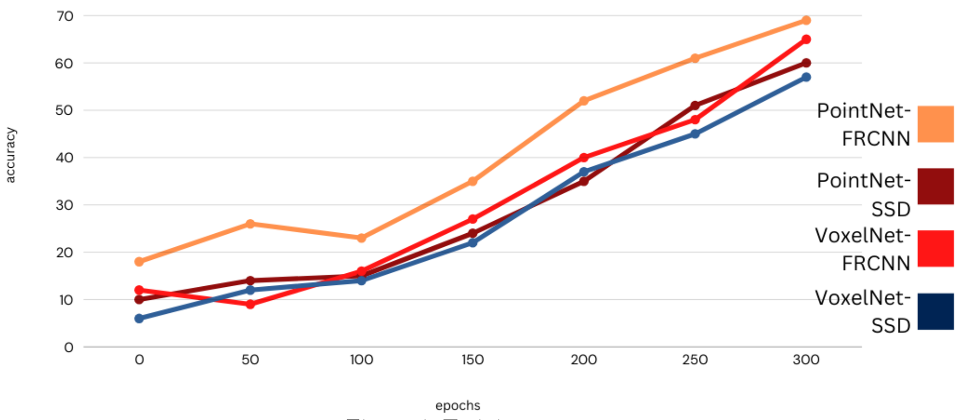


Figure 4: Training accuracy

Methods	PointNet-Faster RCNN	PointNet-SSD	VoxelNet-Faster R-CNN	VoxelNet-SSD
Ball	69.86	60.12	65.79	59.20

Table 1: Best accuracies of models

Aspect Ratios	(1:1:1)	(1.5:1:1)	(1:1.5:1)	(1:1:1.5)
mAP (%)	62.7	64.9	69.8	65.4

Table 2: Ratio values of PointNet-fasterRCNN

5. Conclusions

By combining cutting-edge architectures and modified detection heads, this research demonstrates the possibility of optimizing point cloud data computation. Additionally, examining those comparison analysis demonstrates that PointNet's ability to process raw data directly combined with Faster R-CNN's robust two-stage detection approach enables more precise localization output. Also, this kind of model would offer substantial benefits for tasks involving detailed object detection from point clouds. Further study may focus on improving an optimization for these works and applications of these methods in other cases of 3d space data processing.

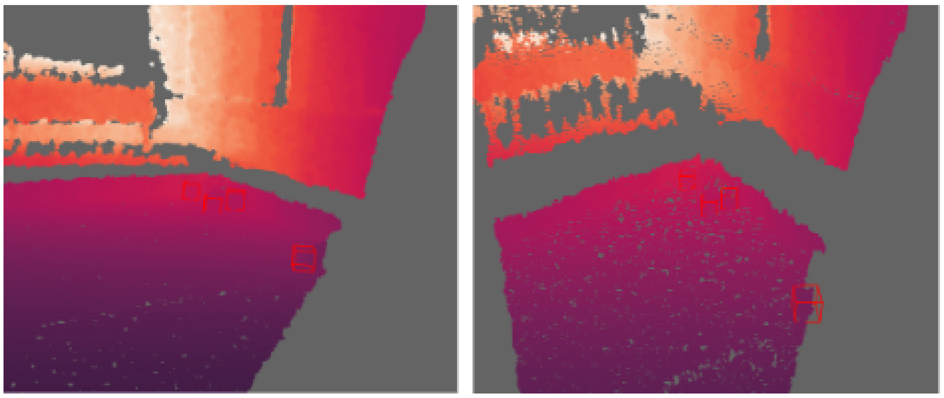


Figure 5: Result

References

[1] Y. Zhou and O. Tuzel, “VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection,” *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, https://doi.org/10.1109/CVPR.2018.00472.

[2] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation,” *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, https://doi.org/10.1109/CVPR.2017.16.

[3] S. Liu, L. Qi, H. Qin, and J. Shi, “SSD: Single Shot MultiBox Detector,” *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 21–37, https://doi.org/10.1007/978-3-319-46448-0_2.

[4] W. Liu *et al.*, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *arXiv preprint arXiv:1506.01497*, 2015.

[5] Y. Li *et al.*, “PointCNN: Convolution On X-Transformed Points,” *arXiv preprint arXiv:1812.04244*, 2018.

[6] V. A. Sindagi, Y. Zhou, and O. Tuzel, “MVX-Net: Multimodal VoxelNet for 3D Object Detection,” *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2019, pp. 7276–7282. doi: 10.1109/ICRA.2019.8793882.

[7] L. Tetrel, G. Ferret, and R. Cazorla, “3D Point Cloud Object Detection in LidarView,” *Kitware Software Company*, 2023.

[8] T. Zhi *et al.*, “Fully Convolutional One-Stage 3D Object Detection on LiDAR Range Images,” *Proc. NeurIPS*, 2022, https://doi.org/10.48550/arXiv.2205.13764.

[9] Y. Yan, Y. Mao, and B. Li, “SECOND: Sparsely Embedded Convolutional Detection,” *Sensors*, vol. 18, no. 10, 2018, https://doi.org/10.3390/s18103337.

[10] X. Chen *et al.*, “Monocular 3D Object Detection for Autonomous Driving,” *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2147–2156.

[11] B. Li, T. Zhang, and T. Xia, “Vehicle Detection from 3D LiDAR Using Fully Convolutional Network,” *arXiv preprint arXiv:1608.07916*, 2016.

[12] B. Yang, W. Luo, and R. Urtasun, “PIXOR: Real-Time 3D Object Detection From Point Clouds,” *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7652–7660.

[13] M. Simon, S. Milz, K. Amende, and H. M. Gross, “Complex-YOLO: Real-Time 3D Object Detection on Point Clouds,” *arXiv preprint arXiv:1803.06199*, 2018.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.

[15] D. Z. Wang and I. Posner, “Voting for Voting in Online Point Cloud Object Detection,” *Proc. Robot.: Sci. Syst. (RSS)*, 2015.

[16] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, “Deep Learning for 3D Point Clouds: A Survey,” *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019, pp. 4338–4364.