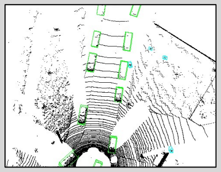
Optimization of Real-time 3D Object Detection Inference in Point Clouds

Anish Chougule   Alina Lazar
Youngstown State University

Abstract

Object detection in LiDAR point clouds is an important application especially for autonomous driving. So far, most work has been focused to develop deep learning architectures and to improve the detection performance. However, in addition to accuracy, inference speed is a very important component to quickly detect moving objects. We present an evaluation and comparison of the inference of state-of-the-art deep learning models for LiDAR-based 3D object detection. These models were created using the OpenPCDet library. All the experiments were performed on the KITTI benchmark dataset and show the differences in performance between the six models:

- PointPillars
- SECOND
- SECOND_IOU
- PvRCNN
- PointRCNN
- PointRCNN_IOU



Methods

To measure the inference time of the proposed 3D Object Detection models, LiDar technology is utilized to generate **point clouds** data.

Lidar is a method for determining ranges by targeting an object with a laser and measuring the time for the reflected light to return to the receiver. It can be used to make digital 3-D representations of areas on the earth's surface.

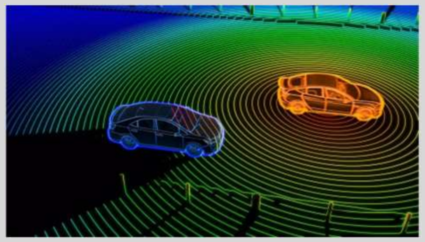


Figure 1 | Illustration of lidar

Point Cloud data is passed through 3 different modules-

1. Pillar Feature Extraction Network
2. Region Proposal Network
3. Detection Head

1. Pillar Feature Extraction Network (PFE):

- Point pillars are calculated by clustering of points in point clouds.
- Each point cloud in the stacked pillars is passed through linear, batch normalization, and ReLU (Rectified Linear Unit).
- Once features are extracted from PFE (Pillar Feature Extraction)network, they are scattered back to the original coordinate positions.
- These features along with the coordinates form voxel input for the detection part of model.

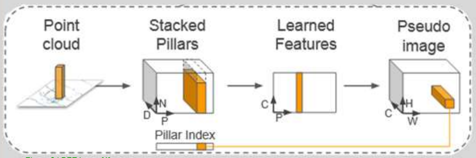


Figure 2 | PFE Layer [1]

2. Region Proposal Network (RPN):

- RPN (Region Proposal Network) is composed of 3 types of blocks. The first block type is the convolution block, which produces a smaller resolution output feature map than the input feature map.
- The second one is a transposed convolution block, which performs up-sampling from the input feature map.
- The last one is concatenation, which concatenates all the output feature maps from the transposed convolution blocks.

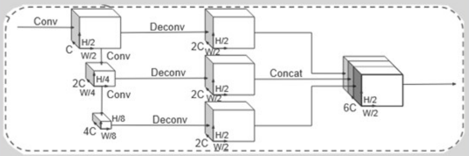


Figure 3 | RPN Layer [1]

3. Detection Head:

- An SSD (Single Stage Detector) is used to detect bounding boxes and classifying them. Each model has a distinct SSD, using different algorithms and serving different purposes.
- NMS (Non-Maximum Suppression) algorithm is applied after the detection stage. The most reliable bounding box is selected regarding the IoU (Intersection of Union) metric of bounding boxes.

Detection and Classification

Figure 4 | Input Image

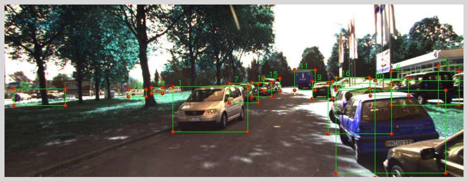


Figure 5 | Detected objects in image

Experiments

- Models were tested on 7481 image and point cloud pairs with available 3D bounding box annotations for three categories: Car, Pedestrian and Cyclist. Training and testing timings were measured along with their performance over various categories for all 6 models.
- The detection performance is evaluated using Average Precision (AP) in moderate level for the 3D object detection task with Intersection over Union (IoU) thresholds of 0.7 for Car and 0.5 for Pedestrian and Cyclist.
- KITTI PointPillars configuration from the OpenPCDet codebase were used to construct and train the model.[2]
- The experiments were performed over the Pitzer cluster, on NVIDIA Volta V100 16GB GPUs . All the dependencies were installed in a CUDA 10.2 conda environment, using PyTorch.

Results

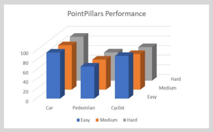


Figure 6 | Pointpillars performance

Pointpillars Performance:

- Point Pillars performance dropped as the testing data got harder.
- On average accuracy: Car > Cyclist > Pedestrian
  92% 77% 60%

Results

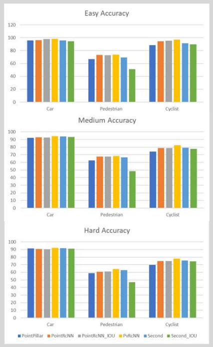


Figure 7 | Accuracy comparison

Detection Accuracy:

- Pv-RCNN model outperformed other models in all.
- SECOND_IOU model performed the worst.
- Models are detecting cars with relatively higher accuracy in all conditions.

Detection Runtime:

- SECOND model was the fastest, which detected more objects than there were in the image and had lower accuracy.
- Point Pillars was relatively faster and gave good accuracy.
- PointRCNN was the slowest but the most accurate model.

Model	Runtime	Avg no. of objects detected in each image
SECOND	01:32 mins	14
Point Pillar	01:38 mins	16
SECOND-IOU	01:52 mins	14
Pv-RCNN	04:09 mins	8
PointRcnn-IOU	04:31 mins	7
PointRcnn	04:33 mins	6

Table 1 | Inference time comparison

Conclusion

- Pv-RCNN model has the best accuracy.
- SECOND model has the fastest inference time.
- Inference Time can be further improved with ONNX and TorchScript, by extracting the trained model skeleton from python with ONNX and running it on a C++/Java framework.

References

1. Lang, Alex H., et al. "Pointpillars: Fast encoders for object detection from point clouds." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019.
2. Team, Openpcdet Development. 'OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds'. N.p., 2020. Web.