3D OBJECT DETECTION(Waymo) TEAM 12

Jay Jatinkumar Patel (015216357)
Rutvik Girishkumar Patel (014557530)

CONTENT

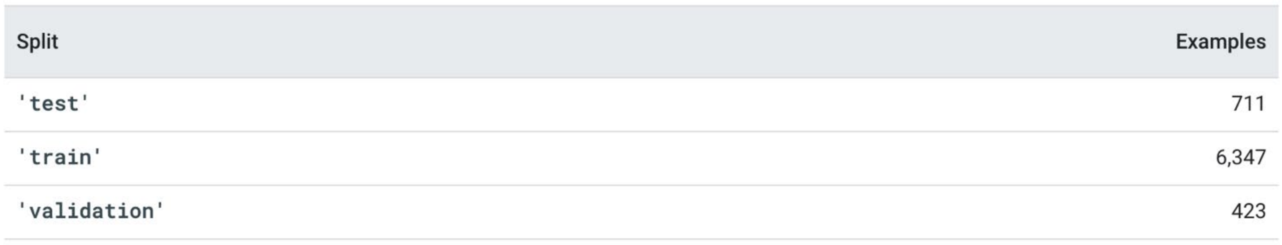
INTRODUCTION

3D object detection from pictures is one of the fundamental and challenging problems with autonomous driving, and it has drawn growing interest from both business and academia in recent years. The rapid advancement of deep learning technology has benefited image-based 3D detection, which has made significant strides. In this area, numerous academics are working to advance autonomous systems. Autonomous systems rely on their perception of their surroundings to be safe and make choices about how to interact with their environment. Object detection, which serves as the autonomous systems' eyesight, enables the ability to distinguish and locate a variety of difficulties. Applications include real-time ball tracking for sports, crop monitoring, and pedestrian detection for self-driving cars, to name a few. I have used MMDetection and PointPillar for 3D detection. We have eight cameras installed all around Tesla vehicles to acquire numerous photos for object detection. Even when a camera is effective and useful, it still struggles to effectively detect vehicles at a distance. The critics of quick lane changes disagree that a camera can provide a full representation of reality when factors like weather are taken into account. We employ LIDAR, a sensor that uses light to detect things and map them as a 3D environment, to take these factors into account. LiDAR systems plot points on a digital map in real-time using light pulses. Self-driving automobiles or autonomous vehicles can utilize this information to move safely and prevent collisions with their surroundings. An advantage of using LiDAR is being able to measure objects and their distance. Safety, which is a key worry with self-driving cars, may be increased by using LiDAR systems with high precision and reliability. they are superior than vision-based systems in their ability to capture the surroundings in great detail. In order to recognize 3D objects in a 3D environment, a new method must be used. It includes identifying and figuring out 3D details about user-selected 3D objects in a picture or range scan, such their stance, volume, or shape. To employ, we require identification algorithms to carry out the detection and a dataset to use for the detection.

Dataset

Kitti Dataset

The dataset which I used initially for the 3D Detection in this task is Kitti Dataset. Kitti contains a suite of vision tasks built using an autonomous driving platform. The full benchmark contains many tasks such as stereo, optical flow, visual odometry, etc. This dataset comprises the object detection dataset, including the monocular images and bounding boxes. The dataset contains 7481 training images annotated with 3D bounding boxes. A full description of the annotations can be found in the readme of the object development kit readme on the Kitti homepage. The dataset is been split into train, val and test as show in the fig 1. And the fig. 2 is the example image of KITTI datset.



Split	Examples
'test'	711
'train'	6,347
'validation'	423

Fig. 1 Split of KITTI Dataset

Fig. 2 Examples of KITTI Dataset

To train our model and test for identification, we need a dataset. The benchmark for 3D object identification includes of 7481 training photos, 7518 test images, and the associated point clouds, totaling 80.256 identified items. We generate precision-recall curves for assessment. We calculate the average precision to order the approaches. All methods must utilize the same set of parameters for each test pair. [1] There is a file with the same name but several frame-specific extensions. The image files are common png files that may be viewed by any program that supports PNG. The label files in the text include the bounding box for objects in 2D and 3D. Each file row is an object with 15 values, one of which is the tag (e.g. Car, Pedestrian, Cyclist). The 2D bounding boxes are defined by the pixels in the camera picture. The three-dimensional bounding boxes are in two-dimensional coordinates. The size (height, weight, and length) is contained in the object coordinate, whereas the center of the bounding box is contained in the camera coordinate. The point cloud file comprises a point's position and reflectance in lidar coordinates. The calibration file includes the values for six matrices. The Px matrices project to the camera x image a point in the corrected referenced camera coordinate. The reference camera coordinate is camera 0.

Waymo Dataset (In Kitti Format)

The Waymo dataset which is provided by Prof. Kaikai Liu in the HPC is in The Kitti format. This is a perception data collected by Wayno. The dataset consists of the training data in both png and Lidar data format. It also consists of the ground truth data. The Blue boxes represent ground truth data. The green boxes represents predicted objects. The Lidar data is 360 degrees. Below is the image frame and Lidar data visualization of Waymo dataset.

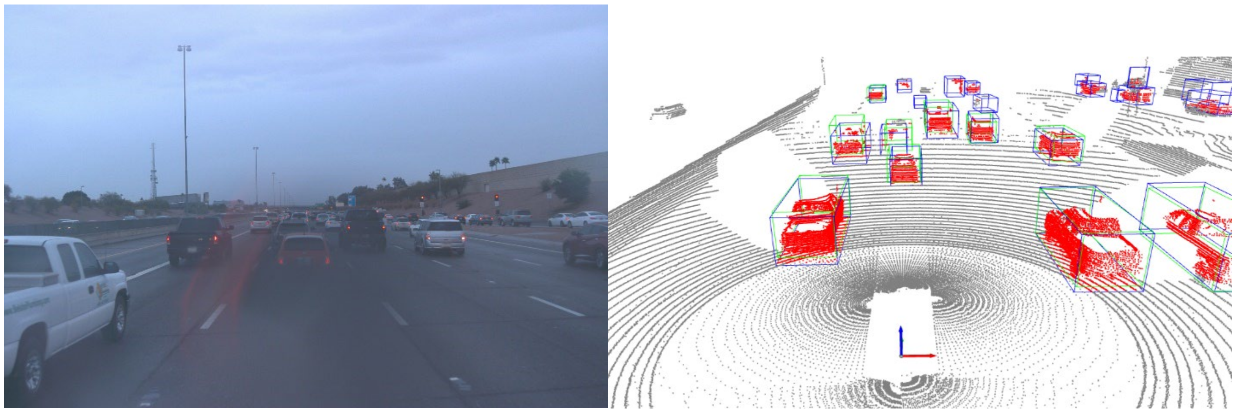


Fig. 2 Examples of Waymo Dataset

Setup & Installation

Path in HPC : /home/015216357

Used the model and code from : https://github.com/open-mmlab/mmdetection3d

requirements :

mmcv-full>=1.4.8,<=1.6.0

mmdet>=2.24.0,<=3.0.0

mmsegmentation>=0.20.0,<=1.0.0

install mmdetection3d :

1. pip install mmcv-full -f https://download.openmmlab.com/mmcv/dist/cu101/torch1.6.0/index.html
2. pip install mmdet
3. pip install mmsegmentation
4. git clone https://github.com/open-mmlab/mmdetection3d.git
5. cd mmdetection3d
6. pip install -v -e . # or "python setup.py develop"

Change datapath in config file:

configs/pointpillars/hv_pointpillars_secfpn_6x8_160e_kitti-3d-3class.py | configs/pointpillars/myhv_pointpillars_secfpn_6x8_160e_waymokitti-3d-3class.py:

- data_root = '/data/cmpe249-fa22/kitti'
- data_root = '/data/cmpe249-fa22/WaymoKitti/4c_train5678/'

configs/_base_/datasets/kitti-3d-3class.py :

- data_root = '/data/cmpe249-fa22/kitti'
- data_root = '/data/cmpe249-fa22/WaymoKitti/4c_train5678/'

Change max_epoch = 25 (due to time constraints)

Train the model :

- python tools/train.py configs/pointpillars/hv_pointpillars_secfpn_6x8_160e_kitti-3d-3class.py
- python tools/train.py configs/pointpillars/myhv_pointpillars_secfpn_6x8_160e_waymokitti-3d-3class.py --work-dir ./mypointpillar_waymokitti/
-

Checkpoints will be store in default dir :

Work_dirs/ hv_pointpillars_secfpn_6x8_160e_kitti-3d-3class/

Test the trained checkpoint :

```
- python tools/test.py --out /home/015216357/results.pkl
  configs/pointpillars/hv_pointpillars_secfpn_6x8_160e_kitti-3d-3class.py
- python tools/test.py --out /home/015216357/results_waymokitti.pkl
  configs/pointpillars/myhv_pointpillars_secfpn_6x8_160e_waymokitti-3d-3class.py
  mypointpillar_waymokitti/latest.pth --eval mAP
work_dirs/hv_pointpillars_secfpn_6x8_160e_kitti-3d-3class/latest.pth --eval mAP
```

Point cloud dataset visualization:

Run Command(On local machine) :

```
python tools/misc/browse_dataset.py
configs/pointpillars/hv_pointpillars_secfpn_6x8_160e_kitti-3d-3class.py --task det --output-dir
results --online
```

Point cloud result visualization and inference pipeline:

Run Command(On local machine) :

```
python tools/misc/visualize_results.py
configs/pointpillars/hv_pointpillars_secfpn_6x8_160e_kitti-3d-3class.py  --result results.pkl --
show-dir ./
```

PointPillars(MMDetection3D)

PointPillars is a method for 3-D object detection using 2-D convolutional layers. PointPillars network has a learnable encoder that uses PointNets to learn a representation of point clouds organized in pillars (vertical columns). The network then runs a 2-D convolutional neural network (CNN) to produce network predictions, decodes the predictions, and generates 3-D bounding boxes for different object classes such as cars, trucks, and pedestrians.

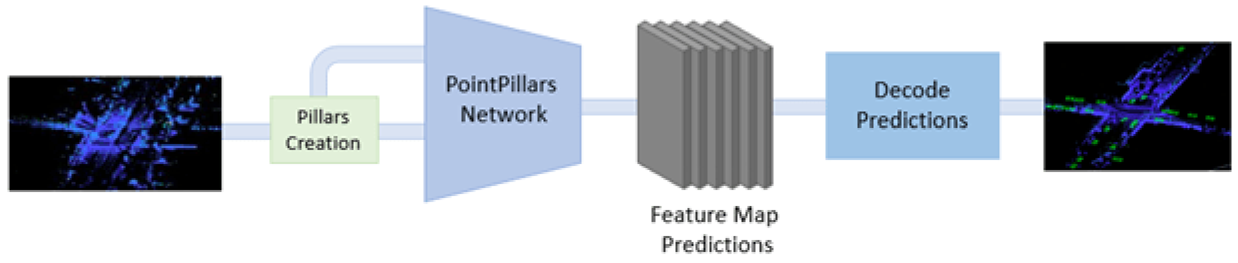


Fig 3. PointPillar

The PointPillars network has these main stages.

1. Use a feature encoder to convert a point cloud to a sparse pseudoimage.

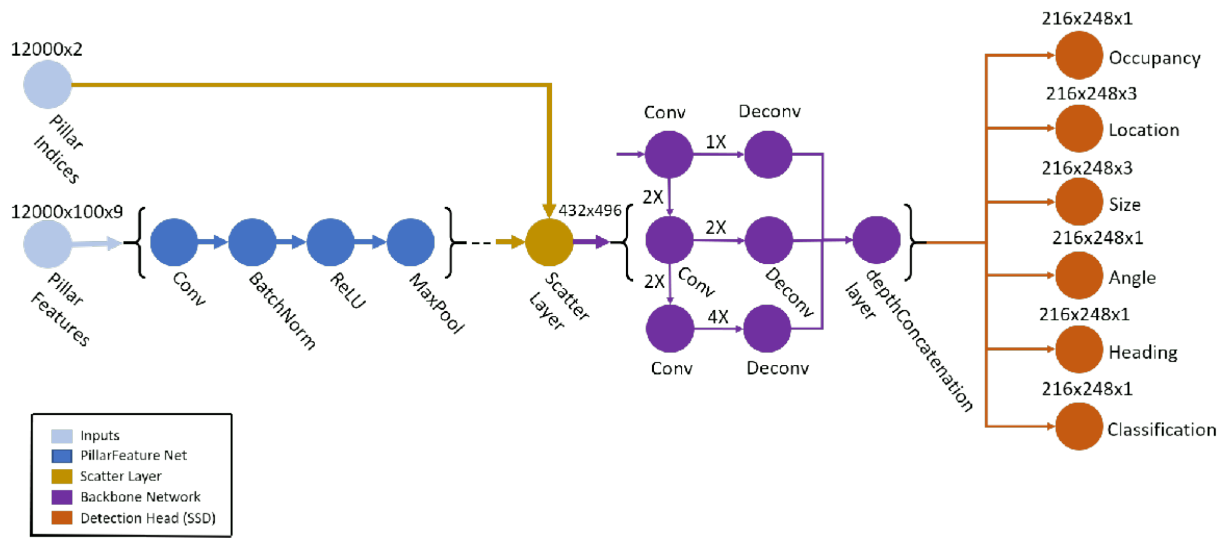
2. Process the pseudoimage into a high-level representation using a 2-D convolution backbone.
3. Detect and regress 3D bounding boxes using detection heads.

A PointPillars network requires two inputs: pillar indices as a *P*-by-2 and pillar features as a *P*-by-*N*-by-*K* matrix. *P* is the number of pillars in the network, *N* is the number of points per pillar, and *K* is the feature dimension.

The network begins with a feature encoder, which is a simplified PointNet. It contains a series of convolution, batch-norm, and relu layers followed by a max pooling layer. A scatter layer at the end maps the extracted features into a 2-D space using the pillar indices.

Next, the network has a 2-D CNN backbone that consists of encoder-decoder blocks. Each encoder block consists of convolution, batch-norm, and relu layers to extract features at different spatial resolutions. Each decoder block consists of transpose convolution, batch-norm, and relu layers.

The network then concatenates output features at the end of each decoder block, and passes these features through six detection heads with convolutional and sigmoid layers to predict occupancy, location, size, angle, heading, and class.

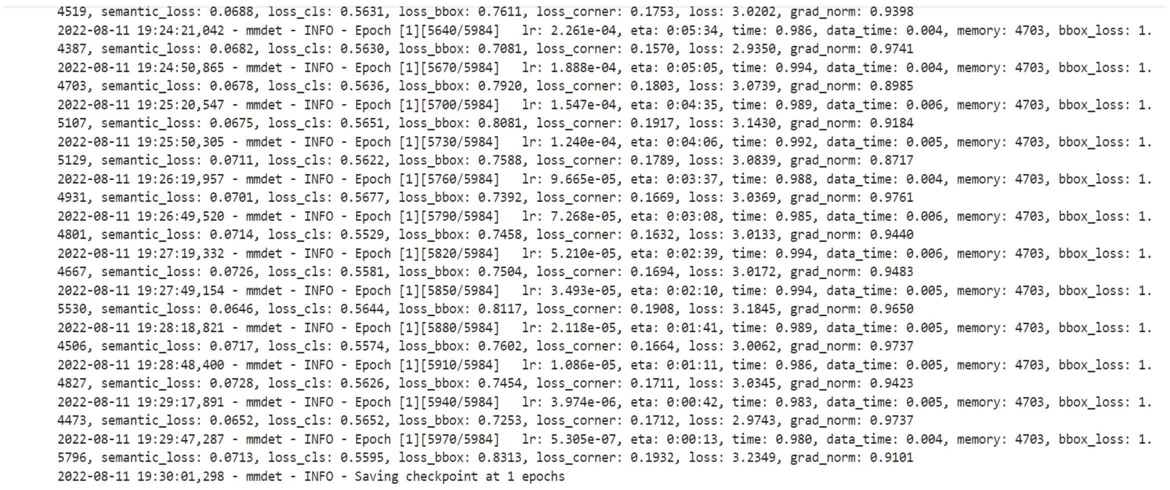


Point-rcnn

In a bottom-up approach, the suggested bin-based 3D box regression loss allows the PointRCNN 3D object detector to immediately create correct 3D box proposals from raw point cloud data. These suggestions are subsequently refined in the canonical coordinate. As far as we are aware, PointRCNN is the first two-stage 3D object detector that simply uses the raw point cloud as input.[4] It i1 a much heavier model compared to PointPillars and therefore takes much more time to train. Nearly 1 day to complete 10 epochs of training below are the screenshots of training and evaluation on the model. To note, we have many iterations happening per epoch which takes much more time and computation.

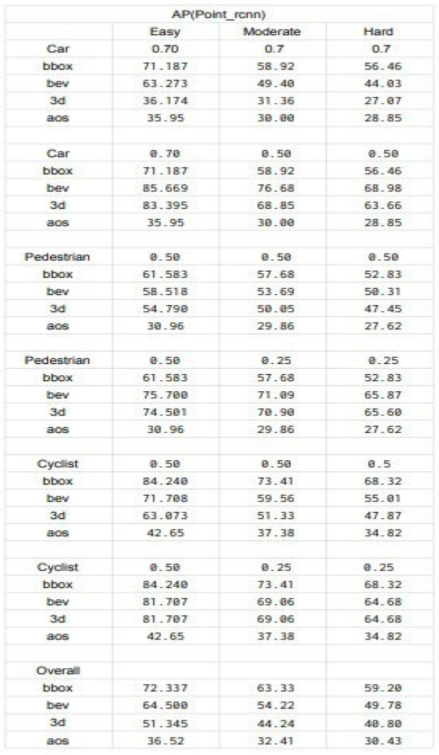
Point-rcnn VS Pointpillars (Kitti Dataset)

Point-rcnn:



```
4519, semantic_loss: 0.0688, loss_cls: 0.5631, loss_bbox: 0.7611, loss_corner: 0.1753, loss: 3.0202, grad_norm: 0.9398
2022-08-11 19:24:21,042 - mmdet - INFO - Epoch [1][5640/5984]   lr: 2.261e-04, eta: 0:05:34, time: 0.986, data_time: 0.004, memory: 4703, bbox_loss: 1.
4387, semantic_loss: 0.0682, loss_cls: 0.5630, loss_bbox: 0.7081, loss_corner: 0.1570, loss: 2.9350, grad_norm: 0.9741
2022-08-11 19:24:50,865 - mmdet - INFO - Epoch [1][5670/5984]   lr: 1.888e-04, eta: 0:05:05, time: 0.994, data_time: 0.004, memory: 4703, bbox_loss: 1.
4703, semantic_loss: 0.0678, loss_cls: 0.5636, loss_bbox: 0.7920, loss_corner: 0.1803, loss: 3.0739, grad_norm: 0.8985
2022-08-11 19:25:20,547 - mmdet - INFO - Epoch [1][5700/5984]   lr: 1.547e-04, eta: 0:04:35, time: 0.989, data_time: 0.006, memory: 4703, bbox_loss: 1.
5107, semantic_loss: 0.0675, loss_cls: 0.5651, loss_bbox: 0.8081, loss_corner: 0.1917, loss: 3.1430, grad_norm: 0.9184
2022-08-11 19:25:50,305 - mmdet - INFO - Epoch [1][5730/5984]   lr: 1.240e-04, eta: 0:04:06, time: 0.992, data_time: 0.005, memory: 4703, bbox_loss: 1.
5129, semantic_loss: 0.0711, loss_cls: 0.5622, loss_bbox: 0.7588, loss_corner: 0.1789, loss: 3.0839, grad_norm: 0.8717
2022-08-11 19:26:19,957 - mmdet - INFO - Epoch [1][5760/5984]   lr: 9.665e-05, eta: 0:03:37, time: 0.988, data_time: 0.004, memory: 4703, bbox_loss: 1.
4931, semantic_loss: 0.0701, loss_cls: 0.5677, loss_bbox: 0.7392, loss_corner: 0.1669, loss: 3.0369, grad_norm: 0.9761
2022-08-11 19:26:49,520 - mmdet - INFO - Epoch [1][5790/5984]   lr: 7.268e-05, eta: 0:03:08, time: 0.985, data_time: 0.006, memory: 4703, bbox_loss: 1.
4801, semantic_loss: 0.0714, loss_cls: 0.5529, loss_bbox: 0.7458, loss_corner: 0.1632, loss: 3.0133, grad_norm: 0.9440
2022-08-11 19:27:19,332 - mmdet - INFO - Epoch [1][5820/5984]   lr: 5.210e-05, eta: 0:02:39, time: 0.994, data_time: 0.006, memory: 4703, bbox_loss: 1.
4667, semantic_loss: 0.0726, loss_cls: 0.5581, loss_bbox: 0.7504, loss_corner: 0.1694, loss: 3.0172, grad_norm: 0.9483
2022-08-11 19:27:49,154 - mmdet - INFO - Epoch [1][5850/5984]   lr: 3.493e-05, eta: 0:02:10, time: 0.994, data_time: 0.005, memory: 4703, bbox_loss: 1.
5530, semantic_loss: 0.0646, loss_cls: 0.5644, loss_bbox: 0.8117, loss_corner: 0.1908, loss: 3.1845, grad_norm: 0.9650
2022-08-11 19:28:18,821 - mmdet - INFO - Epoch [1][5880/5984]   lr: 2.118e-05, eta: 0:01:41, time: 0.989, data_time: 0.005, memory: 4703, bbox_loss: 1.
4506, semantic_loss: 0.0717, loss_cls: 0.5574, loss_bbox: 0.7602, loss_corner: 0.1664, loss: 3.0062, grad_norm: 0.9737
2022-08-11 19:28:48,400 - mmdet - INFO - Epoch [1][5910/5984]   lr: 1.086e-05, eta: 0:01:11, time: 0.986, data_time: 0.005, memory: 4703, bbox_loss: 1.
4827, semantic_loss: 0.0728, loss_cls: 0.5626, loss_bbox: 0.7454, loss_corner: 0.1711, loss: 3.0345, grad_norm: 0.9423
2022-08-11 19:29:17,891 - mmdet - INFO - Epoch [1][5940/5984]   lr: 3.974e-06, eta: 0:00:42, time: 0.983, data_time: 0.005, memory: 4703, bbox_loss: 1.
4473, semantic_loss: 0.0652, loss_cls: 0.5652, loss_bbox: 0.7253, loss_corner: 0.1712, loss: 2.9743, grad_norm: 0.9737
2022-08-11 19:29:47,287 - mmdet - INFO - Epoch [1][5970/5984]   lr: 5.305e-07, eta: 0:00:13, time: 0.980, data_time: 0.004, memory: 4703, bbox_loss: 1.
5796, semantic_loss: 0.0713, loss_cls: 0.5595, loss_bbox: 0.8313, loss_corner: 0.1932, loss: 3.2349, grad_norm: 0.9101
2022-08-11 19:30:01,298 - mmdet - INFO - Saving checkpoint at 1 epochs
```

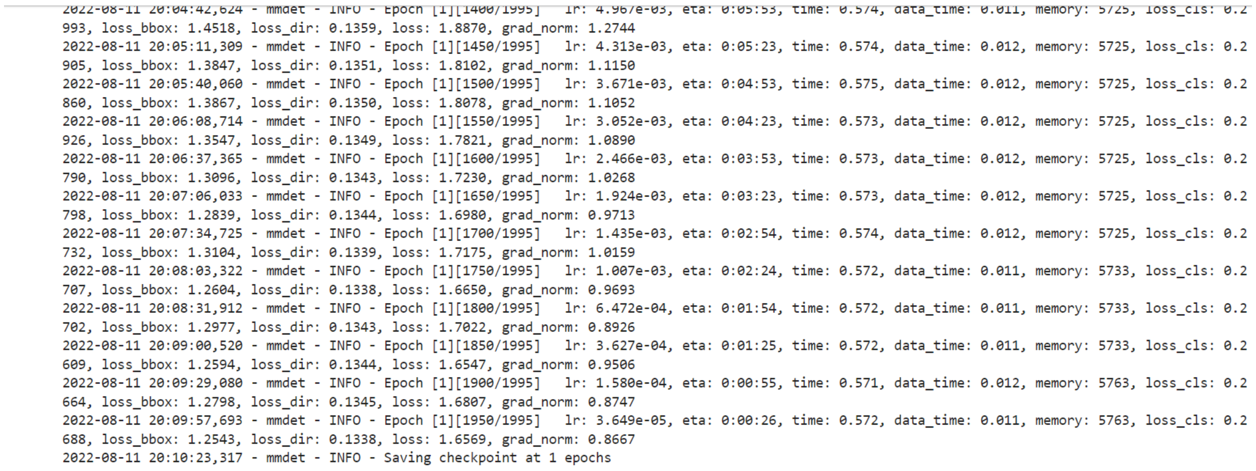
Fig 4. Point-rcnn Training 1 Epoch



AP(Point_rcnn)			
	Easy	Moderate	Hard
Car	0.70	0.7	0.7
bbox	71.187	58.92	56.46
bev	63.273	49.40	44.03
3d	36.174	31.36	27.07
aos	35.95	30.00	28.85
Car	0.70	0.50	0.50
bbox	71.187	58.92	56.46
bev	85.669	76.68	68.98
3d	83.395	68.85	63.66
aos	35.95	30.00	28.85
Pedestrian	0.50	0.50	0.50
bbox	61.583	57.68	52.83
bev	58.518	53.69	50.31
3d	54.790	50.05	47.45
aos	30.96	29.86	27.62
Pedestrian	0.50	0.25	0.25
bbox	61.583	57.68	52.83
bev	75.700	71.09	65.87
3d	74.501	70.90	65.60
aos	30.96	29.86	27.62
Cyclist	0.50	0.50	0.5
bbox	84.240	73.41	68.32
bev	71.708	59.56	55.01
3d	63.073	51.33	47.87
aos	42.65	37.38	34.82
Cyclist	0.50	0.25	0.25
bbox	84.240	73.41	68.32
bev	81.707	69.06	64.68
3d	81.707	69.06	64.68
aos	42.65	37.38	34.82
Overall			
bbox	72.337	63.33	59.20
bev	64.500	54.22	49.78
3d	51.345	44.24	40.80
aos	36.52	32.41	30.43

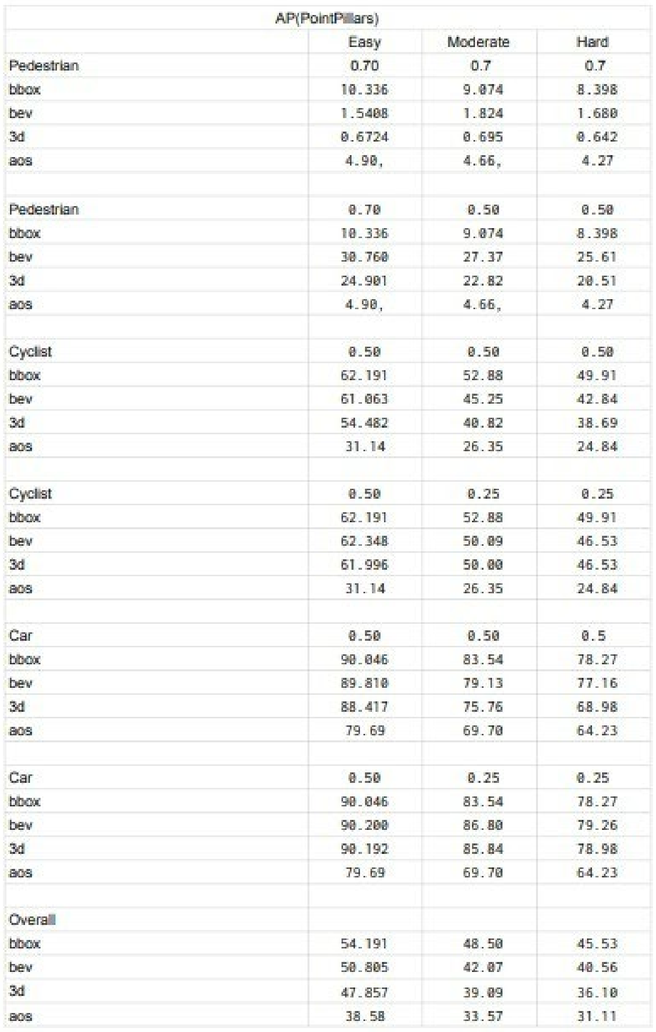
Fig 5. Point-rcnn Results (Kitti eval) 1 Epoch

Point Pillars:



```
2022-08-11 20:04:42,624 - mmdet - INFO - Epoch [1][1400/1995]   lr: 4.967e-03, eta: 0:05:53, time: 0.574, data_time: 0.011, memory: 5725, loss_cls: 0.2
993, loss_bbox: 1.4518, loss_dir: 0.1359, loss: 1.8870, grad_norm: 1.2744
2022-08-11 20:05:11,309 - mmdet - INFO - Epoch [1][1450/1995]   lr: 4.313e-03, eta: 0:05:23, time: 0.574, data_time: 0.012, memory: 5725, loss_cls: 0.2
905, loss_bbox: 1.3847, loss_dir: 0.1351, loss: 1.8102, grad_norm: 1.1150
2022-08-11 20:05:40,060 - mmdet - INFO - Epoch [1][1500/1995]   lr: 3.671e-03, eta: 0:04:53, time: 0.575, data_time: 0.012, memory: 5725, loss_cls: 0.2
860, loss_bbox: 1.3867, loss_dir: 0.1350, loss: 1.8078, grad_norm: 1.1052
2022-08-11 20:06:08,714 - mmdet - INFO - Epoch [1][1550/1995]   lr: 3.052e-03, eta: 0:04:23, time: 0.573, data_time: 0.012, memory: 5725, loss_cls: 0.2
926, loss_bbox: 1.3547, loss_dir: 0.1349, loss: 1.7821, grad_norm: 1.0890
2022-08-11 20:06:37,365 - mmdet - INFO - Epoch [1][1600/1995]   lr: 2.466e-03, eta: 0:03:53, time: 0.573, data_time: 0.012, memory: 5725, loss_cls: 0.2
790, loss_bbox: 1.3096, loss_dir: 0.1343, loss: 1.7230, grad_norm: 1.0268
2022-08-11 20:07:06,033 - mmdet - INFO - Epoch [1][1650/1995]   lr: 1.924e-03, eta: 0:03:23, time: 0.573, data_time: 0.012, memory: 5725, loss_cls: 0.2
798, loss_bbox: 1.2839, loss_dir: 0.1344, loss: 1.6980, grad_norm: 0.9713
2022-08-11 20:07:34,725 - mmdet - INFO - Epoch [1][1700/1995]   lr: 1.435e-03, eta: 0:02:54, time: 0.574, data_time: 0.012, memory: 5725, loss_cls: 0.2
732, loss_bbox: 1.3104, loss_dir: 0.1339, loss: 1.7175, grad_norm: 1.0159
2022-08-11 20:08:03,322 - mmdet - INFO - Epoch [1][1750/1995]   lr: 1.007e-03, eta: 0:02:24, time: 0.572, data_time: 0.011, memory: 5733, loss_cls: 0.2
707, loss_bbox: 1.2604, loss_dir: 0.1338, loss: 1.6650, grad_norm: 0.9693
2022-08-11 20:08:31,912 - mmdet - INFO - Epoch [1][1800/1995]   lr: 6.472e-04, eta: 0:01:54, time: 0.572, data_time: 0.011, memory: 5733, loss_cls: 0.2
702, loss_bbox: 1.2977, loss_dir: 0.1343, loss: 1.7022, grad_norm: 0.8926
2022-08-11 20:09:00,520 - mmdet - INFO - Epoch [1][1850/1995]   lr: 3.627e-04, eta: 0:01:25, time: 0.572, data_time: 0.011, memory: 5733, loss_cls: 0.2
609, loss_bbox: 1.2594, loss_dir: 0.1344, loss: 1.6547, grad_norm: 0.9506
2022-08-11 20:09:29,080 - mmdet - INFO - Epoch [1][1900/1995]   lr: 1.580e-04, eta: 0:00:55, time: 0.571, data_time: 0.012, memory: 5763, loss_cls: 0.2
664, loss_bbox: 1.2798, loss_dir: 0.1345, loss: 1.6807, grad_norm: 0.8747
2022-08-11 20:09:57,693 - mmdet - INFO - Epoch [1][1950/1995]   lr: 3.649e-05, eta: 0:00:26, time: 0.572, data_time: 0.011, memory: 5763, loss_cls: 0.2
688, loss_bbox: 1.2543, loss_dir: 0.1338, loss: 1.6569, grad_norm: 0.8667
2022-08-11 20:10:23,317 - mmdet - INFO - Saving checkpoint at 1 epochs
```

Fig 6. PointPillars Training 1 Epoch



AP(PointPillars)			
	Easy	Moderate	Hard
Pedestrian	0.70	0.7	0.7
bbox	10.336	9.074	8.398
bev	1.5408	1.824	1.680
3d	0.6724	0.695	0.642
aos	4.90,	4.66,	4.27
Pedestrian	0.70	0.50	0.50
bbox	10.336	9.074	8.398
bev	30.760	27.37	25.61
3d	24.901	22.82	20.51
aos	4.90,	4.66,	4.27
Cyclist	0.50	0.50	0.50
bbox	62.191	52.88	49.91
bev	61.063	45.25	42.84
3d	54.482	40.82	38.69
aos	31.14	26.35	24.84
Cyclist	0.50	0.25	0.25
bbox	62.191	52.88	49.91
bev	62.348	50.09	46.53
3d	61.996	50.00	46.53
aos	31.14	26.35	24.84
Car	0.50	0.50	0.5
bbox	90.046	83.54	78.27
bev	89.810	79.13	77.16
3d	88.417	75.76	68.98
aos	79.69	69.70	64.23
Car	0.50	0.25	0.25
bbox	90.046	83.54	78.27
bev	90.200	86.80	79.26
3d	90.192	85.84	78.98
aos	79.69	69.70	64.23
Overall			
bbox	54.191	48.50	45.53
bev	50.805	42.07	40.56
3d	47.857	39.09	36.10
aos	38.58	33.57	31.11

Fig 7. PointPillars Results (Kitti eval) 1 Epoch

Fig 8: PointPillar test input Image

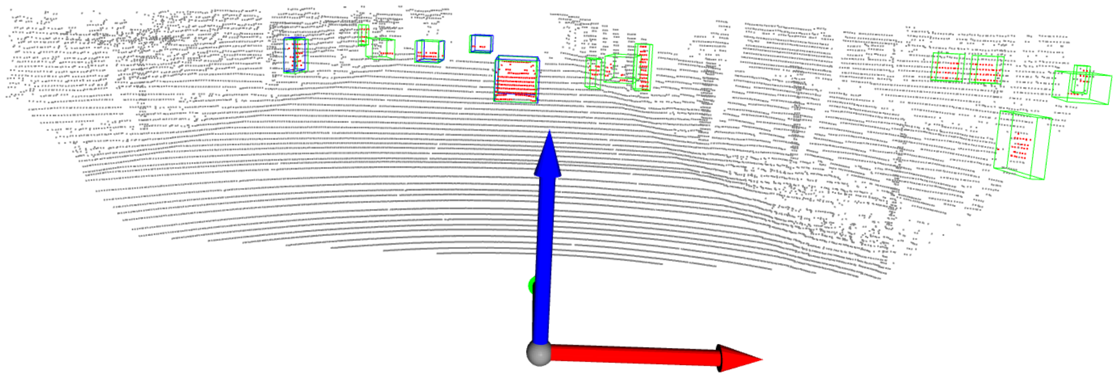
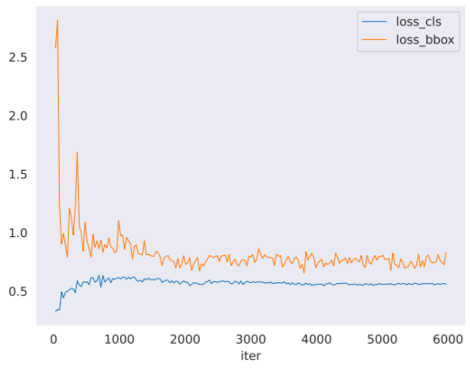
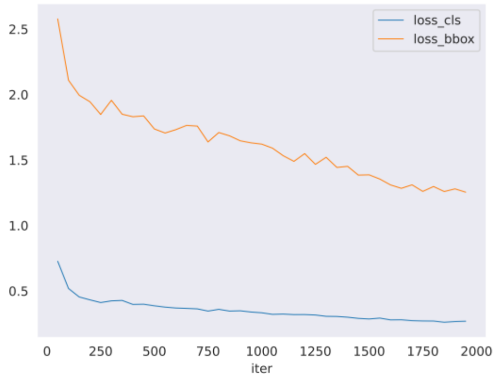


Fig 9 : PointPillar Results After Training





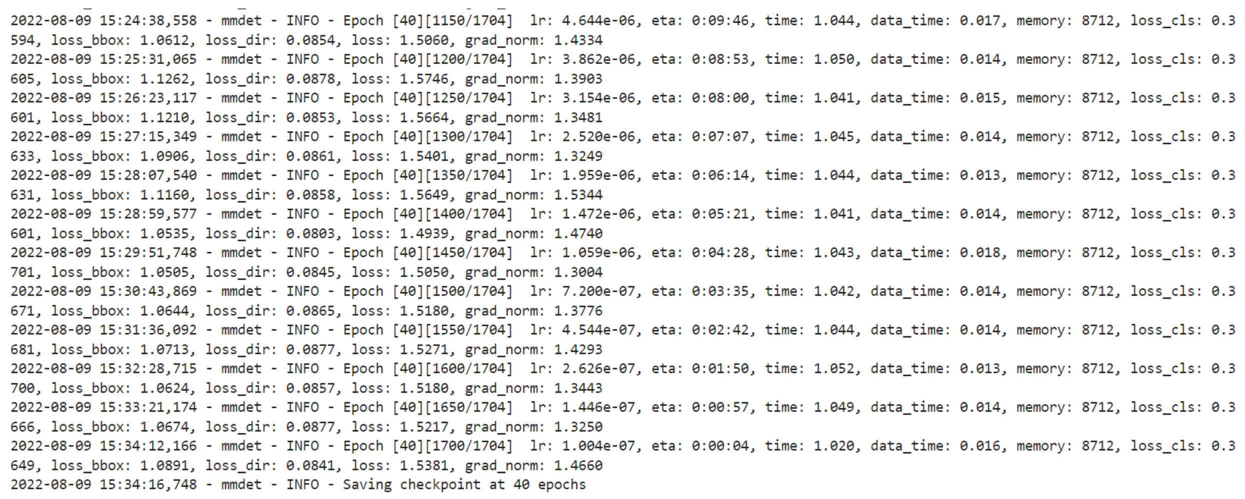
Losses for Point-rcnn and point pillars for kitti dataset

Observation:

PointPillar performs better than Point-rcnn at detecting car while point-rcnn performs better at detecting pedestrian and cyclist. Overall point-rcnn performs better at detecting objects when trained for 1 epochs. Point-rcnn takes more time for training and testing for 1 epoch. Training for more epochs might give different results.

PointPillars (Waymo-Kitti Dataset)

As we want better performance(fast and accurate detection), we decided to train pointpillars for more epoch to observe its performance on waymo-kitti dataset.

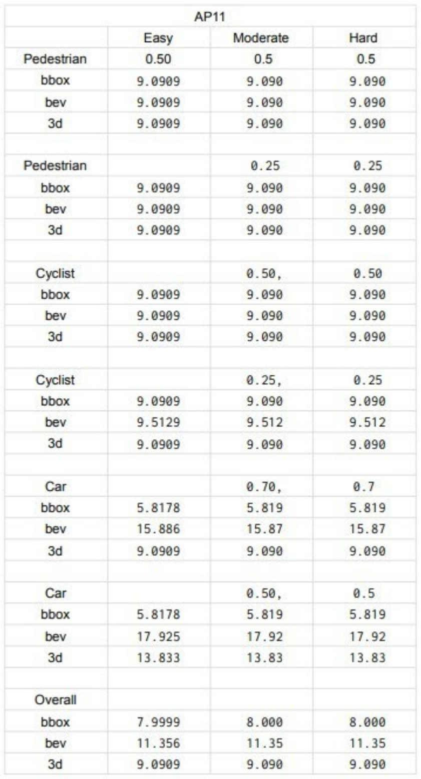


```
2022-08-09 15:24:38,558 - mmdet - INFO - Epoch [40][1150/1704]  lr: 4.644e-06, eta: 0:09:46, time: 1.044, data_time: 0.017, memory: 8712, loss_cls: 0.3
594, loss_bbox: 1.0612, loss_dir: 0.0854, loss: 1.5060, grad_norm: 1.4334
2022-08-09 15:25:31,065 - mmdet - INFO - Epoch [40][1200/1704]  lr: 3.862e-06, eta: 0:08:53, time: 1.050, data_time: 0.014, memory: 8712, loss_cls: 0.3
605, loss_bbox: 1.1262, loss_dir: 0.0878, loss: 1.5746, grad_norm: 1.3903
2022-08-09 15:26:23,117 - mmdet - INFO - Epoch [40][1250/1704]  lr: 3.154e-06, eta: 0:08:00, time: 1.041, data_time: 0.015, memory: 8712, loss_cls: 0.3
601, loss_bbox: 1.1210, loss_dir: 0.0853, loss: 1.5664, grad_norm: 1.3481
2022-08-09 15:27:15,349 - mmdet - INFO - Epoch [40][1300/1704]  lr: 2.520e-06, eta: 0:07:07, time: 1.045, data_time: 0.014, memory: 8712, loss_cls: 0.3
633, loss_bbox: 1.0906, loss_dir: 0.0861, loss: 1.5401, grad_norm: 1.3249
2022-08-09 15:28:07,540 - mmdet - INFO - Epoch [40][1350/1704]  lr: 1.959e-06, eta: 0:06:14, time: 1.044, data_time: 0.013, memory: 8712, loss_cls: 0.3
631, loss_bbox: 1.1160, loss_dir: 0.0858, loss: 1.5649, grad_norm: 1.5344
2022-08-09 15:28:59,577 - mmdet - INFO - Epoch [40][1400/1704]  lr: 1.472e-06, eta: 0:05:21, time: 1.041, data_time: 0.014, memory: 8712, loss_cls: 0.3
601, loss_bbox: 1.0535, loss_dir: 0.0803, loss: 1.4939, grad_norm: 1.4740
2022-08-09 15:29:51,748 - mmdet - INFO - Epoch [40][1450/1704]  lr: 1.059e-06, eta: 0:04:28, time: 1.043, data_time: 0.018, memory: 8712, loss_cls: 0.3
701, loss_bbox: 1.0505, loss_dir: 0.0845, loss: 1.5050, grad_norm: 1.3004
2022-08-09 15:30:43,869 - mmdet - INFO - Epoch [40][1500/1704]  lr: 7.200e-07, eta: 0:03:35, time: 1.042, data_time: 0.014, memory: 8712, loss_cls: 0.3
671, loss_bbox: 1.0644, loss_dir: 0.0865, loss: 1.5180, grad_norm: 1.3776
2022-08-09 15:31:36,092 - mmdet - INFO - Epoch [40][1550/1704]  lr: 4.544e-07, eta: 0:02:42, time: 1.044, data_time: 0.014, memory: 8712, loss_cls: 0.3
681, loss_bbox: 1.0713, loss_dir: 0.0877, loss: 1.5271, grad_norm: 1.4293
2022-08-09 15:32:28,715 - mmdet - INFO - Epoch [40][1600/1704]  lr: 2.626e-07, eta: 0:01:50, time: 1.052, data_time: 0.013, memory: 8712, loss_cls: 0.3
700, loss_bbox: 1.0624, loss_dir: 0.0857, loss: 1.5180, grad_norm: 1.3443
2022-08-09 15:33:21,174 - mmdet - INFO - Epoch [40][1650/1704]  lr: 1.446e-07, eta: 0:00:57, time: 1.049, data_time: 0.014, memory: 8712, loss_cls: 0.3
666, loss_bbox: 1.0674, loss_dir: 0.0877, loss: 1.5217, grad_norm: 1.3250
2022-08-09 15:34:12,166 - mmdet - INFO - Epoch [40][1700/1704]  lr: 1.004e-07, eta: 0:00:04, time: 1.020, data_time: 0.016, memory: 8712, loss_cls: 0.3
649, loss_bbox: 1.0891, loss_dir: 0.0841, loss: 1.5381, grad_norm: 1.4660
2022-08-09 15:34:16,748 - mmdet - INFO - Saving checkpoint at 40 epochs
```

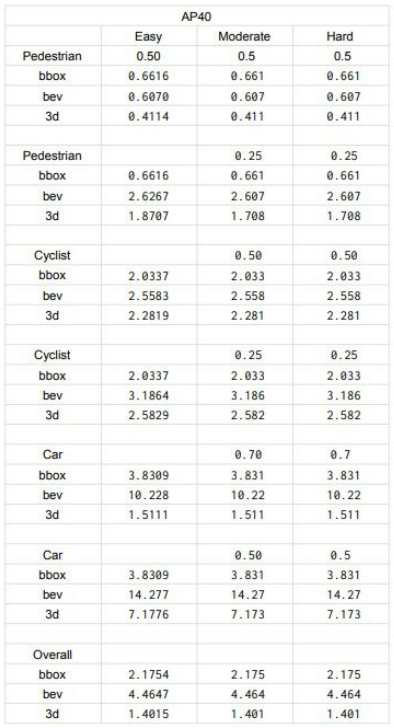
Fig 10. PointPillars Training 40 Epochs

Results

Point Pillars on WaymoKitti



AP11			
	Easy	Moderate	Hard
Pedestrian	0.50	0.5	0.5
bbox	9.0909	9.090	9.090
bev	9.0909	9.090	9.090
3d	9.0909	9.090	9.090
Pedestrian		0.25	0.25
bbox	9.0909	9.090	9.090
bev	9.0909	9.090	9.090
3d	9.0909	9.090	9.090
Cyclist		0.50,	0.50
bbox	9.0909	9.090	9.090
bev	9.0909	9.090	9.090
3d	9.0909	9.090	9.090
Cyclist		0.25,	0.25
bbox	9.0909	9.090	9.090
bev	9.5129	9.512	9.512
3d	9.0909	9.090	9.090
Car		0.70,	0.7
bbox	5.8178	5.819	5.819
bev	15.886	15.87	15.87
3d	9.0909	9.090	9.090
Car		0.50,	0.5
bbox	5.8178	5.819	5.819
bev	17.925	17.92	17.92
3d	13.833	13.83	13.83
Overall			
bbox	7.9999	8.000	8.000
bev	11.356	11.35	11.35
3d	9.0909	9.090	9.090



AP40			
	Easy	Moderate	Hard
Pedestrian	0.50	0.5	0.5
bbox	0.6616	0.661	0.661
bev	0.6070	0.607	0.607
3d	0.4114	0.411	0.411
Pedestrian		0.25	0.25
bbox	0.6616	0.661	0.661
bev	2.6267	2.607	2.607
3d	1.8707	1.708	1.708
Cyclist		0.50	0.50
bbox	2.0337	2.033	2.033
bev	2.5583	2.558	2.558
3d	2.2819	2.281	2.281
Cyclist		0.25	0.25
bbox	2.0337	2.033	2.033
bev	3.1864	3.186	3.186
3d	2.5829	2.582	2.582
Car		0.70	0.7
bbox	3.8309	3.831	3.831
bev	10.228	10.22	10.22
3d	1.5111	1.511	1.511
Car		0.50	0.5
bbox	3.8309	3.831	3.831
bev	14.277	14.27	14.27
3d	7.1776	7.173	7.173
Overall			
bbox	2.1754	2.175	2.175
bev	4.4647	4.464	4.464
3d	1.4015	1.401	1.401

Fig 11. PointPillars Results (mAP eval) 40 Epochs

The mAP scores are not much good because the Lidar range for the detection is 180 degree (Front View). The modal is performing fair for the front view but not detecting the objects on the back as we can see in the Fig.12.

Link to the video of visualization :
https://drive.google.com/file/d/1dTBL4xv15FpjYqlxlZUPx7MQd4pvYxb2/view?usp=sharing

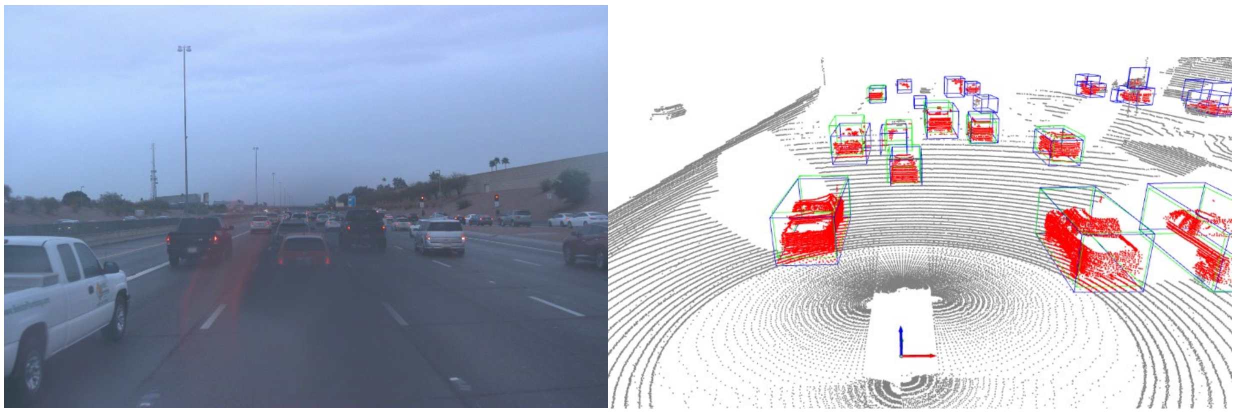


Fig. 12 Image to be inferred and the inference plot (img 184) from testing

Losses:

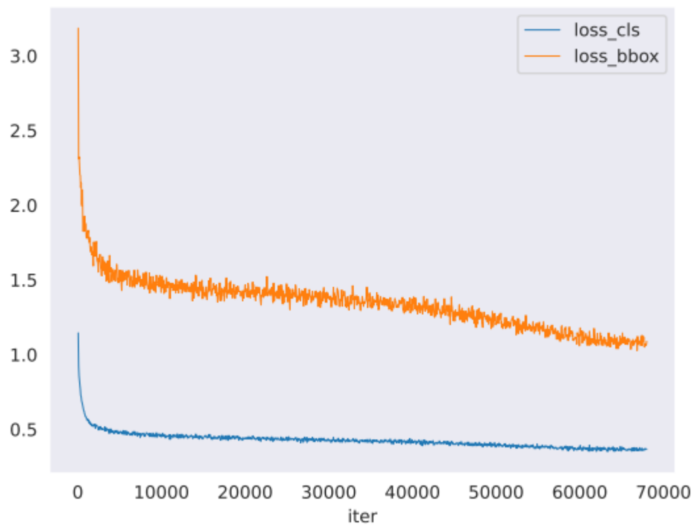
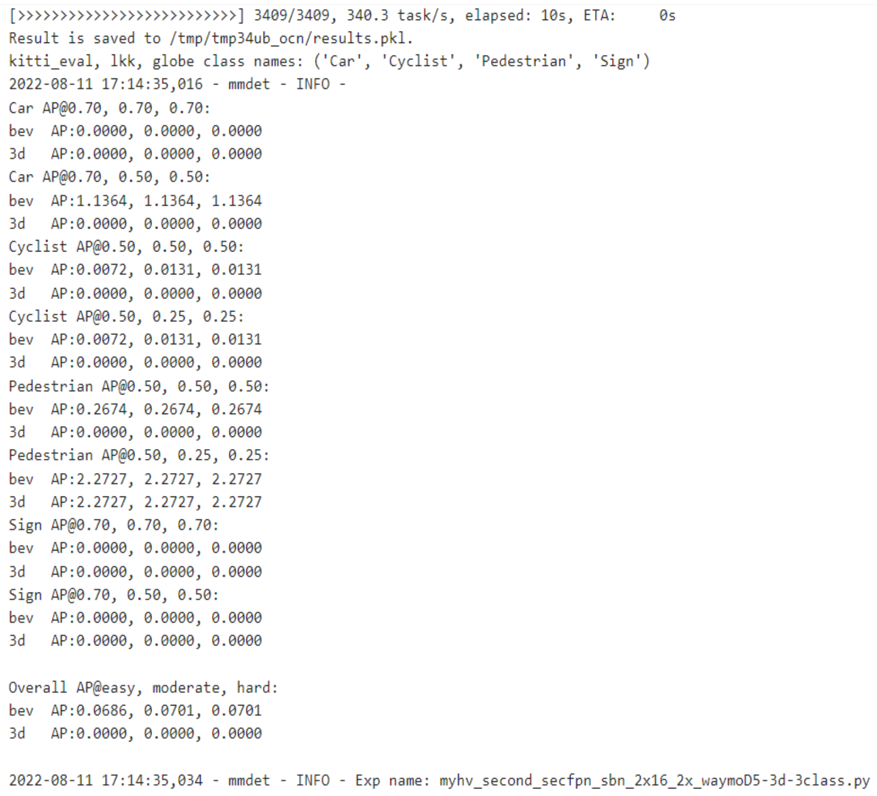


Fig 13. Loss Comparison

Here we compared the classification losses and bounding box losses.

Additional unsuccessful attempt



```
[>>>>>>>>>>>>>>>>>>>>>>>>>>>] 3409/3409, 340.3 task/s, elapsed: 10s, ETA:     0s
Result is saved to /tmp/tmp34ub_ocn/results.pkl.
kitti_eval, lkk, globe class names: ('Car', 'Cyclist', 'Pedestrian', 'Sign')
2022-08-11 17:14:35,016 - mmdet - INFO -
Car AP@0.70, 0.70, 0.70:
bev  AP:0.0000, 0.0000, 0.0000
3d   AP:0.0000, 0.0000, 0.0000
Car AP@0.70, 0.50, 0.50:
bev  AP:1.1364, 1.1364, 1.1364
3d   AP:0.0000, 0.0000, 0.0000
Cyclist AP@0.50, 0.50, 0.50:
bev  AP:0.0072, 0.0131, 0.0131
3d   AP:0.0000, 0.0000, 0.0000
Cyclist AP@0.50, 0.25, 0.25:
bev  AP:0.0072, 0.0131, 0.0131
3d   AP:0.0000, 0.0000, 0.0000
Pedestrian AP@0.50, 0.50, 0.50:
bev  AP:0.2674, 0.2674, 0.2674
3d   AP:0.0000, 0.0000, 0.0000
Pedestrian AP@0.50, 0.25, 0.25:
bev  AP:2.2727, 2.2727, 2.2727
3d   AP:2.2727, 2.2727, 2.2727
Sign AP@0.70, 0.70, 0.70:
bev  AP:0.0000, 0.0000, 0.0000
3d   AP:0.0000, 0.0000, 0.0000
Sign AP@0.70, 0.50, 0.50:
bev  AP:0.0000, 0.0000, 0.0000
3d   AP:0.0000, 0.0000, 0.0000

Overall AP@easy, moderate, hard:
bev  AP:0.0686, 0.0701, 0.0701
3d   AP:0.0000, 0.0000, 0.0000

2022-08-11 17:14:35,034 - mmdet - INFO - Exp name: myhv_second_secfpn_sbn_2x16_2x_waymoD5-3d-3class.py
```

Fig 14. Erroneous attempt of training SECOND on WaymoKitti dataset

References

1. http://www.cvlibs.net/datasets/kitti/eval object.php?obj benchmark=3d
2. https://github.com/kuixu/kitti object vis
3. Alex H Hang, PointPillars: Fast Encoders for Object Detection from Point Clouds, 2019
4. Shaoshuai Shi, Xiaogang Wang, Hongsheng Li, PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud, 2019
5. https://github.com/lkk688/mymmdetection3d.git
6. https://github.com/open-mmlab/mmdetection3d.git