PointPillars: Fast Encoders for Object Detection from Point Clouds

Date: 2022.07.13

Name: ChangMin An

Github: Link

I. Introduction

PointPillars는 downstream detection pipeline에 적합한 pillar인 point cloud encoder이다. 기존에 사용되는 encoder은 크게 두 가지로 나뉘는데, 빠르지만 정확도가 떨어지는 fixed encoder와 느리지만 정확도가 더 높은 encoders that are learned from data로 나뉜다. Pillars는 기존의 2D Conv를 사용할 수 있어 빠르고, 정확도 또한 다른 encoder에 비해 높은 장점을 가지고 있다.

PointPillars는 pillar마다 feature를 구하여 object의 3D oriented bounding box를 구하는 방법이다. PointPillars의 장점은 첫째로 fixed encoder를 통해 point cloud로부터 얻을 수 있는 모든 information을 활용할 수 있다. 두번째로 voxel대신 pillars를 사용함으로써 hyper parameter로 tuning하던 vertical direction의 binning을 할 필요가 없어졌다. 마지막으로 모든 key operation이 2D Conv로 이루어지기에 매우 빠르다.

즉 voxel 단위로 feature를 나눈 후 각 voxel마다 PointNet을 적용한다. 이후 3D Conv를 진행하여 수직축을 통합하고 그 이후 2D Conv를 진행한다.

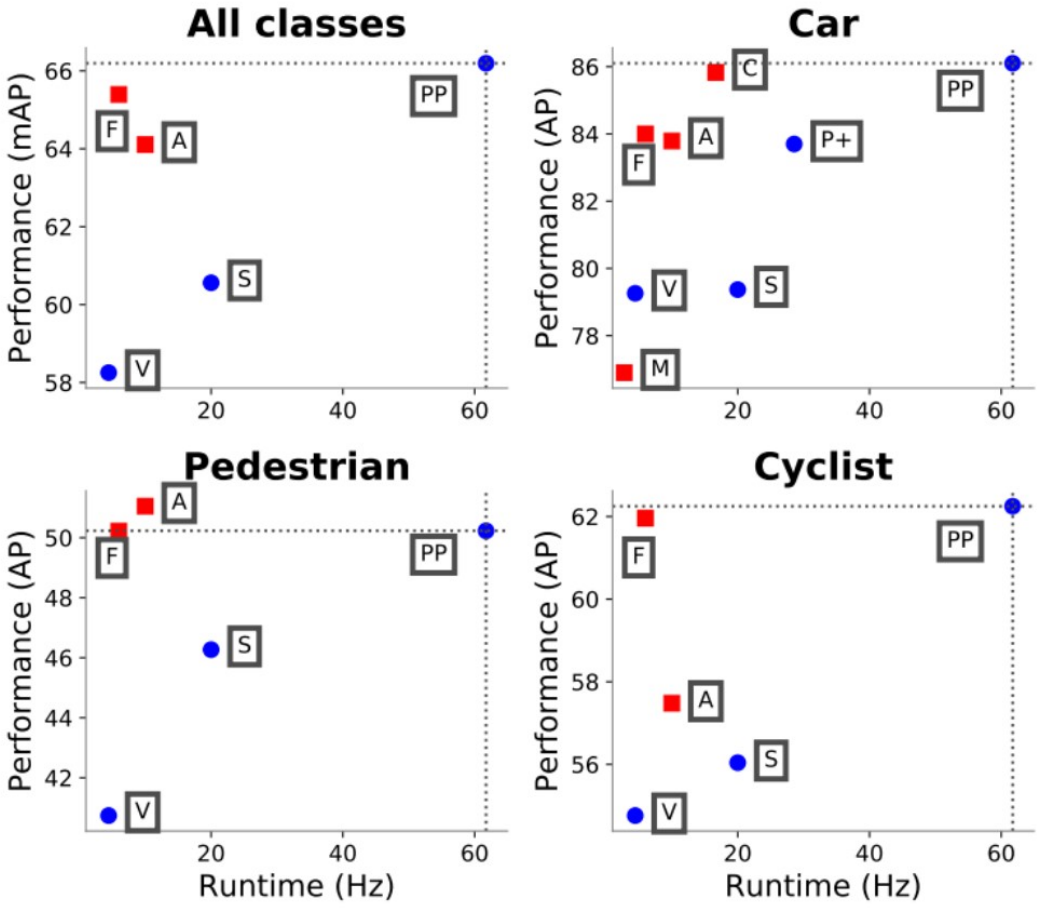

Figure 1. KITTI dataset에서 PointPillars(PP)를 비롯한 여러 알고리즘의 성능 정리(A: AVOD, M: MV3D, C: Contfuse, V: VoxelNet, F: Frustum PointNet, S: SECOND, P+: PIXOR++)

Main Contribution

- End-to-end 3D object detection에 쓰이는 새로운 point cloud encoder인 PointPillars를 제시하였다.
- 다른 3D object detection 알고리즘에 비해 2~4배 빠르게 pillars에 2D Conv를 사용하는지 보여주었다.
- KITTI Dataset에서 실험을 수행하여 3D와 BEV benchmarks에 대해 SOTA의 성능을 보여주었다.
- Ablation study를 통해 strong detection performance를 보여주는 key factor가 무엇인지 알 수 있다.

II. PointPillars Network

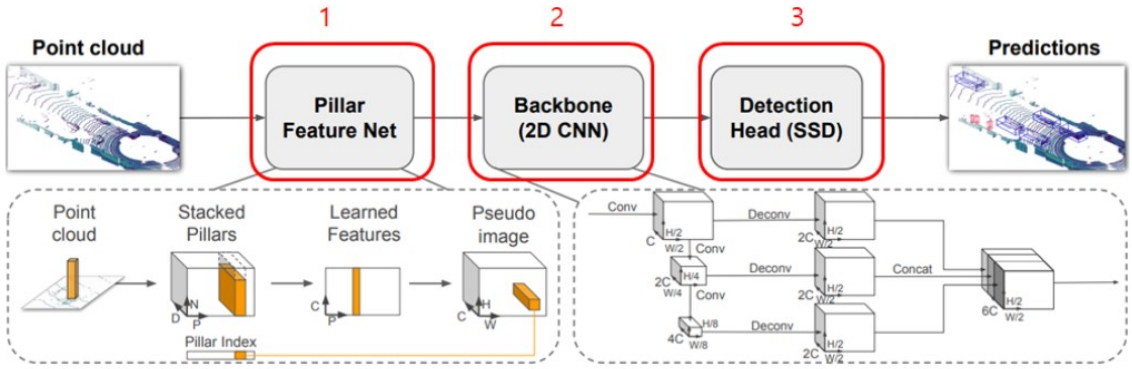

Figure 2. PointPillars Architecture

PointPillars는 크게 3가지 단계로 나뉜다.

- Point cloud를 sparse pseudo-image로 바꾸는 feature encoder network
- Pseudo-image를 high-level representation으로 바꿔주는 2D conv network
- 3D boxes를 regression하는 detection head

1. Pointcloud to Pseudo-Image

2D convolutional architecture를 구현하기 위해 point cloud를 pseudo-image 형태로 변환해야 한다. 먼저 point cloud를 x-y 평면 위의 grid로 discretized 하여 size가 B인 pillar의 set P를 만들어 준다. 이는 z 평면으로 spatial limit이 존재하지 않아 이에 대한 hyper parameter가 필요없다. 이 때 cluster center의 x, y, z값을 얻게 되고, pillar center x, y 값을 얻게 된다. 이렇게 총 9 dimension을 얻게 된다.

$$(x, y, z, r, x_C, y_C, z_C, x_P, y_P)$$

생성된 pillar는 LiDAR point가 sparse하기 때문에 대부분 비어있게 된다. 이 sparsity는 비어 있지 않은 pillar의 수(P)와 pillar당 들어있는 point 수(N)에 제한을 두어 (D, P, N)크기의 dense한 tensor를 정의하여 처리한다. 만약 data양이 많아 tensor의 크기를 초과한다면 data는 random하게 sample되어 들어간다. 너무 적은 data를 갖게 되면 zero padding을 사용하여 해결한다.

그 이후 PointNet을 사용한다. 먼저 각 point마다 linear layer, batch norm, ReLU를 적용하여 (C, P, N) 크기의 tensor를 생성한다. 그리고 각 channel에 대해 Max Pooling을 하여 (C, P) 크기의 tensor를 생성한다. Encoding 이후 결과적으로 (C, H, W)의 pseudo-image를 생성한다.

2. Backbone

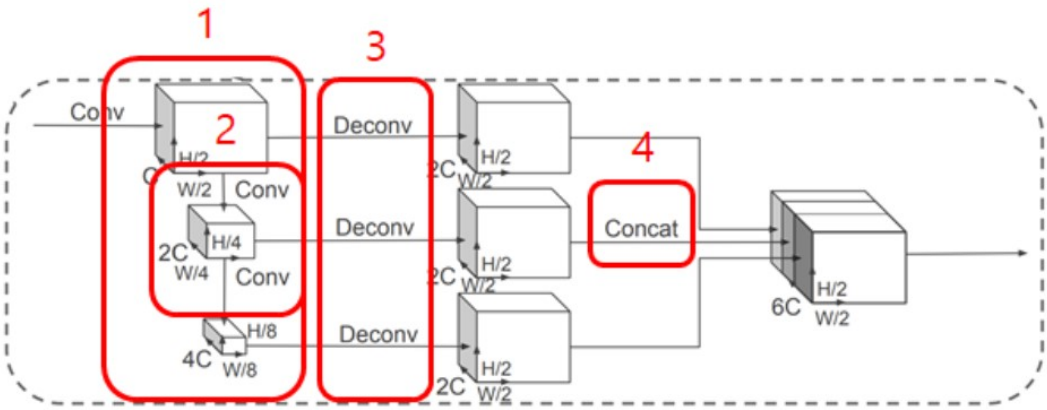

Figure 3. Backbone Network(2D conv)

PointPillars의 backbone은 VoxelNet의 3D Conv와 유사하다. Backbone network는 두 개의 sub-network를 갖는다. Top-down network는 점점 작은 spatial resolution feature를 생성하고, 두 번째 network는 이 top-down network의 feature를 up-sampling하고 concatenation한다.

3. Detection Head

Single Shot Detector(SSD) setup을 사용하여 3D object detection을 하였다. 2D와의 차이점은 height와 elevation이 추가된 regression target으로 사용되었다. 만약 다른 task를 하고 싶다면 detection head 대신 다른 head로 바꾸면 된다.

III. Implementation Details

1. Network

Initialization은 uniform distribution을 사용하여 random하게 initialize하였다. Encoder Network는 C=64개의 output feature를 가졌고, car과 pedestrian/cyclist 각각에 대해 backbone은 첫 번째 block에서의 stride(S = 2 for car, S = 1 for ped/cyc)를 제외하고는 일치한다.

각 network는 3개의 Block으로 이루어져 있으며, 각각의 block들 또한 upsampling을 한 후 concatenated되어 총 6C개의 feature가 detection head에 사용된다.

2. Loss

PointPillars에 사용된 Loss function은 SECOND에서와 같다. Ground truth와 anchors는 (x, y, z, w, h, l, theta)로 정의된다. 이 때 angle localization 만으로는 뒤집어진 box들을 구별하지 못하므로, softmax classification을 통해 heading을 학습한다.

$$\Delta x = \frac{x^{gt} - x^a}{d^a}, \Delta y = \frac{y^{gt} - y^a}{d^a}, \Delta z = \frac{z^{gt} - z^a}{h^a}$$

$$\Delta w = \log \frac{w^{gt}}{w^a}, \Delta l = \log \frac{l^{gt}}{l^a}, \Delta h = \log \frac{h^{gt}}{h^a}$$

$$\Delta \theta = \sin\left(\theta^{gt} - \theta^a\right),$$

$$\mathcal{L}_{loc} = \sum_{b \in (x,y,z,w,l,h,\theta)} \text{SmoothL1}\left(\Delta b\right)$$

IV. Experimental Setup

Dataset은 KITTI dataset을 사용하였으며, xy resolution은 0.16m로 max number of pillars per sample(P): 12000, max number of points per pillar(N): 100으로 하였다. 각 class anchor는 width, length, height, z center, 두 방향의 orientation(0, 90)으로 표현하였다. Positive anchor와 negative anchor에 대해서 loss를 적용하였고 NMS(IOU = 0.5)를 통해 box regression을 하였다.

- **Data Augmentation**

  Data augmentation은 좋은 성능을 위해서 매우 중요하다. SECOND를 따라 모든 3D box의 class에 대한 ground truth와 그 3D box들 안에 속하는 point cloud에 관한 lookup table을 만든다. 그리고 각 ground truth sample에 대해 class 별로 15, 0, 8개의 car, ped, cyc sample을 각각 뽑아서 현재 point cloud에 추가한다. 다음으로 모든 ground truth box들은 개별적으로 rotation과 translation을 통해 augmented 된다. 마지막으로 모든 point cloud와 gt box들에 대해 x축방향으로 mirroring flip을 하고 global rotation과 scaling도 적용하고 global translation도 적용한다.

V. Results

Method	Modality	Speed (Hz)	mAP	Car			Pedestrian			Cyclist		
			Mod.	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
MV3D [2]	Lidar & Img.	2.8	N/A	86.02	76.90	68.49	N/A	N/A	N/A	N/A	N/A	N/A
Cont-Fuse [15]	Lidar & Img.	16.7	N/A	88.81	85.83	77.33	N/A	N/A	N/A	N/A	N/A	N/A
Roarnet [25]	Lidar & Img.	10	N/A	88.20	79.41	70.02	N/A	N/A	N/A	N/A	N/A	N/A
AVOD-FPN [11]	Lidar & Img.	10	64.11	88.53	83.79	77.90	**58.75**	**51.05**	**47.54**	68.09	57.48	50.77
F-PointNet [21]	Lidar & Img.	5.9	65.39	88.70	84.00	75.33	58.09	50.22	47.20	75.38	61.96	54.68
HDNET [29]	Lidar & Map	20	N/A	89.14	**86.57**	78.32	N/A	N/A	N/A	N/A	N/A	N/A
PIXOR++ [29]	Lidar	35	N/A	**89.38**	83.70	77.97	N/A	N/A	N/A	N/A	N/A	N/A
VoxelNet [31]	Lidar	4.4	58.25	89.35	79.26	77.39	46.13	40.74	38.11	66.70	54.76	50.55
SECOND [28]	Lidar	20	60.56	88.07	79.37	77.95	55.10	46.27	44.76	73.67	56.04	48.78
PointPillars	Lidar	**62**	**66.19**	88.35	86.10	**79.83**	58.66	50.23	47.19	**79.14**	**62.25**	**56.00**

Table 1. Results on the KITTI test BEV detection benchmark.

Method	Modality	Speed (Hz)	mAP	Car			Pedestrian			Cyclist		
			Mod.	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
MV3D [2]	Lidar & Img.	2.8	N/A	71.09	62.35	55.12	N/A	N/A	N/A	N/A	N/A	N/A
Cont-Fuse [15]	Lidar & Img.	16.7	N/A	82.54	66.22	64.04	N/A	N/A	N/A	N/A	N/A	N/A
Roarnet [25]	Lidar & Img.	10	N/A	**83.71**	73.04	59.16	N/A	N/A	N/A	N/A	N/A	N/A
AVOD-FPN [11]	Lidar & Img.	10	55.62	81.94	71.88	66.38	50.80	42.81	40.88	64.00	52.18	46.61
F-PointNet [21]	Lidar & Img.	5.9	57.35	81.20	70.39	62.19	51.21	**44.89**	40.23	71.96	56.77	50.39
VoxelNet [31]	Lidar	4.4	49.05	77.47	65.11	57.73	39.48	33.69	31.5	61.22	48.36	44.37
SECOND [28]	Lidar	20	56.69	83.13	73.66	66.20	51.07	42.56	37.29	70.51	53.85	46.90
PointPillars	Lidar	**62**	**59.20**	79.05	**74.99**	**68.30**	**52.08**	43.53	**41.49**	**75.78**	**59.07**	**52.92**

Table 2. Results on the KITTI test 3D detection benchmark.

Method	Modality	Speed (Hz)	mAOS	Car			Pedestrian			Cyclist		
			Mod.	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
SubCNN [27]	Img.	0.5	**72.71**	**90.61**	88.43	78.63	**78.33**	**66.28**	**61.37**	71.39	63.41	56.34
AVOD-FPN [11]	Lidar & Img.	10	63.19	89.95	87.13	79.74	53.36	44.92	43.77	67.61	57.53	54.16
SECOND [28]	Lidar	20	54.53	87.84	81.31	71.95	51.56	43.51	38.78	80.97	57.20	55.14
PointPillars	Lidar	**62**	68.86	90.19	**88.76**	**86.38**	58.05	49.66	47.88	**82.43**	**68.16**	**61.96**

Table 3. Results on the KITTI test average orientation similarity (AOS) detection benchmark. SubCNN is the best performing image only method, while AVOD-FPN, SECOND, and PointPillars are the only 3D object detectors that predict orientation.

속도 측면에서나 정확도 부분에서 SOTA의 성능을 보였다. 특히 속도가 압도적으로 빨랐다.

VI. Realtime Inference

PointPillars는 quantitative analysis에서 볼 수 있듯이 running time에서 큰 improvement를 보였다. 이러한 결과에는 Encoding 단계에 원인이 있다. 기존에 VoxelNet에서 쓰이던 encoder가 190ms, SECOND에서 쓰이던 encoder가 48ms였는데, 본 논문에서는 1.3ms로 큰 향상을 보였다. 또한 2개의 연속적인 PointNet의 사용 대신 간소화된 PointNet을 사용함으로써 2.5ms를 단축하였고, 첫번째 block dimension을 64로 줄임으로써 4.5ms를 단축하였다.

LiDAR의 Hz가 20Hz로 이미 실행시간은 만족하지만, PointPillar의 경우 FOV만 사용했기에 전체적인 scene에 대해서도 빠른 실행시간을 보여야 하며, autonomous vehicle의 특성상, 더 낮은 computing power에서도 빠른 실행시간을 보여야 하는 문제가 남아있다.

VII. Ablation Studies

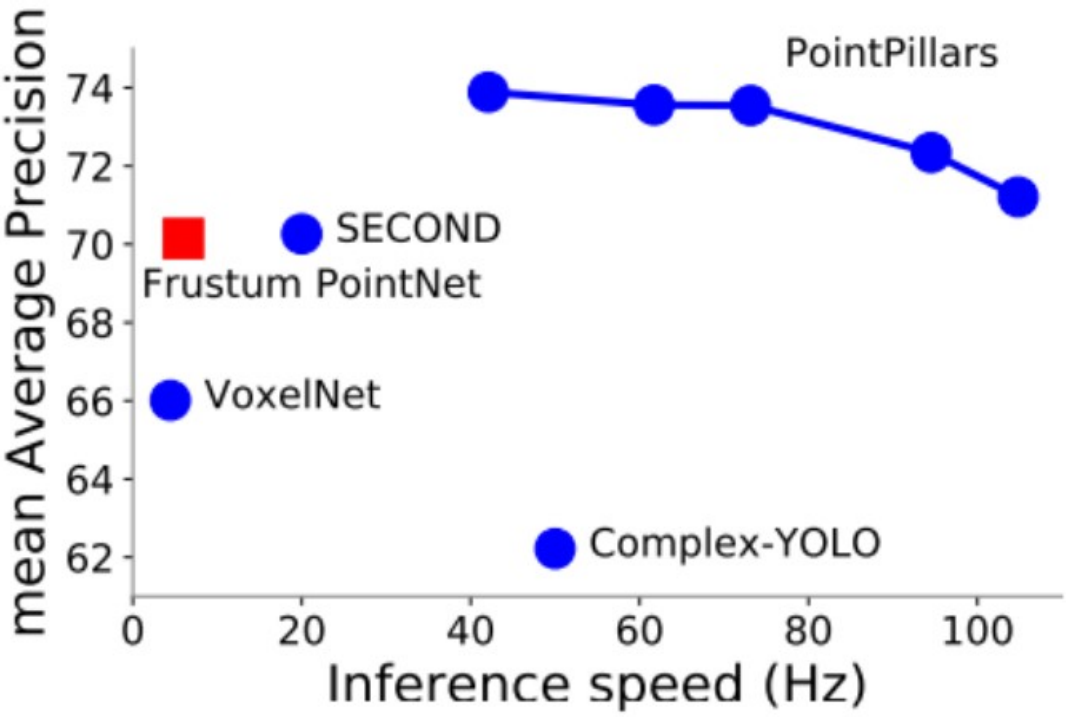

Figure. PointPillars의 다른 network들과의 비교와 speed-accuracy trade-off

1. Spatial Resolution

Spatial binning을 다양하게 하는 것은 speed와 accuracy의 trade off를 보였다. 작은 pillar는 보다 정확한 localization 성능과 많은 feature를 얻을 수 있었고, 큰 pillar는 빠른 속도를 보였다. Car class는 bin size가 달라져도 성능이 stable했는 반면, ped/cyc class는 pillar의 크기에 영향을 받았다.

2. Per Box Data Augmentation

VoxelNet과 SECOND에 대해서도 대규모의 augmentation을 적용하였다. 하지만 최소한의 box augmentation이 더 좋은 성능을 보였다. Ground truth sampling이 box augmentation의 효과를 저하시키는 것으로 보인다.

3. Point Decorations

Encoder는 raw LiDAR point의 x, y, z, r(reflection)를 받아 추가적인 channel을 생성한다. 이 channel을 생성함으로써 point들은 standardized된 local context를 갖게 된다. Cluster offset에 따라 표준화하는 방식이 다르게 되는데, 이는 point들을 통계적으로 summary하여 point간에 dependency를 생성하는 것에 이용된다. 이를 통해 data를 subsampling과 augmentation을 할 때 높은 variance를 가지게 되는 것을 막아준다.