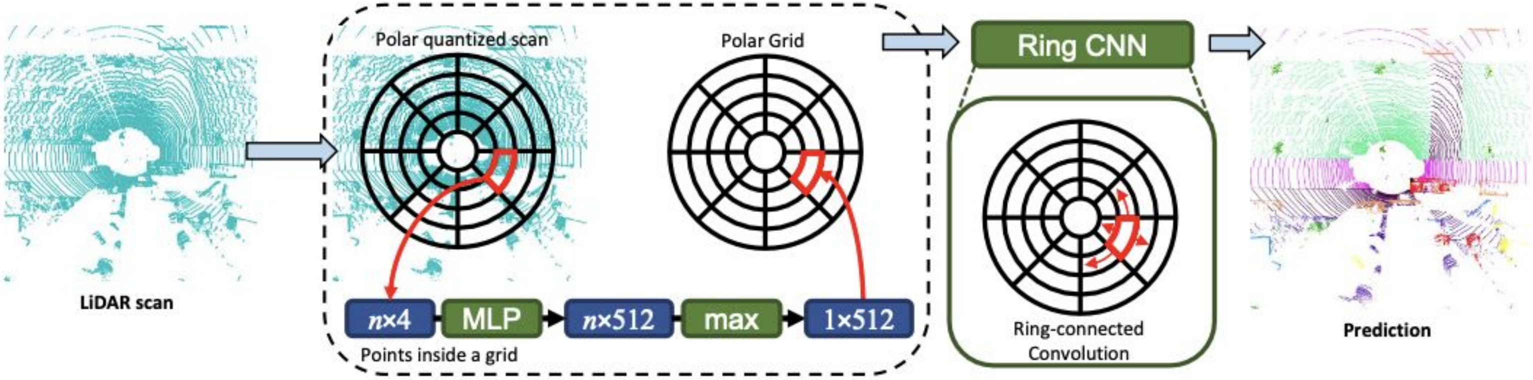
Paper Review

PolarNet: An Improved Grid Representation for Online LiDAR Point Clouds Semantic Segmentation

CVPR, 2020



Point Cloud

: Sparse and imbalanced spatial distribution

What constitutes a good input representation of one LiDAR point cloud scan?

- Perception field

  : How much context a neural network can “perceive” before it classifies a pixel to semantic class

- 2D : dilated convolution, feature pyramid
- **3D** : Not only the size but also the **shape** of the perception field

Bird eye view

: Top-down projection with Cartesian Coordinate

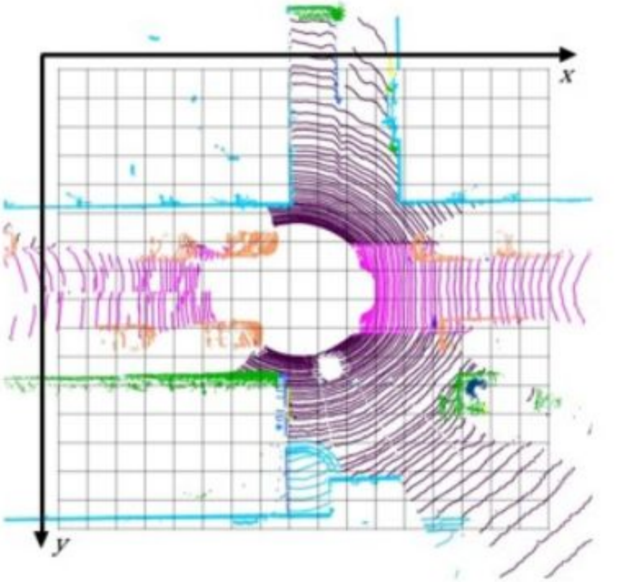
- without losing any scale and range information
- Points are organized in **rings of various radii**
- Points into the grid cells in a **nonuniform manner**

\- Cells close to the sensor : condense points

➡ blurring out fine-details of the points

\- Cells far away from the sensor : sparse points

➡ limited information & Computational waste



(a) Cartesian BEV

Polar Coordinate

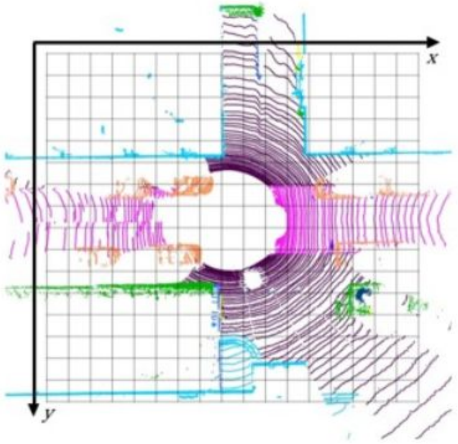
: Calculate each point’s azimuth and radius on the XY plane

1. Evenly distributes the points

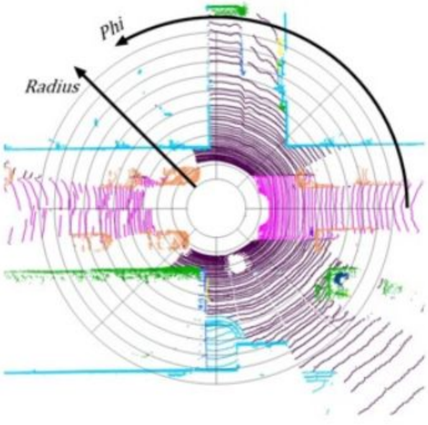
2. More balanced point distribution lessens the burden on predictors

- minor points’ predictions will be suppressed by the majority in the output

- Cartesian : 98.75%, Polar : 99.3% of points in every grid cell share the same label



(a) Cartesian BEV



(b) Polar BEV

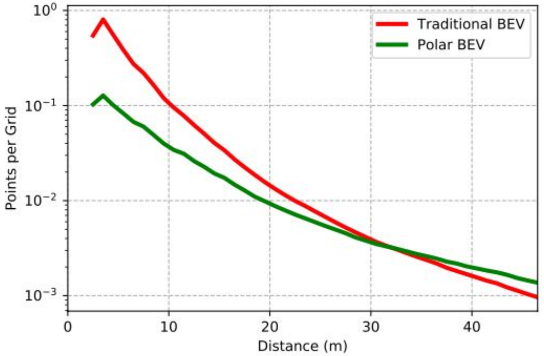
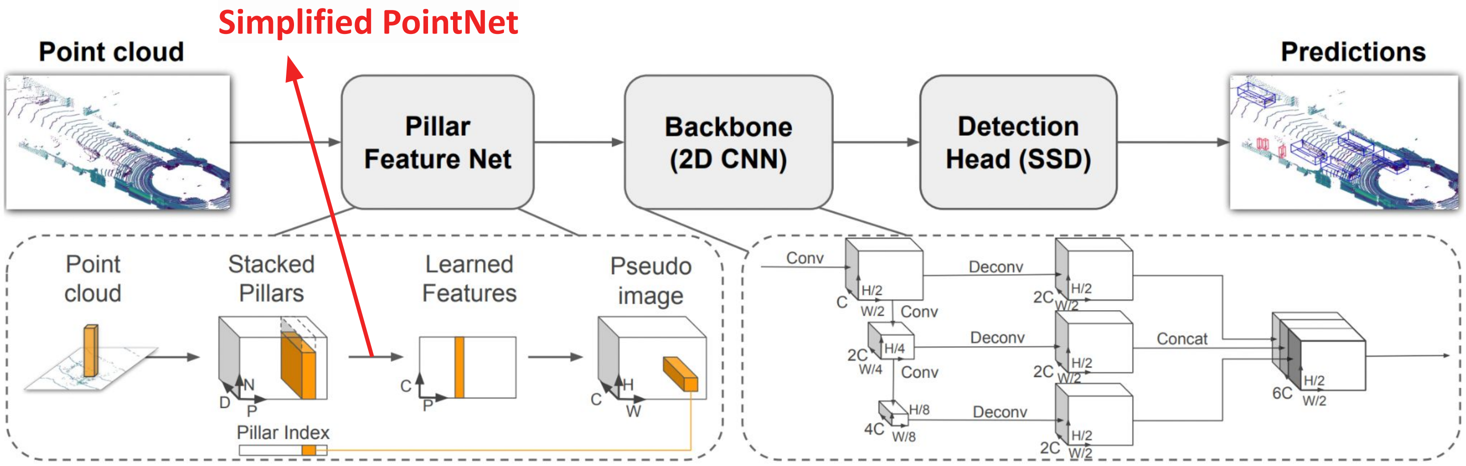


Figure 4. Grid cell distance from the sensor vs. logarithmically spaced mean number of points per grid cell. The traditional BEV representation allocates most of its grid cells to the further end with few points in them.

Segmentation

: Learned representation represents the entire vertical column of a grid

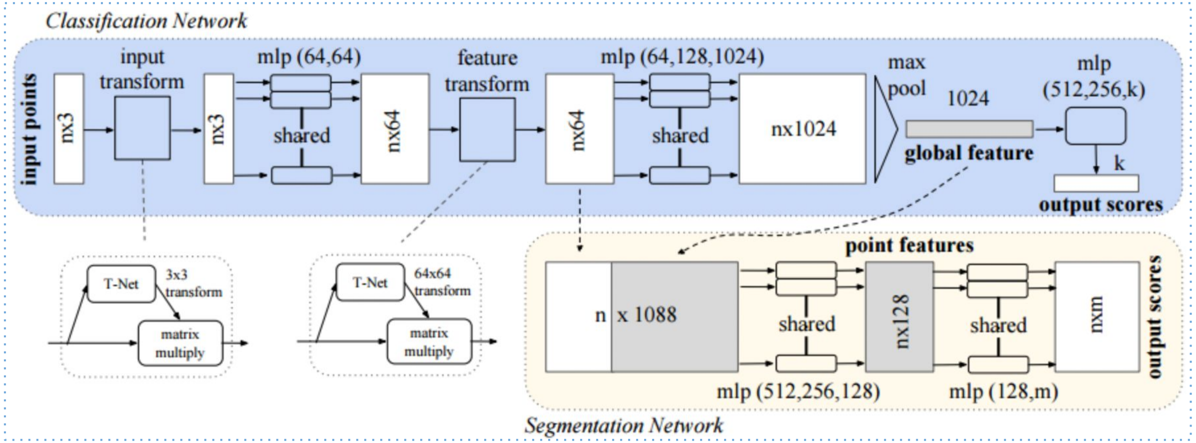
- output : each spatial location encoding the class prediction for each voxel along the z-axis of that location



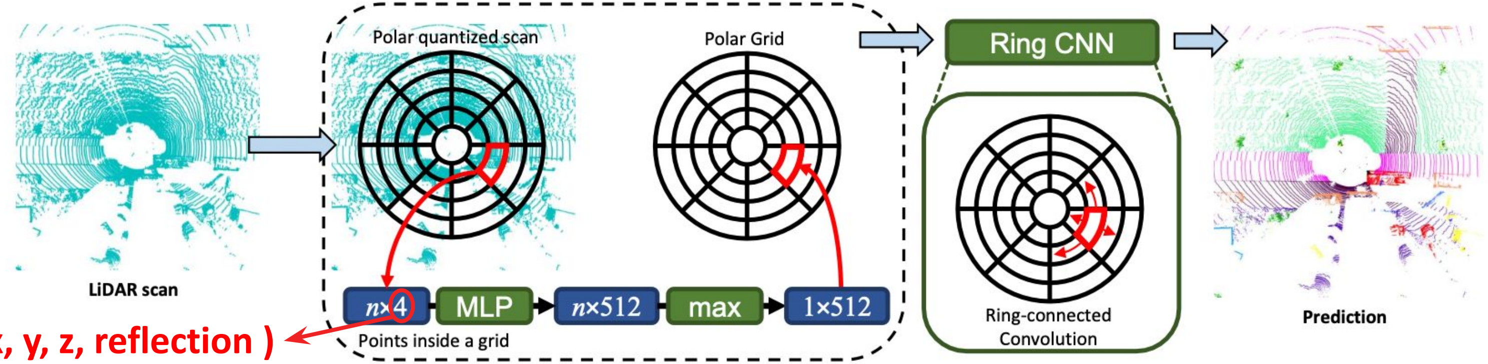
PointPillars: Fast Encoders for Object Detection from Point Clouds, CVPR 2019

Simplified PointNet

: Capture the distribution of points in each grid



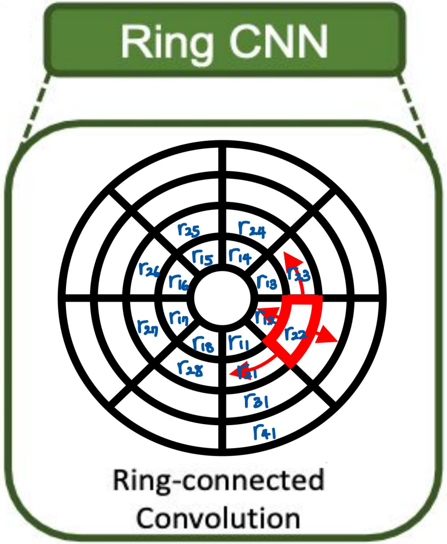

```
if pt_model == 'pointnet':
    self.PPmodel = nn.Sequential(
        nn.BatchNorm1d(fea_dim),

        nn.Linear(fea_dim, 64),
        nn.BatchNorm1d(64),
        nn.ReLU(inplace=True),

        nn.Linear(64, 128),
        nn.BatchNorm1d(128),
        nn.ReLU(inplace=True),

        nn.Linear(128, 256),
        nn.BatchNorm1d(256),
        nn.ReLU(inplace=True),

        nn.Linear(256, out_pt_fea_dim)
    )
```



(x, y, z, reflection)

Ring CNN

: Ring conv kernel will convolve the matrix assuming the matrix is connected on both ends of the radius axis.



r18	r11	r12	r13	r14	r15	r16	r17	r18	r11
r28	r21	r22	r23	r24	r25	r26	r27	r28	r21
r38	r31	r32	r33	r34	r35	r36	r37	r38	r31
r48	r41	r42	r43	r44	r45	r46	r47	r48	r41

```
class double_conv_circular(nn.Module):
    '''(conv => BN => ReLU) * 2'''
    def __init__(self, in_ch, out_ch,group_conv,dilation=1):
        super(double_conv_circular, self).__init__()
        if group_conv:
            self.conv1 = nn.Sequential(
                nn.Conv2d(in_ch, out_ch, 3, padding=(1,0),groups = min(out_ch,in_ch)),
                nn.BatchNorm2d(out_ch),
                nn.LeakyReLU(inplace=True)
            )
            self.conv2 = nn.Sequential(
                nn.Conv2d(out_ch, out_ch, 3, padding=(1,0),groups = out_ch),
                nn.BatchNorm2d(out_ch),
                nn.LeakyReLU(inplace=True)
            )
        else:
            self.conv1 = nn.Sequential(
                nn.Conv2d(in_ch, out_ch, 3, padding=(1,0)),
                nn.BatchNorm2d(out_ch),
                nn.LeakyReLU(inplace=True)
            )
            self.conv2 = nn.Sequential(
                nn.Conv2d(out_ch, out_ch, 3, padding=(1,0)),
                nn.BatchNorm2d(out_ch),
                nn.LeakyReLU(inplace=True)
            )

    def forward(self, x):
        #add circular padding
        x = F.pad(x,(1,1,0,0),mode = 'circular')
        x = self.conv1(x)
        x = F.pad(x,(1,1,0,0),mode = 'circular')
        x = self.conv2(x)
        return x
```

➡ Extend receptive fields

Ring CNN

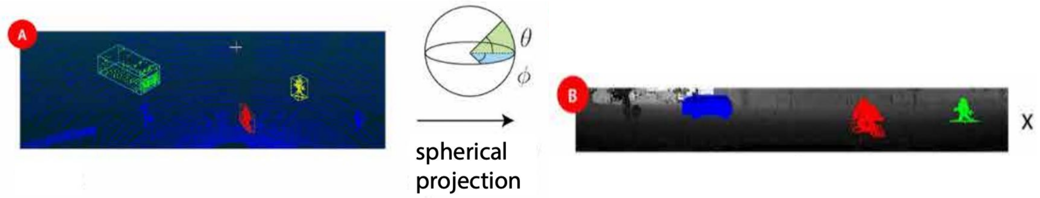
It is applicable to various backbone models.

- Unet stand out from those backbone segmentation networks

Table 3. How projection methods impact models' segmentation performance on **val** split of SemanticKITTI.

Model	Projection	FPS	Latency	MACs	Params	mIoU	Per class IoU car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign
Squeezeseg	Spherical	83.6	0.012s	14B	0.9M	31.8%	79.4%	0.0%	0.0%	3.2%	1.3%	0.0%	0.0%	0.0%	90.9%	19.8%	74.7%	0.0%	75.3%	31.6%	80.6%	37.3%	71.1%	13.2%	26.3%
	Cartesian BEV	19.5	0.051s	101B	1.5M	**42.6%**	90.4%	15.2%	16.6%	13.5%	16.8%	39.0%	45.8%	0.0%	85.7%	25.3%	65.2%	0.0%	86.1%	32.1%	79.7%	54.4%	60.1%	50.9%	33.2%
	Polar BEV	17.8	0.056s	105B	1.5M	42.2%	89.8%	22.1%	19.8%	14.2%	9.2%	37.0%	14.3%	0.4%	83.7%	15.8%	65.6%	0.0%	85.9%	40.2%	85.6%	54.2%	72.1%	54.9%	36.7%
Resnet-FCN	Spherical	38.6	0.048s	92B	117M	41.6%	82.3%	1.5%	13.7%	65.8%	15.5%	20.3%	31.2%	0.0%	92.1%	32.4%	75.6.2%	0.1%	77.3%	31.6%	78.1%	43.9%	66.8%	36.6%	25.2%
	Cartesian BEV	11.7	0.088s	197B	117M	49.2%	89.9%	28.2%	15.6%	56.5%	30.5%	41.0%	66.1%	0.0%	88.6%	38.3%	71.5%	6.1%	86.5%	30.4%	81.5%	52.2%	65.7%	46.7%	39.3%
	Polar BEV	11.5	0.091s	200B	117M	**52.5%**	92.1%	22.8%	36.2%	57.5%	24.6%	42.5%	63.9%	0.0%	92.1%	43.6%	77.5%	1.7%	90.0%	46.9%	84.4%	56.0%	73.1%	53.3%	40.2%
DRN-DL	Spherical	39.1	0.038s	94B	41M	43.4%	82.6%	3.1%	24.5%	51.1%	18.3%	27.3%	23.9%	0.0%	93.0%	37.2%	77.4%	0.2%	76.8%	42.1%	79.7%	46.2%	68.7%	39.2%	32.9%
	Cartesian BEV	10.0	0.100s	171B	41M	46.7%	90.4%	14.1%	20.3%	51.4%	37.3%	39.3%	42.3%	0.0%	87.6%	30.6%	68.0%	1.5%	86.5%	33.0%	83.2%	49.2%	69.8%	44.3%	39.0%
	Polar BEV	9.9	0.101s	173B	41M	**51.2%**	91.6%	19.4%	35.0%	34.6%	20.8%	50.8%	55.1%	0.0%	92.5%	38.6%	77.5%	1.1%	88.5%	44.4%	84.8%	59.7%	70.6%	56.7%	40.2%
Resnet-DL	Spherical	89.5	0.031s	45B	59M	41.6%	81.0%	0.6%	17.1%	58.9%	12.1%	21.3%	24.7%	0.0%	92.5%	33.5%	76.4%	0.0%	76.0%	40.4%	78.6%	45.7%	68.3%	35.1%	28.6%
	Cartesian BEV	11.8	0.090s	107B	60M	50.4%	92.6%	17.8%	41.9%	62.0%	24.2%	42.0%	66.3%	0.0%	87.1%	27.2%	69.6%	0.4%	87.4%	41.5%	84.7%	54.8%	71.0%	48.7%	39.1%
	Polar BEV	11.7	0.094s	109B	60M	**53.6%**	91.5%	30.7%	38.8%	46.4%	24.0%	54.1%	62.2%	0.0%	92.4%	47.1%	78.0%	1.8%	89.1%	45.5%	85.4%	59.6%	72.3%	58.1%	42.2%

Spherical Projection



(A) LiDAR Point Cloud

$$\phi = \arcsin \frac{y}{\sqrt{x^2 + y^2}}, \quad \tilde{\phi} = \lfloor \phi / \triangle\phi \rfloor$$

$$\theta = \arcsin \frac{z}{\sqrt{x^2 + y^2 + z^2}}, \quad \tilde{\theta} = \lfloor \theta / \triangle\theta \rfloor$$

$\triangle\theta$ and $\triangle\phi$ are resolutions for discretization

1. The front-view projection essentially has an occlusion.
2. Distance information is lost during projection, which enables points distant in space to locate in neighboring 2D grids and easily get misclassified as the same label.

Table 3. How projection methods impact models' segmentation performance on **val** split of SemanticKITTI.

Model	Projection	FPS	Latency	MACs	Params	mIoU	Per class IoU																		
							car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign
Squeezeseg	Spherical	83.6	0.012s	14B	0.9M	31.8%	79.4%	0.0%	0.0%	3.2%	1.3%	0.0%	0.0%	0.0%	90.9%	19.8%	74.7%	0.0%	75.3%	31.6%	80.6%	37.3%	71.1%	13.2%	26.3%
	Cartesian BEV	19.5	0.051s	101B	1.5M	**42.6%**	90.4%	15.2%	16.6%	13.5%	16.8%	39.0%	45.8%	0.0%	85.7%	25.3%	65.2%	0.0%	86.1%	32.1%	79.7%	54.4%	60.1%	50.9%	33.2%
	Polar BEV	17.8	0.056s	105B	1.5M	42.2%	89.8%	22.1%	19.8%	14.2%	9.2%	37.0%	14.3%	0.4%	83.7%	15.8%	65.6%	0.0%	85.9%	40.2%	85.6%	54.2%	72.1%	54.9%	36.7%
Resnet-FCN	Spherical	38.6	0.048s	92B	117M	41.6%	82.3%	1.5%	13.7%	65.8%	15.5%	20.3%	31.2%	0.0%	92.1%	32.4%	75.6.2%	0.1%	77.3%	31.6%	78.1%	43.9%	66.8%	36.6%	25.2%
	Cartesian BEV	11.7	0.088s	197B	117M	49.2%	89.9%	28.2%	15.6%	56.5%	30.5%	41.0%	66.1%	0.0%	88.6%	38.3%	71.5%	6.1%	86.5%	30.4%	81.5%	52.2%	65.7%	46.7%	39.3%
	Polar BEV	11.5	0.091s	200B	117M	**52.5%**	92.1%	22.8%	36.2%	57.5%	24.6%	42.5%	63.9%	0.0%	92.1%	43.6%	77.5%	1.7%	90.0%	46.9%	84.4%	56.0%	73.1%	53.3%	40.2%
DRN-DL	Spherical	39.1	0.038s	94B	41M	43.4%	82.6%	3.1%	24.5%	51.1%	18.3%	27.3%	23.9%	0.0%	93.0%	37.2%	77.4%	0.2%	76.8%	42.1%	79.7%	46.2%	68.7%	39.2%	32.9%
	Cartesian BEV	10.0	0.100s	171B	41M	46.7%	90.4%	14.1%	20.3%	51.4%	37.3%	39.3%	42.3%	0.0%	87.6%	30.6%	68.0%	1.5%	86.5%	33.0%	83.2%	49.2%	69.8%	44.3%	39.0%
	Polar BEV	9.9	0.101s	173B	41M	**51.2%**	91.6%	19.4%	35.0%	34.6%	20.8%	50.8%	55.1%	0.0%	92.5%	38.6%	77.5%	1.1%	88.5%	44.4%	84.8%	59.7%	70.6%	56.7%	40.2%
Resnet-DL	Spherical	89.5	0.031s	45B	59M	41.6%	81.0%	0.6%	17.1%	58.9%	12.1%	21.3%	24.7%	0.0%	92.5%	33.5%	76.4%	0.0%	76.0%	40.4%	78.6%	45.7%	68.3%	35.1%	28.6%
	Cartesian BEV	11.8	0.090s	107B	60M	50.4%	92.6%	17.8%	41.9%	62.0%	24.2%	42.0%	66.3%	0.0%	87.1%	27.2%	69.6%	0.4%	87.4%	41.5%	84.7%	54.8%	71.0%	48.7%	39.1%
	Polar BEV	11.7	0.094s	109B	60M	**53.6%**	91.5%	30.7%	38.8%	46.4%	24.0%	54.1%	62.2%	0.0%	92.4%	47.1%	78.0%	1.8%	89.1%	45.5%	85.4%	59.6%	72.3%	58.1%	42.2%

Result

Table 1. Segmentation results on **test** split of SemanticKITTI.

Model	FPS	Latency	MACs	Params	Acc	mIoU	Per class IoU																		
							car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign
PointNet [22]	11.5	0.087s	141B	3.5M	-	14.6%	46.3%	1.3%	0.3%	0.1%	0.8%	0.2%	0.2%	0.0%	61.6%	15.8%	35.7%	1.4%	41.4%	12.9%	31.0%	4.6%	17.6%	2.4%	3.7%
PointNet++ [23]	-	-	-	6M	-	20.1%	53.7%	1.9%	0.2%	0.9%	0.2%	0.9%	1.0%	0.0%	72.0%	18.7%	41.8%	5.6%	62.3%	16.9%	46.5%	13.8%	30.0%	6.0%	8.9%
Squeezeseg [35]	49.2	0.031s	13B	0.9M	-	29.5%	68.8%	16.0%	4.1%	3.3%	3.6%	12.9%	13.1%	0.9%	85.4%	26.9%	54.3%	4.5%	57.4%	29.0%	60.0%	24.3%	53.7%	17.5%	24.5%
TangentConv [30]	-	-	-	0.4M	-	35.9%	86.8%	1.3%	12.7%	11.6%	10.2%	17.1%	20.2%	0.5%	82.9%	15.2%	61.7%	9.0%	82.8%	44.2%	75.5%	42.5%	55.5%	30.2%	22.2%
Squeezesegv2 [36]	36.7	0.036s	14B	0.9M	-	39.7%	81.8%	18.5%	17.9%	13.4%	14.0%	20.1%	25.1%	3.9%	88.6%	45.8%	67.6%	17.7%	73.7%	41.1%	71.8%	35.8%	60.2%	20.2%	36.3%
DarkNet53 [1]	12.7	0.087s	378B	50M	87.8%	49.9%	86.4%	24.5%	32.7%	25.5%	22.6%	36.2%	33.6%	4.7%	**91.8%**	64.8%	74.6%	**27.9%**	84.1%	55.0%	78.3%	50.1%	64.0%	38.9%	52.2%
RangeNet++ [19]	-	-	378B	50M	89.0%	52.2%	91.4%	25.7%	**34.4%**	25.7%	23.0%	38.3%	38.8%	4.8%	**91.8%**	**65.0%**	**75.2%**	27.8%	87.4%	58.6%	80.5%	55.1%	64.6%	47.9%	55.9%
RandLA [12]	-	-	-	1.2M	-	53.9%	**94.2%**	26.0%	25.8%	**40.1%**	**38.9%**	**49.2%**	**48.2%**	**7.2%**	90.7%	60.3%	73.7%	20.4%	86.9%	56.3%	81.4%	**66.8%**	49.2%	47.7%	38.1%
Unet w/ Cartesian BEV	19.7	0.051s	134B	14M	87.6%	50.7%	92.7%	26.8%	23.1%	26.7%	24.2%	48.1%	41.0%	4.4%	86.7%	52.3%	67.2%	12.9%	89.5%	57.7%	80.8%	62.5%	62.5%	50.3%	53.5%
PolarNet	16.2	0.062s	135B	14M	90.0%	**54.3%**	93.8%	**40.3%**	30.1%	22.9%	28.5%	43.2%	40.2%	5.6%	90.8%	61.7%	74.4%	21.7%	**90.0%**	**61.3%**	**84.0%**	65.5%	**67.8%**	**51.8%**	**57.5%**

Table 2. Segmentation results on **test** split of A2D2.

Model	FPS	Latency	MACs	Params	Acc	mIoU	Per class IoU																	
							car	bicycle	pedestrian	truck	small vehicles	traffic signal	traffic sign	utility vehicle	sidebars	speed bumper	curbstone	solid line	irrelevant signs	road blocks	tractor	non-drivable street	zebra crossing
Squeezeseg [35]	87.5	0.009s	15B	0.9M	-	8.9%	9.7%	0.0%	0.0%	15.8%	0.0%	0.7%	64.4%	0.0%	0.4%	0.0%	2.2%	15.6%	0.5%	15.9%	0.0%	0.0%	0.0%
Squeezesegv2 [36]	67.1	0.015s	15B	0.9M	81.0%	16.4%	15.4%	0.2%	8.6%	63.8%	0.0%	16.8%	61.7%	0.6%	0.1%	0.0%	14.8%	24.7%	12.7%	33.2%	0.0%	5.8%	0.0%
DarkNet53 [1]	16.1	0.063s	378B	50M	82.0%	17.2%	15.2%	0.8%	6.1%	68.5%	0.0%	15.5%	63.8%	0.4%	0.3%	0.0%	17.3%	23.8%	13.3%	35.6%	0.0%	6.3%	0.0%
Unet w/ Cartesian BEV	49.5	0.028s	60B	14M	83.5%	20.3%	27.0%	7.3%	20.3%	66.0%	1.9%	25.2%	54.7%	6.5%	12.7%	0.0%	20.3%	26.8%	21.4%	42.5%	0.0%	9.5%	0.0%
PolarNet	38.4	0.031s	60B	14M	**85.4%**	23.9%	23.8%	10.1%	18.2%	69.7%	9.6%	49.1%	58.5%	0.0%	11.3%	0.0%	28.3%	37.6%	24.8%	42.8%	0.0%	14.8%	0.0%

Table 4. Segmentation results on **test** split of Paris-Lille-3D.

Model	Acc	mIoU	Per class IoU								
			ground	building	pole	bollard	trash can	barrier	pedestrian	car	vegetation
Squeezesegv2 [36]	87.3%	36.9%	95.9%	82.7%	18.7%	9.9%	3.8%	15.2%	3.4%	49.9%	52.8%
DarkNet53 [1]	**88.9%**	40.0%	96.7%	**84.9%**	19.5%	16.7%	4.8%	17.6%	3.4%	58.2%	**57.9%**
Unet w/ Cartesian BEV	80.9%	40.3%	96.0%	44.0%	**38.4%**	**42.8%**	**12.7%**	12.4%	**12.1%**	70.4%	33.60%
PolarNet	87.5%	**43.7%**	**96.8%**	69.1%	32.2%	27.6%	2.4%	**27.5%**	**12.1%**	**74.0%**	51.60%

3D Segmentation Result on SemanticKITTI

Rank	Model	mIoU	Paper	Code	Result	Year	Tags
1	**AF2S3Net**	70.8%	(AF)2-S3Net: Attentive Feature Fusion with Adaptive Feature Selection for Sparse Semantic Segmentation Network			2021	
2	**Cylinder3D**	68.9%	Cylindrical and Asymmetrical 3D Convolution Networks for LiDAR Segmentation			2020	
3	**SPVNAS**	66.4%	Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution			2020	
4	**KPRNet**	63.1%	KPRNet: Improving projection-based LiDAR semantic segmentation			2020	
5	**TORNADONet-HiRes**	63.1%	TORNADO-Net: mulTiview tOtal vaRiatioN semAntic segmentation with Diamond inceptiOn module			2020	
6	**Cylinder3D**	61.8%	Cylinder3D: An Effective 3D Framework for Driving-scene LiDAR Semantic Segmentation			2020	
11	**PolarNet**	57.2%	PolarNet: An Improved Grid Representation for Online LiDAR Point Clouds Semantic Segmentation			2020	

Polar Coordinate