# PointPillars: Fast Encoders for Object Detection from Point Clouds

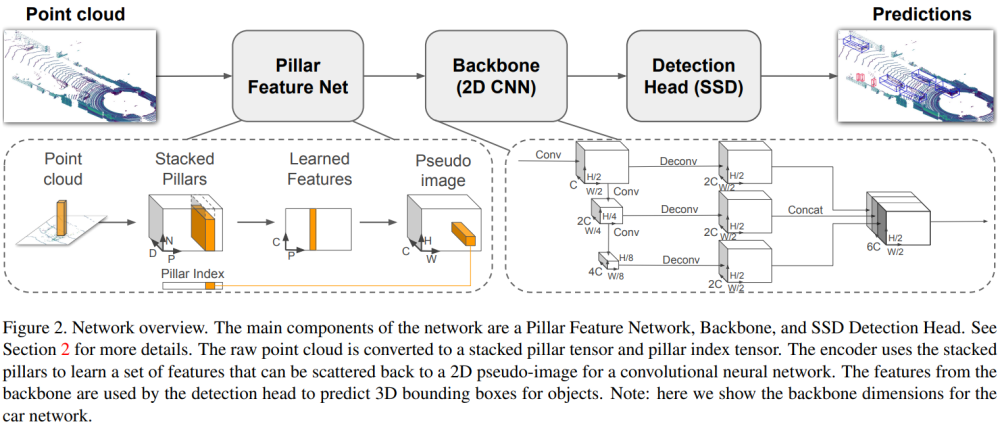
## 论文地址

<https://arxiv.org/abs/1812.05784>

## 代码地址

<https://github.com/nutonomy/second.pytorch>

## 模型架构



### Pillar Feature Net(PFN)

* 首先，输入点云被分割成多个Pillar单元，每个Pillar是在XY平面上（笛卡尔坐标系）以一定的步长对点云进行划分得到的一个3维的小单元格

* 然后每个Pillar中的每个点云被编码成一个9维的向量**D**：(x,y,z,r,xc,yc,zc,xp,yp)，其中x,y,z,r分别表示点云在三维空间中的3个坐标和反射强度；xc,yc,zc表示到该Pillar中所有点的算术平均值点的距离，xp,yp表示该点到该Pillar的x,y中心的偏移值。由于点云数据的稀疏性，可能很多Pillar都不含点云或者包含的点云数量比较少，考虑到计算复杂度的问题，会对Pillar的数量进行限制，最多处理**P**个非空的Pillar，同时每个Pillar中最多包含**N**个点云特征向量，如果点云数大于N，则采用随机采样的方法从中选取N个，反之，如果点云的数量少于N，则用零填充的方法填充到N个。通过上述方法，就将一帧点云数据编码成了一个维度为**(D,P,N)**的稠密张量

* 接下来，用一个简化版的PointNet网络进行处理。首先将每个包含D维特征的点用一个**线性层+BatchNorm+ReLU**激活函数处理后，生成维度为(C,P,N)的张量；然后对每个Pillar单元进行最大池化操作，得到维度为**(C,P)**的张量

* 最后一步是通过一个scatter算子生成伪图像。就是通过每个点的Pillar索引值将上一步生成的(C,P)张量转换回其原始的Pillar坐标用来创建大小为**(C,H,W)**的伪图像。伪图像的高度H和宽度W：在第一步对点云进行Pillar划分的时候会设置XY平面上点云坐标的范围和每个Pillar的大小，假设X轴的范围是[0,69.12]，Y轴的范围是[-39.68,39.68]，每个Pillar的大小是``0.16x0.16，那么以X轴表示宽，Y轴表示高，一个Pillar表示一个像素的话，那么这个伪图像的宽W = (69.12 - 0) / 0.16 = 432，高H = (39.68 -(-39.68)) / 0.16 = 496

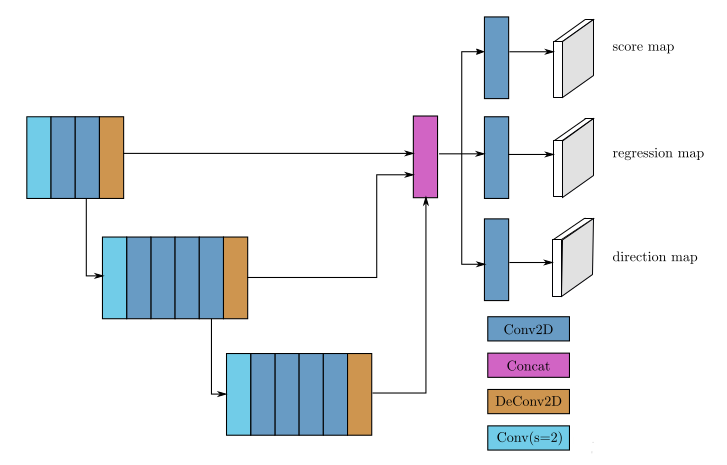
* **P=12000、N=100、D=128**

### Backbone

* PointPillars采用了与VoxelNet一样的RPN(Region Proposal Network)骨干网络，该部分由一个2D卷积神经网络组成，其作用是用于在第一部分网络输出的伪图像上提取高维特征。RPN骨干网分为两个子网络：一个自顶向下的子网络用于在越来越小的空间分辨率特征图上提取特征，另一个子网络则负责将不同分辨率特征图上提取的特征通过反卷积操作进行上采样至同样维度大小然后进行串联，具体架构图见整体架构图

### Detection Head

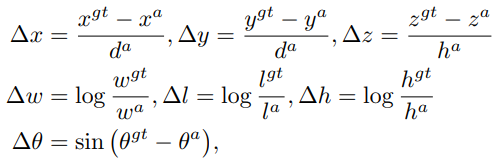
* PointPillars采用了SSD的检测头用来实现3D目标检测。与SSD类似，PointPillars在2D网格中进行目标检测，而Z轴坐标和高度则是通过回归的方式得到



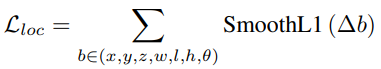
### 损失函数

* PointPillars采用了与SECOND一样的损失函数。

* 每个目标的3D框用一个7维向量来表示：，其中x，y，z表示3D框的中心点坐标；w，l，h分别表示3D框的宽、长、高；表示3D框的朝向角。那么在检测框定位回归任务中Ground truth和锚框之间的残差定义为：



其中，，则定位损失使用L1损失如下所示:



: 由于localization loss不能区分box的  , 所以加上direction loss

分类损失使用focal loss:



α = 0.25 and γ = 2

总的损失如下所示:

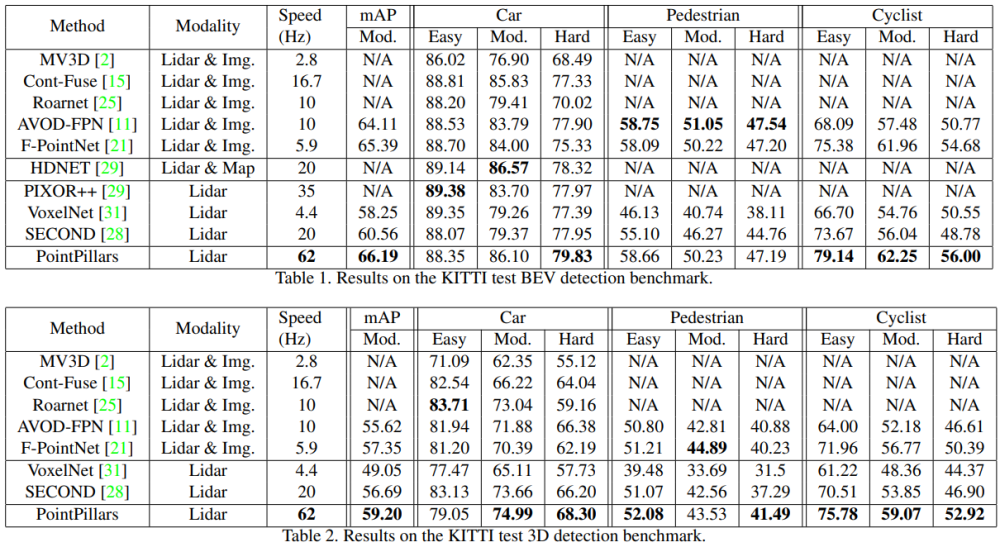


where Npos is the number of positive anchors and βloc = 2, βcls = 1, and βdir = 0.2

### 实验对比

学习率: 2 \* 10e-4

优化器: Adam，每隔15epoch学习率衰减0.8倍，160epoch



### 总结

#### 优势:

* 一种新的点云编码方式，丢弃了原来使用的3D卷积，直接使用2D卷积，部署方便，速度快，优点明显，精度上也不错，其中Apollo 6.0使用了该方法

* 在柱状体而非体素（voxel）上进行操作，无需手动调整垂直方向的分箱

* 不适用手工特征，而是让网络自动的学习特征，因此无需手动的调整点云的配置，即可推广到其他的激光雷达中使用

#### 改进点:

* 特征融合部分采用的FPN是否可以换成 PAN

* 检测头采用的 SSD 是否可以换成更新的检测器

* 回归损失函数是否可以将角度和 BBox 紧耦合