几何计算前沿

2300012908 晋禹超

2025 年 5 月 12 日

PointNet

1、核心思想

设计一种可以直接对输入的点云数据进行处理的深度神经网络，适配点云数据无序性，非孤立性，仿射变换无关性等特点。
具体地，将输入的一帧全部点云数据表示为一个$n \times 3$的$2dtensor$，然后与经过T-Net学习到转换矩阵相乘，然后通过多次MLP对各点云数据进行特征提取，在用一个T-Net对特征进行对齐，然后在每个特征维度上使用max pooling得到最终的全局特征，最后经过一个MLP预测分类分数，输出分类结果。

2、方法

(1)无序性：PointNet设计了一种对称函数用于表征
因为难以找到一种稳定的排序方法，所以放弃了将点云中的点以某种顺序输入（比如按坐标轴从小到大）；
因为RNN难以处理好成千上万长度的输入元素，所以也不适合训练一个RNN来学习排布的不变性。
于是，PointNet选择使用一个简单的对称函数去聚集每个点的信息：
$f(\{x_1, ..., x_n\}) \approx g(h(x_1), ...h(x_n))$
其中，左边f是目标函数，g是期望得到的对称函数，即对各个元素分别用h处理，然后传入对称函数g中，以实现排列不变性。具体了，作者使用了MLP作为h，$max\ pooling$作为g。
(2)非孤立性：PointNet采用局部全局特征结合
(3)某些变换无关性：PointNet设计了一个小型网络T-Net用于预测空间变换矩阵，对输入的点进行变换。将所有的输入点集对齐到一个统一的点集空间，用于实现网络对于仿射变换、刚体变换等变换的无关性。比较特殊的是，在特征空间的对齐中，由于维度比较高，所以生成对齐矩阵的维度很大，不好优化，要在loss加入一项$L_{reg} = ||I - AA^T||_F^2$进行约束。

3、My running results

原文代码仓库的实现是Tensorflow版本，我本次作业参考了pytorch版本的实现。训练了200个epoch，每个epoch训练完后都在测试集测试，训练过程中训练集的loss和instance accuracy以及测试集的instance accuracy和class accuracy如下：

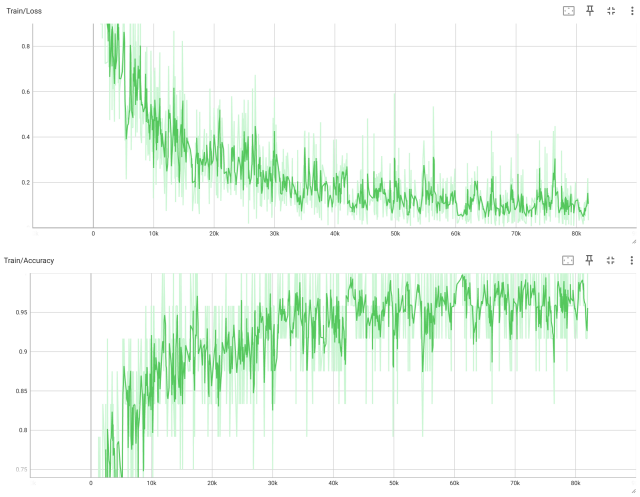


图 1: Training Curves

最终模型在测试集上的instance accuracy为90.67%，class accuracy为87.01%，

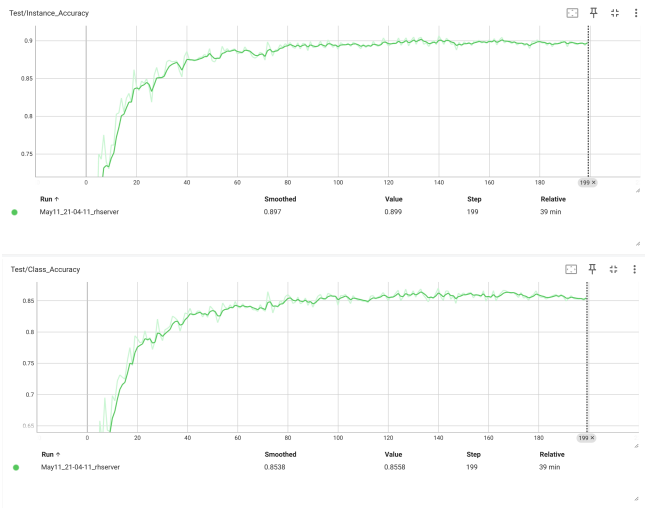


图 2: Testing Curves

```
(ocnn) (base) jyc@rhserver:~/homework3/Part1_PointNet$ python test_classification.py --log_dir='pointnet_log'
PARAMETER ...
Namespace(use_cpu=False, gpu='0', batch_size=24, num_category=40, num_point=1024, log_dir='pointnet_log', use_normals=False, use_uniform_sample=False, num_votes=3)
Load dataset ...
The size of test data is 2468
100%| | 103/103 [00:03<00:00, 31.70it/s]
Test Instance Accuracy: 0.906715, Class Accuracy: 0.870144
```

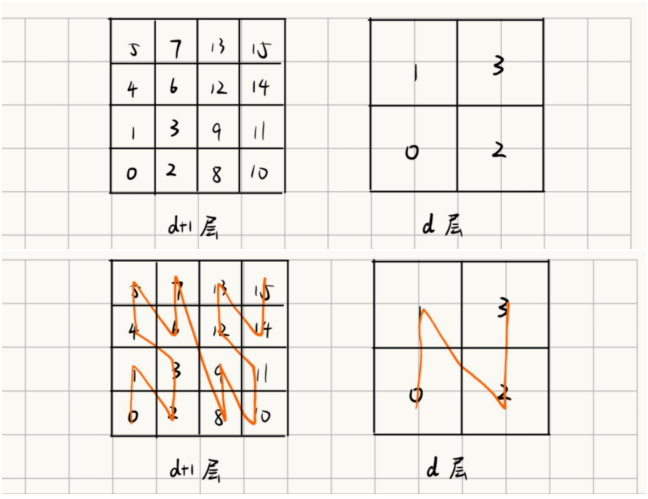
原论文中的instance accuracy为89.2%，class accuracy为86.2%，pytorch版代码参库只给出了instance accuracy为90.6%，看以看出pytorch版的实现分类结果略优于原版的Tensorflow实现。

OCNN

1、核心思想

OCNN的核心思想是使用平均法向量作为CNN的输入信号而不是binary indication function；把三维空间的卷积限定在有3D shape存在的区域，避免全体素空间卷积；使用一个连续的一维数组把同一个层级下的八叉树节点连续储存在一起，便于卷积操作中的查邻域。所有的这些设计使得卷积更快，效果更好，内存占用变小，使得处理高分辨率的3D shape输入成为可能。

如何把每一级的数据组织成一个连续数组的形式，论文提出的方法颇具特色，具体采用了数学中分形的思想，借助'Z'字形的分形结构，把同一级的数据串在一起，使得卷积查邻域变成常数时间，如图以二维四叉树为例：



2、方法

(1) 八叉树构建：
step1 将输入的3D模型均匀缩放，并且轴对称地放在一个有边界的3D盒子中
step2 广度优先递归地划分盒子，每一步，遍历非空的(即有3D数据占有)格子，把它们划分成8个子节点，作为下一层
step3 重复直到达到预定义的最大深度
最后，把输入信号(average normal vector)存在finest octant(划分最精细的那一层的子节点)，卷积得到的特征记录在每一个节点。
(2)shuffle key：
这是O-CNN的核心。一个在l层的octant O的shuffle key是一个长度为3*l的0-1字串，编码了它在这个3D形状中的位置。
$key(O) := x_1y_1z_1x_2y_2z_2...x_ly_lz_l$
$x_i, y_i, z_i \in \{0, 1\}$，表示第i层的节点与它的父节点的相对位置，$2^3 = 8$，正好可以表示八个孩子。第l层的shuffle key 存入S_l。
(3)Label：
L_l表示第l层的标签，$p_l(O)$表示octant O是l层的第几个非空octant，空的Octant标签为0。
(4)输入信号：
使用finest octant处sample到的average normal vector作为CNN的输入信号：空子节点处设置为0信号；非空子节点处采样一些点，然后计算它们的average normal vector作为CNN的输入信号作为信号。最后把所有的leaf octant的信号合成一个向量，长度是所有finest octant的个数。
(5)CNN feature：
每个定义在第l层的卷积核，记录每个octant节点的卷积结果在feature map vector T_l中。
(6) 关于3D model输入batch的处理：
一个batch中3D shape对应的八叉树层级结构一般不同，为了便于在GPU上训练，把一个batch中的octree合在一起拼成一棵super-octree，即把每一层的property vectors(L, S, T)拼接在一起，形成super-octree的L^*, S^*, T^*，用shuffle key的高8位记录batch中的物体序号，L^*中的序号更新为整个super-octree中的排序。
(7) Object classification
在物体分类任务中，CNN按照LeNet的思路设计。
用U_l表示作用在第l层的一个基本单元，$U_l = convolution + BN + ReLU + pooling$
则$OCNN(d)$可以表示为$input \longrightarrow U_d \longrightarrow\longrightarrow U_{d-1} \longrightarrow ... \longrightarrow U_2$
为了对齐octree的特征，规定第2层的所有octant存在，不在在的补0。
于是该任务中，网络结构可以表示为：
$OCNN(d) \longrightarrow Dropout \longrightarrow FC(128) \longrightarrow Dropout \longrightarrow FC(N_c) \longrightarrow softmax \longrightarrow output$
N_c 表示分类数

3、My running results

使用了深度为5的Octree，CNN模型使用了LeNet，训练了300个epoch，每训练完5个epoch后都在测试集测试，结果如下：
最终测试结果在测试集上的分类准确率是91.6%，代码官方文档上的准确率是91.7%，原论文中的准确率是90.4%，运行结果与预期接近。

```
The start_epoch is 1
=> Epoch: 0, test/loss: 0.271, test/accu: 0.916, time: 2025/05/12 13:36:33, dur
   ation: 3.39s
(ocnn) (base) jyc@rhserver:~/homework3/Part2_OCNN/projects$
```

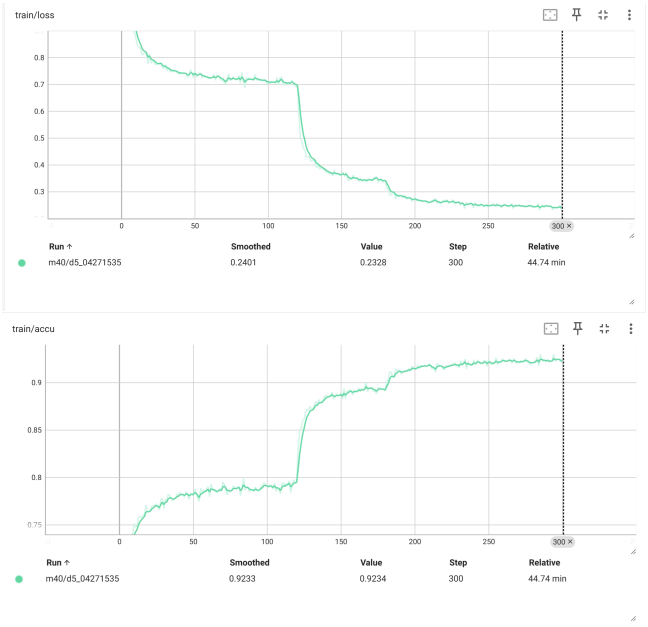


图 4: Training Curves

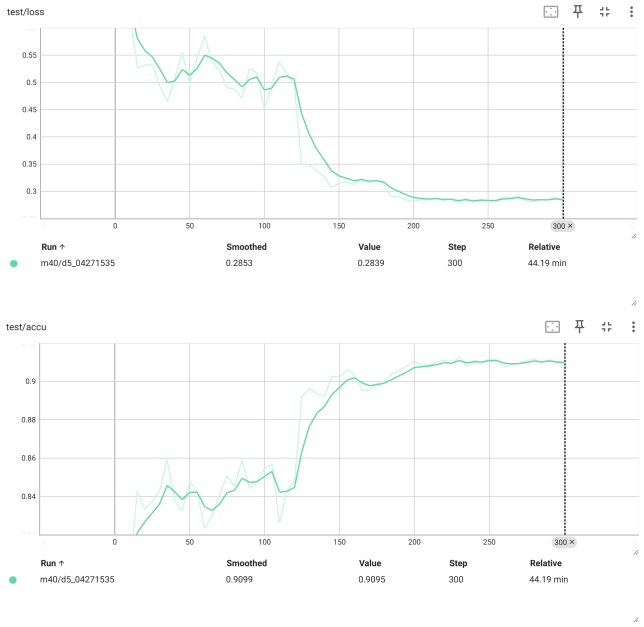


图 5: Testing Curves

Comparison analysis

对比项目	PointNet	OCNN
参数量（处理ModelNet40数据集）	较大，约39.8MB	小得多，约2.3MB(深度5，采用LeNet)
运行速度（使用RTX 4060 16G运行）	较快，只需要35min左右即可完成训练	较慢，需要先花30min左右处理数据，然后再花45min左右进行训练
分类效果（以instance accuracy为例）	略差，89.2%到90.6%	略好，90.4%到91.7%
输入格式	原始的无序点云(xyz坐标)	要先处理成八叉树结构
局部特征	较难编码局部几何结构	八叉树可以较好地捕捉局部几何信息
输入点云稠密性	受限于max_pooling，难以处理密集点云	稀疏结构的效果略逊于PointNet
使用场景	简单物体分类	复杂、大规模场景