使用深度学习的三维点云分类的介绍

原创 2018-01-06 博主particle [点云PCL](https://mp.weixin.qq.com/s?__biz=MzI0MDYxMDk0Ng==&mid=2247484241&idx=1&sn=9ee13bca0de1be9d0b4752f5faed8f35&chksm=e91978bcde6ef1aad771bacb1d59d67550bfa458d1582be9167ba9ca5095f33a964ac6dca10b&scene=38&key=3994e6ec5ef900e53f1f98fb7504c273568fdd5c4f7ac29fd409ea7ebd8e84893fce81736670d277f815b734620d3d9445d6c10604caee30e95f48653a30ef960285778777d8c67e2e213af4124167b1&ascene=7&uin=MTE4Nzc3MDYyMg%3D%3D&devicetype=Windows+10&version=62060028&lang=zh_CN&pass_ticket=Pm6YQmMUif60KgzW56pdGgTOe%2BzVoBZgv1Y5EXOMhj6BN4N6VRyYyP5hVanf0M7h&winzoom=1##)

   在过去的这些年里，对二维图像已经有了大量深入的研究，并且有着长足的发展。它在分类任务上取得了极好的结果主要得益于一下两个关键因素：

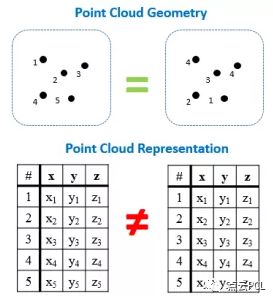
**1.卷积神经网络。**

**2.数据 - 大量图像数据可用。**

   但是对于3D点云，数据正在迅速增长。大有从2D向3D发展的趋势，比如在opencv中就已经慢慢包含了3D点云的处理的相关模块，在数据方面点云的获取也是有多种渠道， 无论是源于CAD模型还是来自LiDAR传感器或RGBD相机的扫描点云，无处不在。 另外，大多数系统直接获取3D点云而不是拍摄图像并进行处理。因此，在深度学习大火的年代，应该如何应用这些令人惊叹的深度学习工具，在3D点云上的处理上达到对二维图像那样起到很好的作用呢？

3D点云应用深度学习面临的挑战。首先在神经网络上面临的挑战：

（1）非结构化数据（无网格）：点云是分布在空间中的XYZ点。 没有结构化的网格来帮助CNN滤波器。  
（2）不变性排列：点云本质上是一长串点（nx3矩阵，其中n是点数）。 在几何上，点的顺序不影响它在底层矩阵结构中的表示方式，例如， 相同的点云可以由两个完全不同的矩阵表示。 如下图所示：

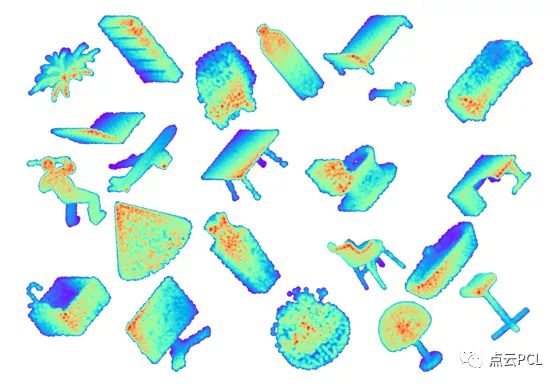


（3）点云数量上的变化：在图像中，像素的数量是一个给定的常数，取决于相机。 然而，点云的数量可能会有很大的变化，这取决于各种传感器。

在点云数据方面的挑战：

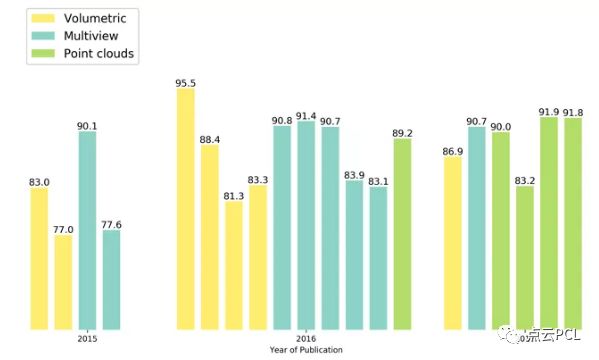
（1）缺少数据：扫描的模型通常被遮挡，部分数据丢失。  
（2）噪音：所有传感器都是嘈杂的。 有几种类型的噪声，包括点云扰动和异常值。 这意味着一个点有一定的概率位于它被采样的地方（扰动）附近的某一半径范围内，或者它可能出现在空间的任意位置（异常值）。  
（3）旋转：一辆车向左转，同一辆车向右转，会有不同的点云代表同一辆车

Princeton’s Modelnet40 dataset。 它包含约40个对象类别（如飞机，表格，植物等），用三角形网格表示的12311个CAD模型。 数据分为9843个培训模式和2468个测试模式，如下图



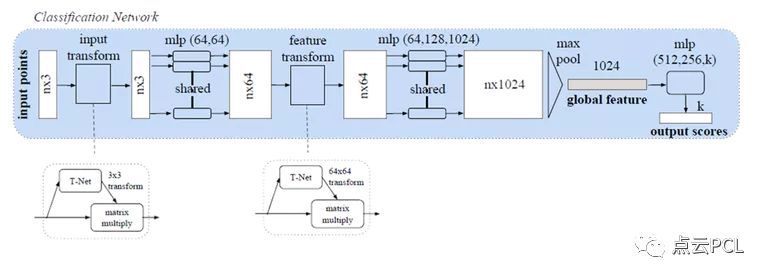
在点云上应用深度学习的直接方法是将数据转换为体积表示。 例如体素网格。 这样我们就可以用没有神经网络问题的3D滤波器来训练一个CNN（网格提供了结构，网格的转换解决了排列问题，体素的数量也是不变的）。 但是，这有一些不利因素。 体积数据可能变得非常大，非常快。 让我们考虑256×256 = 65536像素的典型图像大小，现在让我们添加一个维度256x256x256 = 16777216体素。 这是很大的数据量（尽管GPU一直在发展）。 这也意味着非常缓慢的处理时间。 因此，通常我们需要妥协并采取较低的分辨率（一些方法使用64x64x64），但是它带来了量化误差的代价。所以，所需的解决方案是一个直接的深度学习的方法，将是3D点云应用深度学习的重点。

作者调查了三种最近发表的文章，主要针对对点云进行深度学习的论文。 正如下图所示，展示了3D点云分类准确性出版（准确性，年份和数据类型），它总结了数据集上的最新准确性结果。 以及每种方法正在处理的数据的类型。 可以看到，在2015年，大多数方法都用于多视图数据（这是一种简短的说法 - 让我们拍摄3D模型的几张照片并使用2D方法处理它们），2016年更多的方法使用了体素表示的点云学习和2017年的基于点的方法有了大幅度的增长。



**PointNet（CVPR2017）**

开拓者！ 来自斯坦福大学，他们的工作引起了很多关注。他们做了一些令人惊讶的简单的事情，并证明了为什么它运作良好，他们分别在每个点上训练了一个MLP（在点之间分享权重）。每个点被“投影”到一个1024维空间。然后，他们用点对称函数（max-pool）解决了点云顺序问题。这为每个点云提供了一个1 x 1024的全局特征，这些特征点被送入非线性分类器。利用他们称为T-net的“迷你网络”解决了旋转问题。它学习了点（3 x 3）和中级特征（64 x 64）上的变换矩阵。称之为“迷你”有点让人误解，因为它实际上与主网络的大小有关。另外，由于参数数量的大量增加，引入了一个损失项来约束64×64矩阵接近正交。也使用类似的网络进行零件分割。也做了场景语义分割。做得好！我强烈推荐阅读（或者您也可以观看演示视频）。本文对ModelNet40数据集的准确率高达89.2％。下图是pointNet点云分类的框架

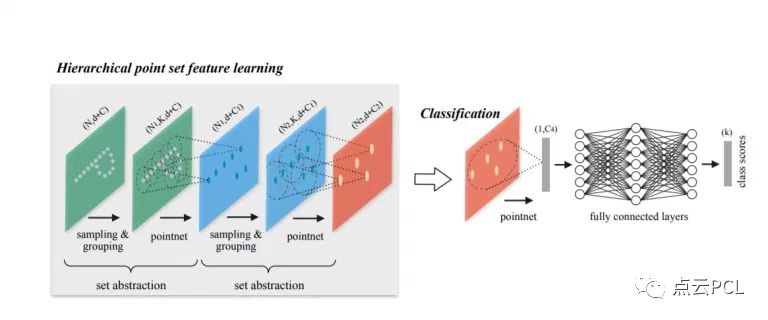


引用: Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classication and segmentation. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.

The code is available on GitHub: PointNet code

**Pointnet ++（NIPS 2017）**

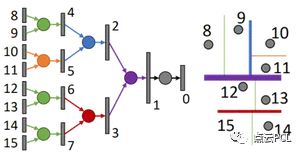
在PointNet之后不久，引入了Pointnet ++。它本质上是PointNet的分层版本。每个图层都有三个子阶段：采样，分组和PointNeting。在第一阶段，选择质心，在第二阶段，把他们周围的邻近点（在给定的半径内）创建多个子点云。然后他们将它们给到一个PointNet网络，并获得这些子点云的更高维表示。然后，他们重复这个过程（样本质心，找到他们的邻居和Pointnet的更高阶的表示，以获得更高维表示）。使用这些网络层中的3个。还测试了不同层级的一些不同聚合方法，以克服采样密度的差异（对于大多数传感器来说这是一个大问题，当物体接近时密集样本，远处时稀疏）。他们在原型PointNet上进行了改进，在ModelNet40上的准确率达到了90.7％。下面是Pointnet++ 架构。

引用: Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. arXiv preprint arXiv:1706.02413, 2017.

**Kd-Network（ICCV 2017）**

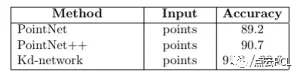
本文使用着名的Kd树在点云中创建一定的顺序结构的点云。一旦点云被结构化，他们就会学习树中每个节点的权重（代表沿特定轴的细分）。每个坐标轴在单个树层级上共享权重如下图中的所有绿色都具有共享权重，因为它们将数据沿x维度细分。测试了随机和确定性的空间细分，并说明了随机版本效果最好。但同时也说出了一些缺点。对旋转（因为它改变树结构）和噪声（如果它改变树结构）敏感。对于每个输入点云数据，都需要上采样，下采样或训练一个新模型。

在Modelnet40上报告了1024点（深度10树）的90.6％准确度数据集和〜32K点（深度15树）的91.8％。做到了部分点云分割，形状检索，并可以在后期工作中尝试其他的树形结构。



引用: Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. arXiv preprint arXiv:1704.01222, 2017.

总结：Pointnet和Pointnet ++使用对称函数来解决顺序问题，而kd-Network使用Kd-tree。 Kd树也解决了结构问题，而在PointNets MLP每个点分别训练。



 该文章翻译http://www.itzikbs.com/3d-point-cloud-classification-using-deep-learning，有问题请指出，同时欢迎大家关注微信公众号,积极分享投稿！一起分享更快乐

或者加入3D视觉微信群一起交流分享