基于体素的方法

1. 体素

1.1. 体素的概念

体素是体积元素(Volume Pixel)的简称，包含体素的立体可以通过立体渲染或者提取给定阈值轮廓的多边形等值面表现出来。一如其名，是数字数据于三维空间分割上的最小单位，体素用于三维成像、科学数据与医学影像等领域。概念上可以理解为二维像素在三维空间的推广，它们是一组分布在正交网格中心的立方体单元。

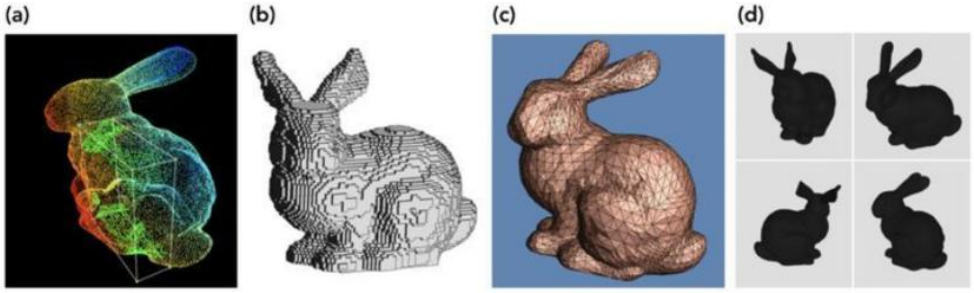


图 1.1 一个 3D 数据的不同表示类型

图 1.1 中是一个 3D 数据的不同表示类型:(a)点云(Point clouds)；(b) 体素网格(Voxel grids)；(c)多边形网格(Polygon meshes)；(d) 多视图表示(Multi-view representations) 。

1.2. 体素化

体素化(Voxelization)是将物体的表面 几何形式表示转换成最接近该物体的体素表示形式，产生体数据集(Volume Datasets)，其不仅 包含模型的表面信息，而且能描述模型内部属性。实现模型体素化的方式有很多，比如基于八叉树的三模网格模型体素化[1]，基于 GPU 并利用渲染管线中 fragment shader 部分实现的栅格化插值。

2. VoxNet 方法

VoxNet[2]:这是一个基本的 3D-CNN 结构，可用于为 3D 点云数据创建快速、准确的对象类检测器。在实验中表示该体系结构在使用三种不同的 3D 数据来源:激光雷达点云、RGBD 点云和 CAD 模型的目标识别任务中达到了最先进的精度。

我们的算法的输入是点云段，如果执行检测，它可以源自诸如的分割方法或“滑动框”。我们的任务就是对这个点云段作分类。网络主要由两部分组成，一个代表空间占用估计的体素网络和一个用于对该被占用的网格进行分类的 CNN。

VoxNet 的结构本身相当简单，它由 2 个卷积层、1 个最大池层和 2 个全连接层组成，最终计算输出类得分向量。由于体素网格与图像非常相似，它们采用的实际跨步卷积和池化操作，只是在 2D 像素上执行的这些操作对 3D 体素的细微适配。卷积操作使用 d×d×d×c 核而非应用于 2D CNNs 中的 d×d×c 核，而池化操作则考虑非重叠的 3D 体素块，而不是 2D 像素块。

2.1. 方法：

2.1.1. Volumetric Occupancy Grid 体积占用网格

占用网格将环境状态表示为随机变量的 3D 网格（每个对应于体素），并根据传入的传感器数据和先验知识保持其占用率的概率估计。

使用原因：

1）它们使我们能够从距离测量中有效地估计空闲、占用和未知的空间，即使对于来自不同视点和时刻的测量也是如此。

2）可以使用简单有效的数据结构存储和操作它们

2.1.2. Reference frame and resolution 参考帧和分辨率

在我们的体积表示中，每个点(x, y, z)被映射到离散的体素坐标(i, j, k)。映射是一个统一的离散化，但取决于空间中体素网格的原点、方向和分辨率。体素化物体的外观很大程度上依赖于这些参数。

原点表示：我们假设它是作为输入给出的，例如通过分割算法或滑动框得到的。

方向表示：我们假设网格框架的 z 轴近似与重力方向一致。

分辨率表示：根据数据集，制定了两种策略。对于我们的 LiDAR 数据集，我们使用固定的空间分辨率，例如$(0.1m)^3$的体素；对于其他数据集，选择分辨率使得感兴趣的对象占据 24×24×24 体素的子体积。在所有实验中，我们使用大小为 32×32×32 体素的固定占据网格。这两种策略之间的权衡是，在第一种情况下，我们保持由物体的相对规模提供的信息(例如，汽车和人往往具有一致的物理尺寸);在第二种情况下，我们避免了当体素太小(这样物体就比网格大)或当体素太大(这样细节就会因为混叠而丢失)时丢失形状信息。

2.1.3. Occupancy models 占有率模型

设$\{z^T\}$是一系列的距离测量，要么命中$(z^t = 1)$，要么通过$(z^t = 0)$给定坐标(i, j, k)的体素。假设一个理想的光束传感器模型，我们使用 3D 射线追踪[32]来计算每个体素命中和穿过的次数。根据这些信息，我们考虑三种不同的入住率网格模型来估计入住率:

1) Binary occupancy grid 二元占用网格

在这个模型中，假定每个体素都有一个二元状态，被占用或未被占用。每个体素占用率的概率估计是用数值稳定性的对数概率计算的。使用如下公式，我们将光束穿过的每个体素更新为：

$$l_{ijk}^t = l_{ijk}^{t-1} + z^t l_{occ} + (1 - z^t) l_{free} \tag{1}$$

其中l_{occ}和l_{free}分别是给定测量命中或未命中单元格的单元格被占用或空闲的对数几率。在这种情况下，网络作用于对数奇数值l_{ijk}

2) Density grid 密度网格

在该模型中，假设每个体素具有连续的密度，与体素阻塞传感器光束的概率相对应。一般用这个模型作为网络的输入。其中我们跟踪 Beta 参数α_{ijk}^{t}和β_{ijk}^{t}，对于所有(i,j,k)具有统一的先验$\alpha_{ijk}^{0}=\beta_{ijk}^{0}=1$。受测量$z^{t}$影响的每个体素的更新为：

$$\alpha_{ijk}^{t}=\alpha_{ijk}^{t-1}+z^{t}$$
$$\beta_{ijk}^{t}=\beta_{ijk}^{t-1}+1-z^{t}$$

并且(i,j,k)处单元格的后验平均值是

$$u_{ijk}^{t}=\frac{\alpha_{ijk}^{t-1}}{\alpha_{ijk}^{t-1}+\beta_{ijk}^{t}} \tag{2}$$

在这种情况下，我们使用μ_{ijk}作为网络的输入

3) Hit grid 命中网格

该模型只考虑命中，忽略了未知空间和自由空间的区别。每个体素都有一个初始值$h_{ijk}^{t}=0$ 并更新为：

$$h_{ijk}^{t}=minh_{ijk}^{t}+z^{t},1 \tag{3}$$

虽然这个模型丢弃了一些潜在的有价值的信息，但是在 VoxNet 中，它表现得出奇的好。此外，它不需要光线跟踪，这在计算受限的情况下是有用的。

2.1.4. 3D Convolutional Network Layers 3D CNN 模型

CNN 的三个原因：首先，它们可以明确地利用问题的空间结构。特别是，他们可以学习对分类任务有用的局部空间过滤器。在我们的例子中，我们期望输入级的过滤器在不同的方向上编码空间结构，比如平面和角。其次，通过多层叠加，网络可以构建一个由表示更大空间区域的更复杂特征组成的层次结构，最终为输入占用网格创建一个全局标签。最后，推理是纯前馈的，可以有效地在图形硬件上执行。

在本文中，我们考虑由以下类型的层组成的 CNN，如图 2.1 所示。

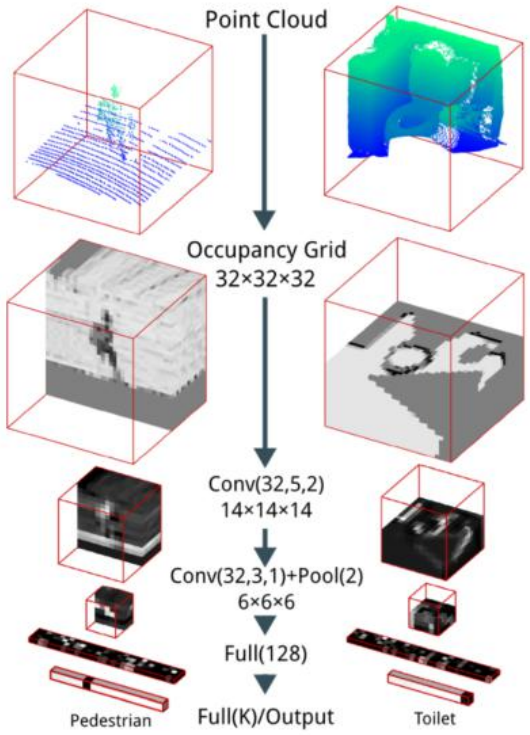


图 2.1 VoxNet 架构. $Conv(f,d,s)$表示大小为 d 且步幅为 s 的 f 个过滤器，$Pool(m)$表示区域为 m 的池化， $Full(n)$ 表示具有 n 个输出的全连接层。展示了实验中两个实例的输入、示例特征图和预测输出。左侧的点云来自 LiDAR，是悉尼城市对象数据集的一部分。右侧的点云来自 RGBD，是 NYUv2 的一部分。我们使用横截面进行可视化

1）输入层：这一层接受一个固定大小的 I×J×K 体素网格。在这项工作中，我们使用 I = J = K = 32。每个网格单元格的每个值都根据占用模型从相应的方程更新。在这三种情况下，我们都要减去 0.5，再乘以 2，所以输入是在$(-1,1)$范围内;不做进一步的预处理。虽然这项工作只考虑标量值输入，但我们的实现可以简单地接受每个单元的额外值，如激光雷达强度值或来自相机的 RGB 信息。

2）卷积层：这些层接受四维输入体积，其中三个维度是空间的，第四个维度包含特征映射。该层将输入与f个形状为 d × d × d × f′的学习滤波器进行卷积，生成f个特征映射，其中d为空间维度，f'为输入特征映射的个数。卷积也可以应用于空间步长s。输出通过参数为 0.1 的有泄漏的整流非线性单元(ReLU)。

3) 池化层： 这些层通过将每个$m \times m \times m$非重叠体素块替换为它们的最大值，沿空间维度以 m 倍的因子对输入体积进行下采样。

4) 全连接层： 全连接层有 n 个输出神经元。每个神经元的输出是前一层所有输出的一个学习线性组合，通过一个非线性。我们使用 ReLU 作用于最终的输出层，其中输出的数量对应于类标签的数量，并使用 softmax 非线性来提供概率输出。

2.1.5. 建议的架构

基本 VoxNet 模型是$C(32,5,2) - C(32,3,1) - P(2) - FC(128) - FC(K)$，其中 K是数字类

2.2. 实现

2.2.1. 数据准备(数据集：ModelNet40)

✧ 解压数据集

✧ 点云数据转换为体素数据

✧ 生成数据列表

✧ 构建数据读取器

✧ 数据查看检验

2.2.2. 构建网络

基本 VoxNet 模型是$C(32,5,2) - C(32,3,1) - P(2) - FC(128) - FC(K)$，其中 K 是数字类

2.2.3. 训练评估

2.2.4. 预测

3. VV-Net 方法

VV-Net[3]是一种基于群卷积的新型体素 VAE 网络，并将其应用于点云分割。此方法对于将对象分割成部分以及将 3D 场景分割成单独的语义对象非常有用。在包括 ShapeNet 和 S3DIS 在内的标准点云数据集上对其性能进行了评估和比较。

- ✧ 开发了一种新的基于信息丰富的体素的点云数据表示。使用以子体素级别的 RBF 为输入的变分自动编码器来捕获每个体素内的点分布。这既提供了规则结构的好处，又为学习算法捕获了详细的分布。
- ✧ 引入了定义在三维数据上的群卷积，在不增加参数数目的情况下，对网络的对称性进行了编码，增加了网络的表达能力。

网络结构如图 3.2 所示。

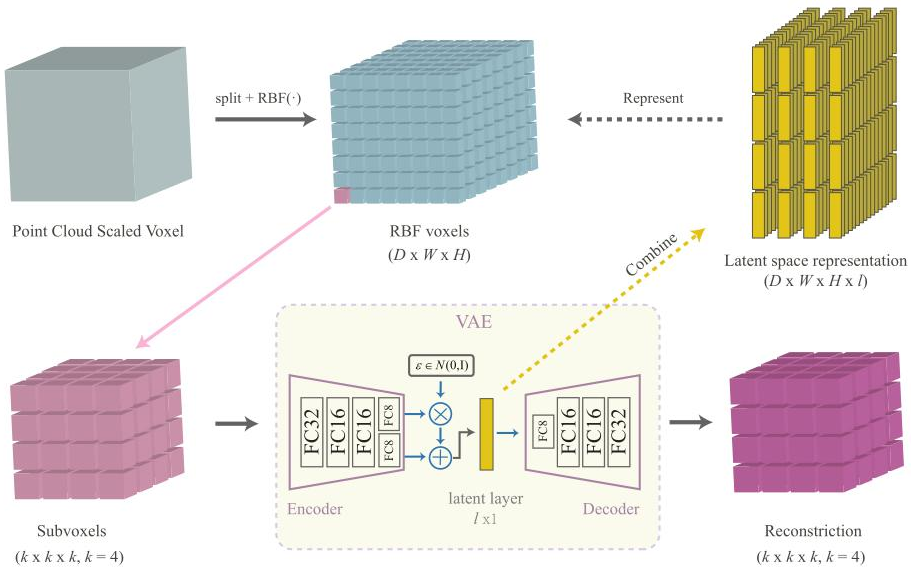


图 3.1 径向基函数插值变分自动编码器模块

对于给定的点云，我们将其划分为等间距的 D×H×W 体素，对于每个体素，我们进一步将其划分为 k×k×k 子体素，其中每个子体素值由径向基函数$f(p) = \max_{v \in V} \left(\exp \frac{-|p-v|_2^2}{2\sigma^2} \right)$定义。根据 VAE 潜在分布，将径向基函数核设置为$\varphi(\|\cdot\|)_2^2$。对于具有 k×k×k 个子体素的体素，我们使用预先训练的变分自动编码器来推断潜在空间表示。最后，点云可以表示为 D×H×W×l 的体素数据，这里 l 表示的是潜在空间的维度。

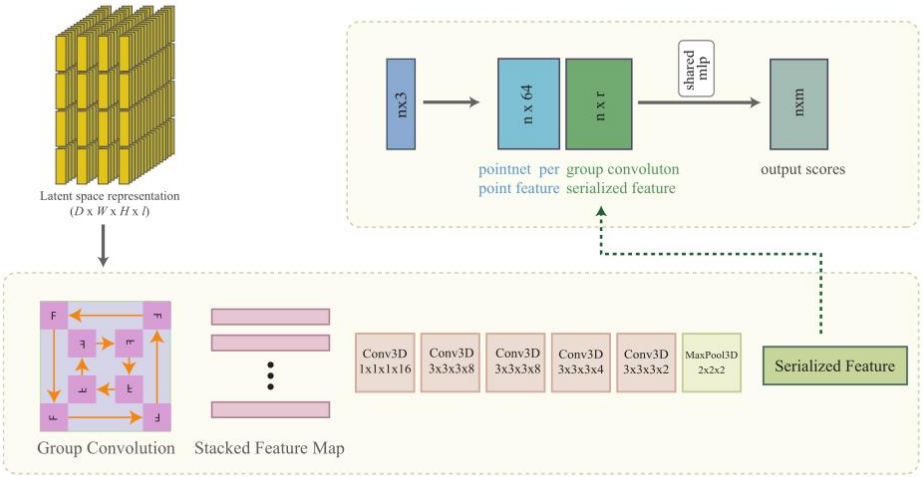


图 3.2 分段网络体系结构

网络的输入是包含多个点的点云，潜在空间表示如图 3.1 所示。输出是点云中每个点的每个类的分数(m 类)。我们使用群卷积模块来检测特征空间中的共现(见公式 4)。

3.2. 相关工作

3.2.1. 与其他方法的区别

✧ 对对称信息进行编码，而不是对邻域信息进行编码。

✧ 不是通过进行抽样来消除不平衡分布，而是使用规则体素和径向基函数来提高学习能力。

3.2.2. 定义在群、等差和变换上的卷积

3.3. 具有群卷积的 VV-Net

首先基于径向基核的 VAE 算法具有对称性和正定性等优点，适用于任意数据位置的选择。使用更好的 encoder-decoder scheme 代替标准的{0,1}体素占用。其次，我们描述了基于Z^3上的群卷积及其同构集来检测潜在空间中特征共现的数学框架，这在不增加参数数量和层数的情况下增加了 CNN 的表达能力。第三，将 MLP 函数提取的 n×64 个每点特征与我们的网络提取的串行化特征连接起来，其中 n 是点数，64 是使用 PointNet 提取的特征的维数。最

后，在 MLP 层之后，我们输出分数图，该分数图指示一个点属于图 3.2 右上角的 m 个类的概率，其中 m 是分割任务中的类数。

3.3.1. Symbols and Notation

在我们的体系结构中，输入是一个点云，使用欧几里得空间中的 3D 坐标(x, y, z)表示。我们使用符号$(\tilde{x}, \tilde{y}, \tilde{z})$来表示体素网格的坐标. 特别地，对于具有 n 个点的给定点云，其包含分别在 Z, Y,轴上具有范围$\widetilde{D}$，$\widetilde{H}$和$\widetilde{W}$的 3D 空间，我们将整个点云划分为$D \times H \times W$体素。我们的 RBF-VAE 方案的输出是一个(D, H, W, l)大小的矩阵，其中l表示编解码器设置的潜在空间维度。

3.3.2. RBF-VAE Scheme

将 p 处的网格值评估为径向基函数的线性组合，而不是布尔占用信息：

$$f(p) = \sum_{j=1}^{N} w_j \varphi\left(\left|\left|p - v_j\right|\right|_2^2\right) \quad (1)$$

其中N是数据点的数目，w_j是标量值，$\varphi(\cdot)$是关于每个数据点的对称函数，并且根据 Bochner 定理是正定的。使用 variational auto-encoder 测量$k \times k \times k$亚体素上的点分布，得到每个体素的 l 维潜在空间，该空间不仅紧凑，而且捕捉到点的空间分布。总体而言，整个点云的体素表示大小为$D \times H \times W \times l$，这比标准的$D \times H \times W$体积表示更详细。

3.3.2.1. RBF

为了将离散点映射到连续分布，我们使用径向基函数来估计它们在每个子体素中的贡献：

$$f(p) = \max_{v \in V}\left(\exp\frac{-|p-v|_2^2}{2\sigma^2}\right) \quad (2)$$

这里，V表示点集，p表示子体素的中心，σ是预定义参数，通常是子体素大小的倍数。

3.3.2.2. Variational Auto-Encoder

编码器的目标是将数据点$X_{(D_i,\ H_i,\ W_i)}$的后验分布映射到潜在向量$Z_{(D_i,\ H_i,\ W_i)}$，其中$(\mathrm{D_i},\ H_i,\ W_i)$表示$k \times k \times k$个子体素，表示为$K_i$。并且解码器从潜在向量$Z_{K_i}$产生可信的对应数据点$\widehat{X_{K_i}}$。在我们的设置中，数据点$X_{K_i}$由公式(2)中所示的 RBF 核子体素表示。我们模型的总损失函数可以评估为：

$$\begin{aligned}\mathrm{Loss} = \sum_{K_i \in (D,H,W)} & E_{Z_{K_i}}\left[\log P\left(X_{K_i}^{(i)} \middle| Z_{K_i}\right)\right] n \\ & -D_{KL}\left(q_\phi\left(Z_{K_i} \middle| X_{K_i}^{(i)}\right) \middle\| P_\theta\left(Z_{K_i}\right)\right) \\ & +D_{KL}\left(q_\phi\left(Z_{K_i} \middle| X_{K_i}^{(i)}\right) \middle\| P_\theta\left(Z_{K_i} \middle| X_{K_i}^{(i)}\right)\right)\end{aligned} \tag{3}$$

其中，我们从$Z_{K_i}|X_{K_i} \sim N(\mu_\{Z_{K_i}|X_{K_i}\}, \sum_\{Z_{K_i}|X_{K_i}\})$采样$Z_{K_i}|X_{K_i}$，从$X_{K_i}|Z_{K_i} \sim N(\mu_\{X_{K_i}|Z_{K_i}\}, \sum_\{X_{K_i}|Z_{K_i}\})$采样$X_{K_i}|Z_{K_i}$，$q_\varphi(Z_{K_i}|X_{K_i})$表示编码器网络，$p_\{\theta\}(X_{K_i}|Z_{K_i})$表示解码器网络。注意，通过变分自动编码器方案，潜在变量Z_{K_i}仅捕获单个体素内的空间信息。对于预先训练的 VAE 模块，我们从固定参数的 VAE 中推断出每个体素，并计算出大小为$D \times H \times W \times l$的最终点云表示，其中$l$是预先训练的 VAE 模块的潜在空间大小。VAE 以更紧凑的方式捕捉体素内的点数据分布。这不仅减少了内存占用，而且使我们的学习算法更加高效。由于先验分布假设，VAE 比 AE 具有更好的泛化能力，避免了对训练集的潜在过拟合。

3.3.3. 对称群与等变表示

在这一节中，我们给出了利用对称群计算等变表示的算法。我们的目标是建立在基于 VAE 的体素表示的基础上，并在 CNN 中检测特征与过滤器的共现。最终目标是在不增加标准 CNN 的层数或过滤器大小的情况下，增强网络的表达能力。

3.3.3.1. Group p4m

Group p4m 由所有的变换合成、围绕网格中任何旋转中心旋转 90◦以及镜像反射组成。我们可以将整数的 group p4 项的(mx，my，mz，rx，ry，rz，tx，ty，tz)参数化为 Rmx×Rmy×Rmz×T，其中 Rmx 公式如下

$$Rmx = \begin{bmatrix} (-1)^{mx}\cos\left(rx\frac{\pi}{2}\right) & -(-1)^{mx}\sin\left(rx\frac{\pi}{2}\right) & 0 & 0 \\ \sin\left(rx\frac{\pi}{2}\right) & \cos\left(rx\frac{\pi}{2}\right) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

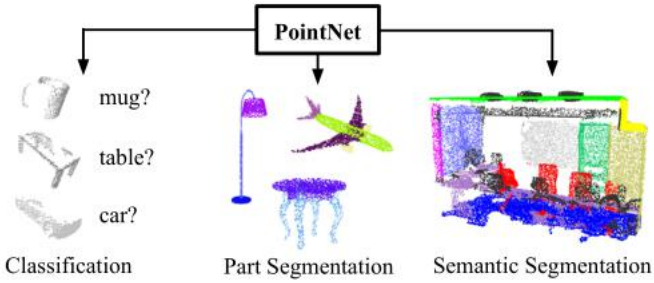
直接基于点云的方法

1. PointNet

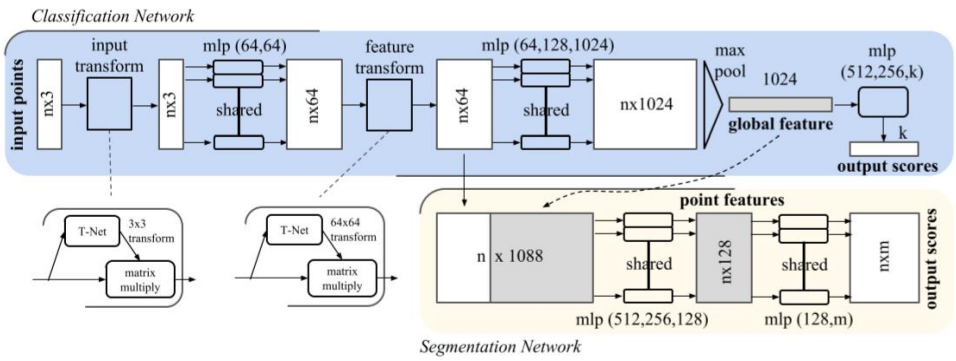
用深度学习方法在处理点云时，往往将其转换为特定视角下的深度图像或者体素(Voxel)等更为规整的格式以便于定义权重共享的卷积操作等。PointNet[4]则允许我们直接输入点云进行处理。

1.1. 输入与输出

输入为三通道点云数据(x_i, y_i, z_i)，也可以有额外的通道比如颜色、法向量等，输出整体的类别/每个点所处的部分/每个点的类别。对于目标分类任务，输出为 k 个分数，分别对应 k 个可能的类别。对于语义分割任务，输出$n \times m$个分数，分别对应 n 个点相对于 m 个类别的分数。



1.2. 网络结构



如图所示，**分类网络**对于输入的点云进行输入变换（input transform）和特征变换（feature transform），随后通过最大池化将特征整合在一起。**分割网络**则是分类网络的延伸，其将整体和局部特征连接在一起出入每个点的分数。第

一次输入变换对空间中点云进行调整，直观上理解是旋转出一个更有利于分类或分割的角度，比如把物体转到正面；第二次特征变换是对提取出的 64 维特征进行对齐，即在特征层面对点云进行变换。图片中"mlp"代表"multi-layer perceptron"（多层感知机）。其中，mlp 是通过共享权重的卷积实现的，第一层卷积核大小是 1x3（因为每个点的维度是 xyz），之后的每一层卷积核大小都是 1x1。即特征提取层只是把每个点连接起来而已。经过两个空间变换网络和两个 mlp 之后，对每一个点提取 1024 维特征，经过 maxpool 变成 1x1024 的全局特征。再经过一个 mlp 得到 k 个 score。这里 T−Net 是一个微型网络，用于生成一个仿射变换矩阵来对点云的旋转、平移等变化进行规范化处理。分类网络最后接的 loss 是 softmax。

关键流程：

i. 输入为一帧的全部点云数据的集合，表示为一个 nx3 的 2d tensor，其中 n 代表点云数量，3 对应 xyz 坐标。

ii. 输入数据先通过和一个 T-Net 学习到的转换矩阵相乘来对齐，保证了模型的对特定空间转换的不变性。

iii. 通过多次 mlp 对各点云数据进行特征提取后，再用一个 T-Net 对特征进行对齐。

iv. 在特征的各个维度上执行 maxpooling 操作来得到最终的全局特征。

v. 对分类任务，将全局特征通过 mlp 来预测最后的分类分数；对分割任务，将全局特征和之前学习到的各点云的局部特征进行串联，再通过 mlp 得到每个数据点的分类结果。

2. PointNet++

PointNet 提取特征的方式是对所有点云数据提取了一个全局的特征，显然，这和目前流行的 CNN 逐层提取局部特征的方式不一样。受到 CNN 的启发，作者提出了 PointNet++[5]，它能够在不同尺度提取局部特征，通过多层网络结构得到深层特征。

2.1. 构成

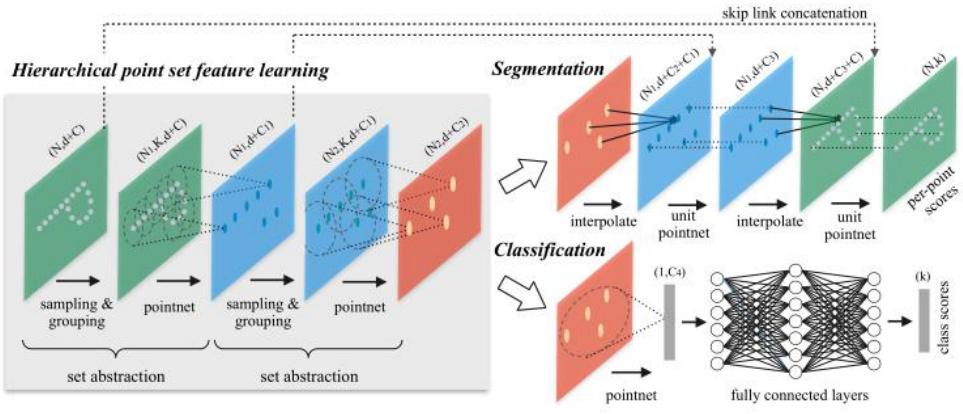
PointNet++由以下几个关键部分构成：

采样层：激光雷达单帧的数据点可以多达 100k 个，如果对每一个点都提取局部特征，计算量是非常巨大的。因此，作者提出了先对数据点进行采样。作者使用的采样算法是最远点采样（farthest point sampling, FPS），相对于随机采样，这种采样算法能够更好地覆盖整个采样空间。

组合层：为了提取一个点的局部特征，首先需要定义这个点的“局部”是什么。一个图片像素点的局部是其周围一定曼哈顿距离下的像素点，通常由卷积层的卷积核大小确定。同理，点云数据中的一个点的局部由其周围给定半径划出的球形空间内的其他点构成。组合层的作用就是找出通过采样层后的每一个点的所有构成其局部的点，以方便后续对每个局部提取特征。

特征提取层：因为 PointNet 给出了一个基于点云数据的特征提取网络，因此可以用 PointNet 对组合层给出的各个局部进行特征提取来得到局部特征。值得注意的是，虽然组合层给出的各个局部可能由不同数量的点构成，但是通过 PointNet 后都能得到维度一致的特征（由上述 K 值决定）。

上述各层构成了 PointNet++的基础处理模块。如果将多个这样的处理模块级联组合起来，PointNet++就能像 CNN 一样从浅层特征得到深层语义特征。对于分割任务的网络，还需要将下采样后的特征进行上采样，使得原始点云中的每个点都有对应的特征。这个上采样的过程通过最近的 k 个临近点进行插值计算得到。完整的 PointNet++的网络示意图如下图所示。



如图，每一个 set abstraction 包括上述三个层，提取层的输入是$N*(d+C)$，其中N是输入点的数量，d是坐标维度，C是特征维度。输出是$N_1*(d+C_1)$，其中N_1是输出点的数量，d是坐标维度不变，C_1是新的特征维度。

分类网络比较简单，对于经过两次 SA 后得到的特征图经过一个 PointNet 提取全局特征然后通过全连接网络得到分类结果。

分割网络较为复杂，需要获得所有原始点的点特征，作者采用基于距离插值的分层传播策略和跨层跳跃链接来实现。在某一层的特征传播过程中，从$N_l * (d + C_1)$向N_{l-1}个点传播特性，这里N_{l-1}和N_l是点集抽象层l的输入和输出的点集数量，并且$N_l < N_{l-1}$。 这里大概可以这么理解，输入 xyz 和 feature 经过一个 SA 后得到了输出 new_xyz 和 new_feature。在上采样过程(FP)中，要将得到的 new_xyz 和 new_feature 再反过来加在输入上，也就是下图所示的$N_1 \times (d + C_2 + C_1) \rightarrow unit\ pointnet \rightarrow N_1 \times (d + C_3)$。文章中通过 k 近邻法(KNN，默认 p=2,k=3)来反向加权求平均实现特征传播，具体公式如下：

$$f^{(j)}(x) = \frac{\sum_{i=1}^{k} w_i(x) f_i^{(j)}}{\sum_{i=1}^{k} w_i(x)} \quad \text{where} \quad w_i(x) = \frac{1}{d(x, x_i)^p},\ j = 1, ..., C$$

简单来说就是距离越远的点权重越小，下面给出 Upsampling 操作的计算过程，由几个 FP 子网络构成。 FP1：输入 SA2.new_xzy, SA2.feature, SA1.new_xzy, SA1.feature，输出 feature。对于 SA1.new_xzy 中每个点，寻找 SA2.new_xzy 中最近的 3 个点；对于这 3 个点，记录 id，计算距离，然后通过距离的倒数计算 3 个点各自的权重； 对这 3 个点的特征进行加权平均求取 SA1.new_xyz 中对应点的 feaure; 得到的 feature 与 SA1.feature 进行 concatenate 操作;通过 MLP 和 MaxPooling 得到 FP1 的输出 feature，SA1.new_xyz 中每个点都对应了一个新的 feature。

2.2. 不均匀点云数据的特征提取

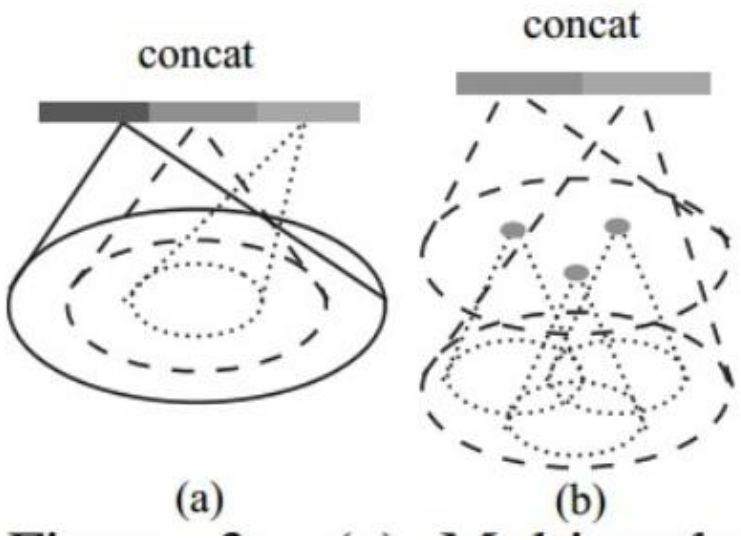
虽然 PointNet 能够用于对各个点云局部提取特征，但是由于点云在各个局部均匀性不一致，很可能导致学习到的 PointNet 不能提取到很好的局部特征。比如说，在越远的地方激光雷达数据通常变得越稀疏，因此在稀疏的地方应该考虑更大的尺度范围来提取特征。为此，作者提出了两种组合策略来保证更优的特征提取。

多尺度组合（multi-scale grouping, MSG）：

比较直接的想法是对不同尺度的局部提取特征并将它们串联在一起，如下图(a)所示。对于同一个中心点，如果使用 3 个不同尺度的话，就分别找围绕每个中心点画 3 个区域，每个区域的半径及里面的点的个数不同。对于同一个中心点来说，不同尺度的区域送入不同的 PointNet 进行特征提取，之后 concat，作为这个中心点的特征。但是因为需要对每个局部的每个尺度提取特征，其计算量的增加也是很显著的。

多分辨率组合（multi-resolution grouping, MRG）：

为了解决 MSG 计算量太大的问题，作者提出了 MRG。此种方法在某一层对每个局部提取到的特征由两个向量串联构成，如下图(b)所示。第一部分由其前一层提取到的特征再次通过特征提取网络得到，第二部分则通过直接对这个局部对应的原始点云数据中的所有点进行特征提取得到。



References:

[1]. 吴晓军, 刘伟军与王天然, 基于八叉树的三维网格模型体素化方法. 工程图学学报, 2005(04): 第1-7页.

[2]. Maturana, D. and S. Scherer, VoxNet: A 3D Convolutional Neural Network for real-time object recognition. 2015: Hamburg, Germany. p. 922-928.

[3]. Meng, H., et al., VV-Net: Voxel VAE Net With Group Convolutions for Point Cloud Segmentation, in 2019 IEEE/CVF International Conference on Computer Vision. 2019. p. 8499-8507.

[4]. Qi, C.R., et al., PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, in 30th IEEE Conference on Computer Vision and Pattern Recognition. 2016.

[5]. Qi, C.R., et al., PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space, in 31th Annual Conference on Neural Information Processing Systems, NIPS 2017. p. 5100-5109.