软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2022,33(11):4356−4378 [doi: 10.13328/j.cnki.jos.006315]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于旋转不变深度层次聚类网络的点云分析*

李冠彬, 张锐斐, 陈 超, 林 倞

(中山大学 计算机学院, 广东 广州 510006)

通信作者: 林倞, E-mail: linliang@ieee.org

摘 要: 由于解决了三维点云的排列不变性问题, 基于三维点云的深度学习方法在计算机三维视觉领域中取得了重大的突破, 人们逐渐倾向于使用三维点云来描述物体并基于神经网络结构来提取点云的特征. 然而, 现有的方法依然无法解决旋转不变性问题, 使得目前的模型鲁棒性较差; 同时, 神经网络结构的设计过于启发式, 没有合理利用三维点云的几何结构与分布特性, 导致网络结构的表达能力有待提升. 鉴于此, 提出了一种具有良好兼容性的严格旋转不变性表达以及深度层次类簇网络, 试图从理论与实践两个层面解决上述问题. 在点云识别、部件分割、语义分割这 3 个经典任务上进行了旋转鲁棒性对比实验, 均取得了最优的效果.

关键词: 三维点云; 旋转不变性; 层次类簇网络; 点云分类; 点云语义分割

中图法分类号: TP391

中文引用格式: 李冠彬, 张锐斐, 陈超, 林倞. 基于旋转不变深度层次聚类网络的点云分析. 软件学报, 2022, 33(11): 4356–4378. http://www.jos.org.cn/1000-9825/6315.htm

英文引用格式: Li GB, Zhang RF, Chen C, Lin L. Rotation-invariant Deep Hierarchical Cluster Network for Point Cloud Analysis. Ruan Jian Xue Bao/Journal of Software, 2022, 33(11): 4356−4378 (in Chinese). http://www.jos.org.cn/1000-9825/6315.htm

Rotation-invariant Deep Hierarchical Cluster Network for Point Cloud Analysis

LI Guan-Bin, ZHANG Rui-Fei, CHEN Chao, LIN Liang

(School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510006, China)

Abstract: In recent years, since the solution of permutation invariance of point cloud, point cloud based deep learning methods have achieved great breakthrough. Point cloud is adopted as input data to describe 3D objects and then neural network is employed to extract features from the point cloud. However, the existing methods cannot solve the rotation-invariance problem, thus existing models are of poor robustness against rotation. Meanwhile, the existing methods merely design the hierarchical structure of neural network by prior knowledge and none of them have made effort to explore the geometric structure underlying the point cloud, which is prone to cause lower capacity of network. For these reasons, a point cloud representation with rotation-invariance and a hierarchical cluster network are proposed, attempting to solve the above two problems in both theoretical and practical ways. Extensive experiments have shown that the proposed method greatly outperforms the state-of-the-arts in rotation robustness on rotation-augmented 3D object classification, object part segmentation, object semantic segmentation benchmarks.

Key words: 3D point cloud; rotation invariance; hierarchical cluster network; point cloud classification; point cloud semantic segmentation

随着无人驾驶、无人机、移动机器人等行业的蓬勃发展, 计算机三维视觉逐渐受到学术界与工业界的广泛关注. 作为一个非常庞大的领域, 其可以划分为底层任务、中层任务、高层任务这 3 个层次.

(1) 底层任务: 包括点云滤波、关键点提取等仅涉及底层信息处理的任务.

(2) 中层任务: 包括特征描述、物体分割等依赖于传统方法对三维物体进行简化、抽象的任务.

* 基金项目: 国家自然科学基金(61976250, 61702565); 广东省基础与应用基础研究基金(2020B1515020048)
收稿时间: 2020-08-06; 修改时间: 2020-10-09, 2020-12-10, 2020-12-26; 采用时间: 2021-01-19

(3) 高层任务: 涉及对三维物体和三维场景进行高层次感知和语义理解, 包括物体识别、物体语义分割、场景语义分割等等. 本文主要关注的是计算机三维视觉领域中的高层任务, 重点研究如何更好地识别三维物体, 感知三维场景.

与图像识别类似, 三维物体识别旨在对单个三维物体进行标签分类, 一般需要对物体提取全局性特征, 最后基于该全局特征进行分类处理; 物体语义分割不仅需要对单个三维物体进行分类, 还要能够从语义层面理解物体的各个部位(例如, 飞机可以划分为机头、机身、机翼、机尾等部位), 这就不仅仅需要全局特征, 还需要学习三维物体在不同粒度下的局部特征; 更进一步地, 我们将问题提升至场景语义分割, 需要对包含多个物体的场景进行语义层面的理解, 该问题更加具有挑战性, 同时也更接近于实际应用场景.

由于计算机三维视觉往往应用于无人驾驶、无人机等需要高安全性、高精确度的领域, 现存的方法具有以下局限性.

(1) 对于旋转变换的鲁棒性较差. 目前, 人们已经解决了三维点云的尺度不变性、平移不变性和排列不变性, 但是还有最后一块短板, 即旋转不变性, 尚未在理论和实践层面解决. 同时, 现有的针对点云对抗攻击而提出的防御措施和方法, 包括对抗训练[1−3]、消除离群点[4]、随机采样[5]、加入高斯噪声[5]、量化[5]等, 也只是在特定的对抗攻击如部分点的移动, 删减或增加等条件下提升了模型的鲁棒性和安全性, 对于旋转变换带来的扰动情况尚未进行深入的分析和研究. 我们知道, 三维空间中的旋转变换具有普适性和无穷性, 即每一个三维物体都有无穷种姿态, 我们不可避免地需要应对模型对于旋转变换的鲁棒性问题. 而经过实验表明, 目前的方法在旋转变换的扰动下是脆弱的.

(2) 未充分利用三维点云所蕴含的几何结构、分布特性. 目前的方法基本上采用最远点采样[6]、K 维-树[7]、八叉树[8]等启发式方法来设计层次神经网络的结构, 这种结构相当于引入了人类的先验知识去分析三维点云的结构特性. 然而, 在人类先验知识指导下所设计出的层次结构, 并不一定真实地反映了三维点云的几何结构, 这导致现存的层次网络结构在模型容量和表达能力上还有很大的提升空间.

基于以上的考量, 本文的主要工作将试图在两个富有挑战性的方向展开充分的研究: 1) 本文将用数学语言去定义严格旋转不变性映射、K 元算子以及映射中的信息损失, 然后运用这一套语言在理论层面上解决三维点云的旋转不变性问题, 并且研究如何应用于现有的模型, 使其能从根本上具有旋转不变性, 从而满足实际应用中所需要的高安全性需求; 2) 本文将探索如何通过机器学习的方法去学习三维点云所蕴含的几何结构, 然后研究如何利用学习到的点云分布特性去自动地指导层次神经网络结构的设计, 提升模型的容量与表达能力, 从而满足实际应用中所需要的高精度需求.

1 三维点云的严格旋转不变性表达

为了更加准确地描述严格旋转不变性, 我们用数学语言定义了一种严格旋转不变性映射.

定义 1 (**严格旋转不变性映射**). 给定 N, $D\in\mathbf{N}^+$, 如果一个点集映射$\mathcal{F}:\mathbb{R}^{N\times 3}\mapsto\mathbb{R}^{N\times D}$ 满足: $\mathcal{F}(S)=\mathcal{F}(R(S))$对于任意点集 $S\in\mathbb{R}^{N\times 3}$、任意旋转变换 $R\in SO(3)$均成立, 则$\mathcal{F}$是严格旋转不变性映射, 且$\mathcal{F}(S)$是点集 S 的严格旋转不变性表达.

我们可以看到, 严格旋转不变性映射的定义要求映射具有旋转变换上的不变性. 不仅如此, 映射还需要严格保证输入点集与输出点集在基数上的一致性, 即原本由 N 个点组成的输入点集必须要被映射为 N 个点组成的输出点集. 此处正是严格旋转不变性映射的名称中“严格”一词的由来.

基于此定义, 我们会发现: 如果能够找到一种严格旋转不变性映射$\mathcal{F}$, 那么让该映射作用于一个三维点云 $S\in\mathbb{R}^{N\times 3}$, 就得到了三维点云 S 的严格旋转不变性表达. 为了便于我们之后的数学证明, 我们还需要定义一种严格旋转不变性 K 元算子, 去描述旋转变换下若干点之间相对位置关系的不变性.

定义 2 (**严格旋转不变性 K 元算子**). 如果一个 K 元算子 $\mathcal{G}:\underbrace{\mathbb{R}^3\times\mathbb{R}^3\times\ldots\times\mathbb{R}^3}_{K}\mapsto\mathbb{R}^n$ 满足: $\mathcal{G}(R\boldsymbol{x}_1,R\boldsymbol{x}_2,\ldots,R\boldsymbol{x}_K)=$

$\mathcal{G}(\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_K)$对于任意 $\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_K\in\mathbb{R}^3$、任意旋转变换 $R\in SO(3)$均成立, 则 K 元算子$\mathcal{G}$是严格旋转不变性 K 元算子.

我们可以看到, 严格旋转不变性 K 元算子严格定义了 K 个点之间 K 元关系的一种不变性, 它不受到旋转变换的影响. 基于严格旋转不变性 K 元算子的定义, 我们可以得到如下引理.

引理 1.

(1) 向量 L_2 范数$|\cdot|_2:\mathbb{R}^3\to\mathbb{R}$, 是严格旋转不变性一元算子.

(2) 向量内积$\cdot,\cdot:\mathbb{R}^3\times\mathbb{R}^3\to\mathbb{R}$, 是严格旋转不变性二元算子.

(3) 给定任意非零向量 $\boldsymbol{p}\in\mathbb{R}^3\backslash\{0\}$以及正交投影算子$\mathcal{T}_p(\cdot):\mathbb{R}^3\to\mathbb{R}^2$, 其中, 正交投影的像空间为过原点的平面 L, 且 $\boldsymbol{p}\perp L$, 则复合算子$\mathcal{G}_1(\boldsymbol{x},\boldsymbol{y},\boldsymbol{p})=\langle\mathcal{T}_p(\boldsymbol{x}),\mathcal{T}_p(\boldsymbol{y})\rangle$和$\mathcal{G}_2(\boldsymbol{x},\boldsymbol{y},\boldsymbol{p})=\langle\mathcal{T}_p(\boldsymbol{x})\times\mathcal{T}_p(\boldsymbol{y}),\boldsymbol{p}\rangle$均为严格旋转不变性三元算子.

引理 1 告诉我们, 存在着 4 种严格旋转不变性算子, 它们分别是向量 L_2 范数、向量内积、$\mathcal{G}_1(\boldsymbol{x},\boldsymbol{y},\boldsymbol{p})=\langle\mathcal{T}_p(\boldsymbol{x}),\mathcal{T}_p(\boldsymbol{y})\rangle$以及$\mathcal{G}_2(\boldsymbol{x},\boldsymbol{y},\boldsymbol{p})=\langle\mathcal{T}_p(\boldsymbol{x})\times\mathcal{T}_p(\boldsymbol{y}),\boldsymbol{p}\rangle$.

基于这 4 种严格旋转不变性算子, 我们可以构建出本文所提出的关于三维点云的严格旋转不变性表达, 而计算的过程则为严格旋转不变性映射. 具体而言, 由于三维点云只包含每个孤立的坐标点的信息, 我们首先要对三维点云进行建图. 为了便于将图用张量形式表示, 并且控制每个点的邻居数相同, 本文采用了 K-近邻图.

我们首先在三维点云 S 上建立 K-近邻图 $G=(S,\mathcal{E})$, 其中, $\mathcal{E}=\{(x,y)\in S\times S|y\in K\text{-}NN(x)\}$. 对于任意的三维点云 $S=\{p_i\in\mathbb{R}^3|1\leqslant i\leqslant N\}$, 我们可以计算其对应的严格旋转表达性表达, 其具体形式为

$$\{(r_i,(r_{i1},\theta_{i1},\phi_{i1}),(r_{i2},\theta_{i2},\phi_{i2}),\ldots,(r_{iK},\theta_{iK},\phi_{iK}))\boldsymbol{p}_i\in S\} \tag{1}$$

其中, 每个变量均为满足旋转不变性的统计量, 其具体的解析表达式为

$$\left.\begin{aligned}
&r_i=\|\boldsymbol{p}_i\|_2\\
&r_{ik}=\|\boldsymbol{p}_{ik}\|_2,\ ((\boldsymbol{p}_{ik}\in K\text{-}NN(p_i),\text{其下标为}k)\\
&\theta_{ik}=\arccos\left(\left\langle\frac{\boldsymbol{p}_i}{r_i},\frac{\boldsymbol{p}_{ik}}{r_{ik}}\right\rangle\right)\\
&\phi_{ik}=\psi_{j^*}\triangleq\min\{\psi_j|1\leqslant j\leqslant K,j\neq k\}\\
&\psi_j=\begin{cases}\text{a}\tan 2(\sin\psi_j,\cos\psi_j), & \text{if a}\tan 2(\sin\psi_j,\cos\psi_j)\geqslant 0\\ 2\pi+\text{a}\tan 2(\sin\psi_j,\cos\psi_j), & \text{if a}\tan 2(\sin\psi_j,\cos\psi_j)<0\end{cases}\\
&\sin\psi_j=\left\langle\frac{\mathcal{T}_{p_i}(\boldsymbol{p}_{ik})}{\|\mathcal{T}_{p_i}(\boldsymbol{p}_{ik})\|_2}\times\frac{\mathcal{T}_{p_i}(\boldsymbol{p}_{ij})}{\|\mathcal{T}_{p_i}(\boldsymbol{p}_{ij})\|_2},\frac{\boldsymbol{p}_i}{r_i}\right\rangle\\
&\cos\psi_j=\left\langle\frac{\mathcal{T}_{p_i}(\boldsymbol{p}_{ik})}{\|\mathcal{T}_{p_i}(\boldsymbol{p}_{ik})\|_2},\frac{\mathcal{T}_{p_i}(\boldsymbol{p}_{ij})}{\|\mathcal{T}_{p_i}(\boldsymbol{p}_{ij})\|_2}\right\rangle
\end{aligned}\right\} \tag{2}$$

观察上述解析表达式(2), 我们会发现这些统计量均通过引理 1 中的 4 种严格旋转不变性算子计算得出; 同时, 我们提出的严格旋转不变性表达(式(1))还具有非常直观的几何意义. 如图 1 所示, 这是一个只包含 3 个点的三维点云, 我们在该点云上建立了 2-近邻图. 在图中, 我们将点描述成向量的形式, 其中标出了向量 $\boldsymbol{p}_i$ (图 1 中的黑色实线箭头)及两个邻居点 $\boldsymbol{p}_{i1}$(图 1 中的红色实线箭头)与 $\boldsymbol{p}_{i2}$ (图 1 中的蓝色实线箭头). 我们可以看到: r_i 和 r_{ik} 的几何意义为向量 $\boldsymbol{p}_i$ 与 $\boldsymbol{p}_{ik}$ 的模长; θ_{ik} 的几何意义为向量 $\boldsymbol{p}_i$ 与 $\boldsymbol{p}_{ik}$ 之间的夹角; 而φ_{i1} 的几何意义为向量 $\boldsymbol{p}_{i1}$ 经过正交投影之后的投影向量 $\mathcal{T}_{p_i}(\boldsymbol{p}_{i1})$ (图 1 中的红色虚线箭头)与逆时针旋转角度最近的投影向量 $\mathcal{T}_{p_i}(\boldsymbol{p}_{i2})$ (图 1 中的蓝色虚线箭头)之间的夹角.

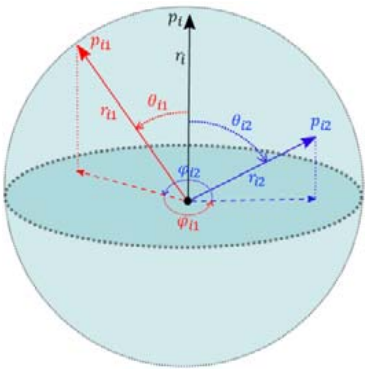


图 1　三维点云的严格旋转不变性表达的示意图

1.1 严格旋转不变性

我们将上文中各个统计量的解析表达式(2)用算法伪代码的形式描述, 如算法 1 所示.

算法 1. 严格旋转不变性映射.

输入: 点集 $S\subset\mathbb{R}^3$, 参数 K.

for $\boldsymbol{p}_i$ in S **do**
  $r_i\leftarrow\|\boldsymbol{p}_i\|_2$
  for $\boldsymbol{p}_{ik}$ in K-$NN(\boldsymbol{p}_i)$ **do**
    $r_{ik}\leftarrow\|\boldsymbol{p}_{ik}\|_2$
    $\theta_{ik}\leftarrow\arccos\left(\left\langle\frac{\boldsymbol{P}_i}{r_i},\frac{\boldsymbol{P}_{ik}}{r_{ik}}\right\rangle\right)$
    $t_{ik}\leftarrow\boldsymbol{T}_{\boldsymbol{P}_i}(\boldsymbol{P}_{ik})$
    for j in $[1,k]$ **do**
      $\sin\psi_j\leftarrow\left\langle\frac{\boldsymbol{t}_{ik}}{\|\boldsymbol{t}_{ik}\|}\times\frac{\boldsymbol{t}_{ij}}{\|\boldsymbol{t}_{ij}\|},\frac{\boldsymbol{p}_i}{r_i}\right\rangle$
      $\cos\psi_j\leftarrow\left\langle\frac{\boldsymbol{t}_{ik}}{\|\boldsymbol{t}_{ik}\|},\frac{\boldsymbol{t}_{ij}}{\|\boldsymbol{t}_{ij}\|}\right\rangle$
      if $\mathrm{atan2}(\sin\psi_j,\cos\psi_j)\geqslant 0$ **then**
        $\psi_j\leftarrow\mathrm{atan2}(\sin\psi_j,\cos\psi_j)$
      else
        $\psi_j\leftarrow 2\pi+\mathrm{atan2}(\sin\psi_j,\cos\psi_j)$
      end if
    end for
    $\phi_{ik}\leftarrow\min\{\psi_j|1\leqslant j\leqslant K,j\neq k\}$
  end for
end for

输出: 严格旋转不变性表达$\{(r_i,(r_{i1},\theta_{i1},\phi_{i1}),\ldots,(r_{iK},\theta_{iK},\phi_{iK}))|\boldsymbol{p}_i\in S\}$.

至此, 我们已经介绍了严格旋转不变性表达(式(1))以及严格旋转不变性映射算法 1. 下面我们以定理的形式声明其满足严格旋转不变性.

定理 1 (严格旋转不变性映射与表达). 给定点集$S\subset\mathbb{R}^3$以及建立在点集S上的K-近邻图$\boldsymbol{G}=(S,\mathcal{E})$, 则公式(1)所定义的表达$\{(r_i,(r_{i1},\theta_{i1},\phi_{i1}),(r_{i2},\theta_{i2},\phi_{i2}),\ldots,(r_{iK},\theta_{iK},\phi_{iK}))\boldsymbol{p}_i\in S\}$为严格旋转不变性表达. 算法 1 所定义的映射算法

是严格旋转不变性映射.

如图 2 所示, 它直观地表示了本文提出的三维点云具有严格旋转不变性不受任意旋转变换 $R\in SO(3)$的影响. 至此, 我们已经以定理的形式证明了我们提出的三维点云表达(式(1))为严格旋转不变性表达, 而算法 1 定义了一种严格旋转不变性映射, 使得我们提出的点云表达在旋转不变性上具有理论保障.

图 2 定理 1 的示意图

1.2 信息无损性

尽管严格旋转不变性表达具有我们迫切需要的旋转不变性, 然而, 当我们把三维点云从原始的笛卡尔坐标转化为严格旋转不变性表达的过程中, 很可能会出现信息损失. 这意味着: 我们在追求严格旋转不变性表达的过程中, 有可能会丢失原始表达中蕴含的关键信息. 这会为我们后续的特征提取带来不可逆的信息丢失, 从而影响下游模型的表现与性能.

为了能够严谨定义严格旋转不变性映射及其表达的有损性与无损性, 我们需要引入“旋转可重建性”的概念. 注意, 这个概念与计算机三维视觉领域中的三维重建有着本质不同. 下面我们给出旋转可重建性的定义.

定义 3 (旋转可重建性). 给定点集 $S\in\mathbb{R}^{N\times 3}$ 以及点集 S 上的映射 $\mathcal{F}:\mathbb{R}^{N\times 3}\to\mathbb{R}^{N\times d}$. 对点云 S 施加一个未知的旋转变换 $R\in SO(3)$后, 我们会得到一个旋转后的未知点云 $R(S)$. 给定旋转后未知点云 $R(S)$中 k 个不共线向量的坐标后($1\leqslant k\leqslant N$), 如果我们可以基于这 k 个不共线向量以及点集 S 的映射结果 $\mathcal{F}(S)$计算出未知点云中剩余 $N-k$ 个点的坐标, 那么映射 $\mathcal{F}$具有 k 点旋转可重建性.

我们会发现: 重建未知点云 $R(S)$时使用的向量数目 k 越小, 那么点集映射 $\mathcal{F}$在映射时所丢失的信息越少, 其所需的额外信息就越少. 所以, 基于旋转可重建性的定义 3, 我们可以采用 k 点旋转可重建性中 k 的大小, 来度量点集映射 $\mathcal{F}$的信息损失量.

为此, 我们需要讨论 k 点旋转可重建性中 k 的有效范围. 当 $k=N$ 时, 点集映射 $\mathcal{F}$完全无法重建未知点云, 此时信息损失最严重. 这种情况是存在的, 例如, 当点集映射 $\mathcal{F}$为常映射(即将所有点均映射为固定的常数)时. 所以, k 的上确界为 $k=N$. 下面我们将以定理的形式表明 k 点旋转可重建性中 k 的下确界为 $k=2$.

定理 2 (k 点旋转可重建性中 k 的下确界). 对于任意具有 k 点旋转可重建性的点集映射 $\mathcal{F}:\mathbb{R}^{N\times 3}\to\mathbb{R}^{N\times d}$, k 的下确界为 $k=2$.

定理 2 符合我们对三维世界的直观感受, 我们可以通过如下朴素的方法来简要证明其正确性. 如果一个三维物体 S 经过一个未知旋转 R, 那么显然旋转后该物体 $R(S)$将会是完全未知的, 因为我们对其姿态一无所知. 现在, 如果我们想要重建出这个未知姿态下的三维物体 $R(S)$, 仅仅借助 $R(S)$中一个点的坐标 x'是肯定不够的. 我们会发现, 仅仅知道原来三维物体 S 中的点 x 旋转后变为 x'无法确定旋转后三维物体 $R(S)$的姿态, 因为此时我们还能够以 x'为欧拉旋转轴、以任意非零角度进行旋转, 其姿态并不能唯一确定. 因此, 对于任意的三维点云, 不存在点集映射 $\mathcal{F}$具有 1 点旋转可重建性, 即 k 点旋转可重建性中的 k 无法取到 $k=1$. 而如果给定旋转后未知点云 $R(S)$中两个不共线的向量 $\boldsymbol{p}_i'$ 和 $\boldsymbol{p}_j'$, 那么我们依靠其中一个向量. 例如 $\boldsymbol{p}_i'$, 基本固定旋转后物体的姿态, 此时它只能以 $\boldsymbol{p}_i'$ 为欧拉旋转轴来改变姿态, 此时, 三维姿态空间被缩小为二维姿态空间, 三维旋转变换的自由度由 3 降至 1. 然后, 我们可以通过剩下的与 $\boldsymbol{p}_i'$ 不共线的向量 $\boldsymbol{p}_j'$, 来确定物体以 $\boldsymbol{p}_i'$ 为欧拉旋转轴的最终姿态. 故, 2 点旋转可重建性在直观上是可行的.

基于定理 2, 我们可以得到如下推论.

推论 1 (三维点云的 2 点旋转可重建性). 假设任意的点云 $S\in\mathbb{R}^{N\times 3}$ 经过未知旋转变换 $R\in SO(3)$后, 得到了未知点云 $R(S)$. 如果给定未知点云 $R(S)$中两个不共线向量 $\boldsymbol{p}_i'$ 与 $\boldsymbol{p}_j'$ (其在原始点云 S 中的对应点分别为 $\boldsymbol{p}_i$ 与 $\boldsymbol{p}_j$), 那么我们可以通过原始点云 S 唯一确定未知点云 $R(S)$中剩下的 N–2 个点.

由定理 2 及其推论 1 可知, 当点集映射$\mathcal{F}$为恒等映射, 即$\mathcal{F}(S)=S$ 时, 最少需要知道未知点云 $R(S)$中的两个不共线的向量, 才能通过映射结果$\mathcal{F}(S)$来重建未知点云 $R(S)$. 由于恒等映射的原像与像完全相等, 显然不存在信息损失. 因此, 如图 3 所示, 如果一个点集映射$\mathcal{G}$也是信息无损的, 那么同理也应该能在给定未知点云 $R(S)$中两个不共线向量的情况下, 通过映射结果$\mathcal{G}(S)$重建未知点云 $R(S)$. 鉴于此, 我们可以正式定义点集映射的信息无损性以及条件信息无损性, 如下所示.

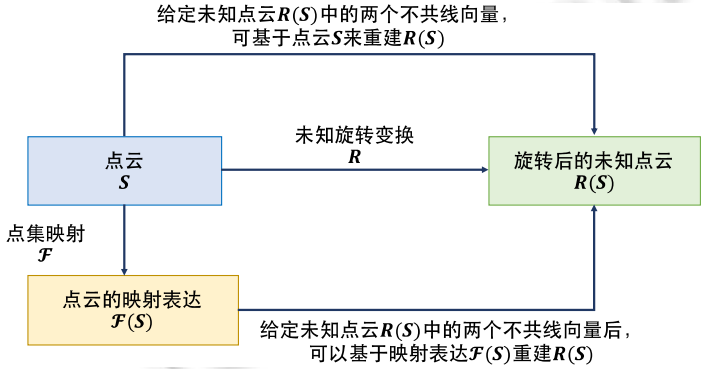


图 3 信息无损性的示意图

定义 4 (信息无损性与条件信息无损性). 如果一个点集映射$\mathcal{F}:\mathbb{R}^{N\times 3}\to\mathbb{R}^{N\times d}$ 具有 2 点旋转可重建性, 即给定未知点云 $R(S)$中两个不共线向量, 原始点云 S 由未知旋转 R 所得到的未知点云 $R(S)$可以通过$\mathcal{F}(S)$重建出来, 则点集映射$\mathcal{F}$具有信息无损性; 如果需要给定未知点云 $R(S)$中更多的向量, 则点集映射$\mathcal{F}$是信息有损的; 而如果给定了未知点云 $R(S)$中两个不共线向量之后, 还对这两个向量有其他约束条件, 则点集映射$\mathcal{F}$具有条件信息无损性.

基于上述信息无损性与条件信息无损性的定义 4, 我们以定理的形式表明: 本文提出的严格旋转不变性映射算法 1 在不考虑存储空间的情况下是满足信息无损性的, 而在一般情况下满足条件信息无损性.

定理 3 (严格旋转不变性映射的信息无损性). 给定点集 $S\subset\mathbb{R}^{N\times 3}$ 以及建立在点集 S 上的 K-近邻图 $\boldsymbol{G}=(S,\mathcal{E})$. 如果 K-近邻图 $\boldsymbol{G}$ 为强连通图, 则算法 1 定义的严格旋转不变性映射: 当 $K=N$ 时, 满足信息无损性; 当 $K<N$ 时, 满足条件信息无损性. 其条件是重建未知点云时, 给定的两个不共线向量 $\boldsymbol{p}_i'$ 与 $\boldsymbol{p}_j'$ 需要满足 K 近邻关系:

$$\boldsymbol{p}_j'\in K\text{-}NN(\boldsymbol{p}_i').$$

定理 3 表明: 本文提出的严格旋转不变性映射以及严格旋转不变性表达在信息损失的角度上是条件信息无损的, 在存储空间不受限制的情况下, 能够实现完全的信息无损性. 这些都为我们所提出的严格旋转不变性表达提供了坚实的理论保障.

1.3 兼容性

在本节中, 我们将详细讨论严格旋转不变性表达(式(1))如何具有兼容性, 从而能广泛应用于目前基于三维点云的深度学习算法, 使得在旋转鲁棒性上性能较差的算法也能具有严格旋转不变性, 在需要高安全性、高鲁棒性的实际应用场景中能发挥自己应有的作用.

首先, 目前大多数基于三维点云的深度学习模型采用的输入数据是大小为 $N\times d$ 的矩阵, 其中, N 为三维点云中点的数目, d 为每个点的维数. 一般情况下, d=3, 即输入数据为 $N\times 3$ 的矩阵, 包含三维空间中 N 个点的笛

卡尔坐标; 当然, 有时 $d>3$, 此时输入数据不仅包含笛卡尔坐标, 还可能包含其他关于点的特征, 例如该点处的颜色、垂直于表面的法向量等等.

第 1.1 节中介绍的严格旋转不变性表达(式(1))是一个大小为 $N\times(K\times3+1)$的张量, 如式(3)所示.

$$\begin{bmatrix} r_1 & (r_{11},\theta_{11},\phi_{11}) & (r_{12},\theta_{12},\phi_{12}) & \cdots & (r_{1K},\theta_{1K},\phi_{1K}) \\ r_2 & (r_{21},\theta_{21},\phi_{21}) & (r_{22},\theta_{22},\phi_{22}) & \cdots & (r_{2K},\theta_{2K},\phi_{2K}) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_N & (r_{N1},\theta_{N1},\phi_{N1}) & (r_{N2},\theta_{N2},\phi_{N2}) & \cdots & (r_{NK},\theta_{NK},\phi_{NK}) \end{bmatrix} \in \mathbb{R}^{N\times(k\times3+1)} \tag{3}$$

其中, 第 i 行表示了三维点云中第 i 个点所对应的旋转不变性表达, 它是一个大小为 $N\times3+1$ 的张量, 其包含该点的模长 r_i 以及 K-近邻点的信息. 第 i 个点的第 i 个近邻点, 其信息用三元组$(r_{ik},\theta_{ik},\phi_{ik})$表示.

为了让严格旋转不变性表达(式(3))具有兼容性, 一种朴素的方法是将其重新整理为 $N\times(3K+1)$的二维矩阵. 而我们采用的是一种更加巧妙的方法: 将三维点云中每个点的 K-近邻域当作子点云进行处理. 由严格旋转不变性表达(式(1))可知, 三维点云中第 i 个点所对应的旋转不变性表达为

$$(r_i,(r_{i1},\theta_{i1},\phi_{i1}),(r_{i2},\theta_{i2},\phi_{i2}),\ldots,(r_{iK},\theta_{iK},\phi_{iK})) \tag{4}$$

它是一个 $N\times3+1$ 的张量, 我们可以将其中包含的变量进行重新组织, 得到如下形式:

$$\underbrace{(r_i,r_{i1},\theta_{i1},\phi_{i1})}_{T_{i1}},\underbrace{(r_i,r_{i2},\theta_{i2},\phi_{i2})}_{T_{i2}},\ldots,\underbrace{(r_i,r_{ik},\theta_{ik},\phi_{ik})}_{T_{ik}} \tag{5}$$

重新组织后, 公式(5)将点 i 的模长分配给点 i 的 K 个近邻, 即放入 K 个三元组之中, 构成了一个 r_i, r_{ik}, θ_{ik}, ϕ_{ik} 组成的四元组. 为了书写的简便性, 我们将这些四元组用 T_{ik} 表示. 于是, 我们就得到了一个 $N\times K\times4$ 的三维张量, 如式(6)所示.

$$\begin{bmatrix} (r_1,r_{11},\theta_{11},\phi_{11}) & (r_1,r_{12},\theta_{12},\phi_{12}) & \cdots & (r_1,r_{1K},\theta_{1K},\phi_{1K}) \\ (r_2,r_{21},\theta_{21},\phi_{21}) & (r_2,r_{22},\theta_{22},\phi_{22}) & \cdots & (r_2,r_{2K},\theta_{2K},\phi_{2K}) \\ \vdots & \vdots & \ddots & \vdots \\ (r_N,r_{N1},\theta_{N1},\phi_{N1}) & (r_N,r_{N2},\theta_{N2},\phi_{N2}) & \cdots & (r_N,r_{NK},\theta_{NK},\phi_{NK}) \end{bmatrix} = \begin{matrix} (T_{11}) & (T_{12}) & \cdots & (T_{1K}) \\ (T_{21}) & (T_{22}) & \cdots & (T_{2K}) \\ \vdots & \vdots & \ddots & \vdots \\ (T_{N1}) & (T_{N2}) & \cdots & (T_{NK}) \end{matrix} \in \mathbb{R}^{N\times k\times4} \tag{6}$$

它通过每个点 $\boldsymbol{p}_i$ 与其 K-近邻点之间的旋转不变量$\{T_{i1},T_{i2},\ldots,T_{iK}\}$来表征 $\boldsymbol{p}_i$. 由于三维点云中每个点 $\boldsymbol{p}_i$ 的邻域特征往往蕴含于点 $\boldsymbol{p}_i$ 的 K-近邻域之中, 严格旋转性表达(式(6))利用了这一特性, 充分挖掘三维点云中每个点的 K-近邻域的特征, 并在此基础上, 还满足旋转不变性与条件信息无损性.

此时, 我们可以引入“子点云”的概念. 具体而言, 原本作为输入数据的是一个三维点云, 而三维点云中每个点的 K-近邻点集$\{T_{i1},T_{i2},\ldots,T_{iK}\}$也可以视为一个“子点云”, 其中每个点 $T_{ik}\in R^4$. 三维点云中每个点所对应的子点云刻画了该点的邻域情况, 包含着重要的邻域信息. 于是, 我们可以采用 PointNet[6], 一种在理论上具有通用近似能力的深度学习网络, 来学习每个点所对应的子点云$\{T_{i1},T_{i2},\ldots,T_{iK}\}$的高维嵌入, 从而基于邻域信息更好地表征每一个点, 如图 4 所示.

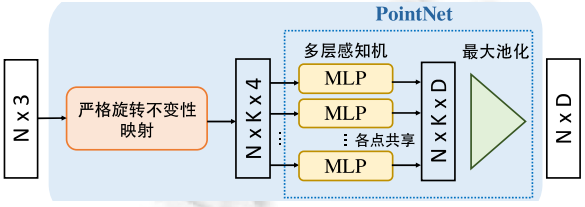


图 4 严格旋转不变性表达的兼容性模块的网络结构图

综上所述, 我们会发现: 原本 $N\times3$ 的三维点云 S, 通过我们提出的严格旋转不变性映射变为 $N\times K\times4$ 的严格旋转不变性表达(式(6)). 然后, 我们通过 PointNet 网络得到 $N\times D$ 兼容性表达. 整个过程相当于是对三维点云中的点 $\boldsymbol{p}_i\in\mathbb{R}^3$, 通过特征提取得到了具有旋转不变性的高维特征 $\boldsymbol{p}_i'\in\mathbb{R}^D$.

以动态图卷积神经网络[9]的角度来看, 如图 4 所示的神经网络结构可以看作是特殊的边卷积 EdgeConv.

然而, 动态图卷积神经网络中的边卷积采用的是中心点 $\boldsymbol{p}_i$ 以及差向量 $\boldsymbol{p}_{ik}-\boldsymbol{p}_i$ 去描述中心点 $\boldsymbol{p}_i$ 的第 k 条边, 其中, $\boldsymbol{p}_{ik}$ 为 $\boldsymbol{p}_i$ 的第 k 近邻点. 我们会发现: 无论是中心点还是差向量, 均不具有旋转不变性. 容易证明, 二者均随着旋转变换而改变. 与之不同的是, 我们采用的是严格旋转不变性表达去描述中心点 $\boldsymbol{p}_i$ 与其第 k 个近邻之间的关系, 即 $\{T_{i1},T_{i2},\ldots,T_{iK}\}$, 这种关系与三维点云的姿态无关, 有利于模型聚焦于近邻点之间更加本质的关系.

1.4 时空复杂度分析

通过分析算法 1 可知, 关于 K-近邻的计算的时间复杂度为 O(Nlog$N$), 关于 $\boldsymbol{r}_i$ 与 $\boldsymbol{r}_{ik}$ 的计算的时间复杂度为 O(N), 关于 $\boldsymbol{\theta}_{ik}$ 的计算的时间复杂度为 O(NK), 关于 $\boldsymbol{\phi}_{ik}$ 的计算的时间复杂度为 O(NKlog$K$). 考虑上述所有变量的计算, 严格旋转不变性映射算法的时间复杂度为 O(Nlog$N$+N+NK+NKlog$K$)=O(N(log$N$+Klog$K$)).

而严格旋转不变性映射算法的空间复杂度分析则较为简单. 我们没有使用任何额外的存储空间, 所有变量均有其对应的解析表达式, 故空间复杂度为 O(1).

2 层次聚类神经网络

2.1 算法整体框架

层次聚类神经网络 ClusterNet 的整体算法框架总共包含两个阶段.

- 第 1 阶段是点云几何结构的学习与层次网络结构的生成. 这一阶段是准备阶段, 不涉及模型参数的学习.
- 第 2 阶段是在第 1 阶段的基础上进行类簇的嵌入表达学习, 即通过深度学习的方式端到端地学习层次神经网络模型中的待学习参数.

层次聚类神经网络的算法整体框架如图 5 所示.

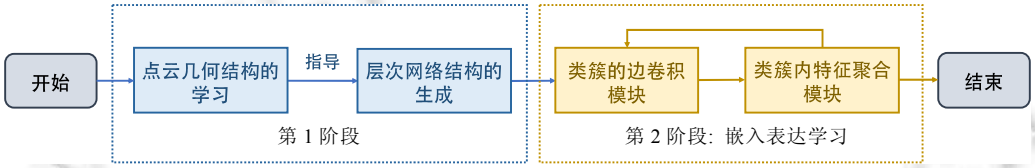


图 5 层次聚类神经网络 ClusterNet 的整体算法框架

第 1 个阶段是对于输入的三维点云 $S\subset\mathbb{R}^{N\times 3}$, 通过层次聚类算法学习三维点云 S 蕴含的几何结构, 即得到一棵层次聚类树. 该层次聚类树记录了类簇不断聚合的过程, 在各个层次下的类簇聚合就相当于是一次聚类, 把相似的类簇划分为同一类, 而把相异的类簇划分为不同类. 于是, 我们可以通过这棵层次聚类树指导层次网络结构的生成, 即让类簇特征的聚合方式与层次聚类树中类簇的聚合方式保持一致. 故在第一阶段中, 我们可以学习点云的几何结构, 并且利用它确定层次网络结构.

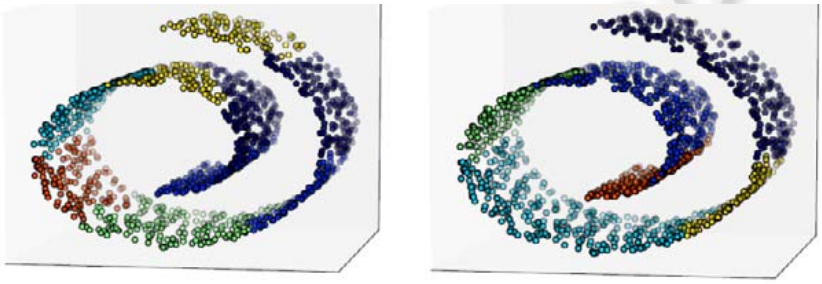
第 2 个阶段是类簇的嵌入表达学习阶段, 在这一阶段中, 我们基于上一阶段确定的层次聚类树与层次网络结构, 自底向上地学习层次聚类树中各个类簇的嵌入表达. 具体来说, 类簇的嵌入表达学习包含类簇的边卷积模块与类簇内特征聚合模块, 其中, 类簇的边卷积模块基于类簇的邻域信息来提取类簇的高维特征; 类簇内特征聚合模块负责将父类簇内的若干个子类簇特征进行聚合, 从而得到父类簇的高维特征. 我们通过不断堆叠类簇的边卷积模块与类簇内特征聚合模块, 从而更好地学习整个三维点云的嵌入表达.

2.2 点云几何结构的无监督学习

由于现存的层次神经网络在层次结构的设计上往往过度依赖于人类的先验知识, 而这种设计方法都过于启发式, 没有充分地挖掘和利用三维点云本身所蕴含的几何结构与分布特性. 针对这一问题, 本文提出采取基于 Ward 链接准则[10,11]的层次聚类算法[12]去学习三维点云本身的层次几何结构. 与其他链接准则不同的是, Ward 链接准则没有简单地定义类簇间距离, 而是采用类簇合并前后的类内方差的增加量作为目标函数, 试图

最小化因为类簇合并而导致的类内方差的增加, 可以有效地避免类簇聚合中出现某些类簇的规模非常大、而大多数类簇只包含极少的点的极端现象. 为了能够与其他基于类簇间距离的链接准则保持一致, Lance 和 Williams 二人提出了 Lance-Williams 算法[13], 将其应用到 Ward 链接准则后, 可以得到类簇间距离的初始化公式与迭代更新公式, 进而可以通过类簇间距离最小化的合并策略去实现基于 Ward 链接准则的层次聚类算法.

同时, 由于三维点云实际上处于嵌入到三维空间的二维流形之中, 即流形虽然处于三维空间, 但是其本真维度为 2. 如果层次聚类算法直接采用三维空间中的欧氏距离作为距离度量, 那么在类簇合并过程中很有可能会出现错误, 把流形中两个并不连通的类簇进行了优先合并, 如图 6(a)所示, 将原本在二维流形中不可达的内层点和外层点错误地归为同一类. 然而, 如果为了避免这一问题而采用本真距离作为距离度量, 则会导致流形上两点之间距离计算的时间复杂度过高, 那么将大大增加距离计算的时间开销.



(a) 无连通性约束的层次聚类　　(b) 有连通性约束的层次聚类

图 6　无连通性约束的层次聚类与有连通性约束的层次聚类在 Swiss Roll 数据集上聚类的对比示意图

为了解决这一问题, 我们采用了具有连通性约束的层次聚类算法. 具体而言, 尽管我们不能直接通过高维空间中的距离度量来计算低维流形中两点的距离, 但是基于流形学习理论, 嵌入到高维空间中的低维流形在局部与欧式空间同胚. 换言之, 低维流形在局部上具有欧式空间的全部性质, 所以可以用欧式距离来计算低维流形上局部两点的本真距离. 因此在实际应用中, 低维流形的局部邻域中两点的欧式距离可以很好地近似本真距离, 即我们可以基于欧式距离来计算近邻点之间的本真距离.

于是, 我们基于欧氏距离去计算三维点云中每个点的 K-近邻点集, 即得到三维点云的 K-近邻图; 然后, 基于该 K-近邻图作为类簇合并时的连通性约束. 在算法的具体实现中, 我们可以通过并查集[14,15]来表示类簇, 并基于并查集与哈希表, 非常高效地实现类簇的连通性约束, 并且将其应用到基于 Ward 链接准则的层次聚类算法中, 其伪代码如算法 2 所示. 图 6(b)展示了有连通性约束的层次聚类在 Swiss Roll 数据集上的聚类结果, 我们可以看到, 没有出现内层点与外层点被划分为同一类簇的现象, 类簇的划分符合流形上的连通性.

算法 2. 有连通性约束的 Ward 层次聚类算法.

输入: 点集 $S=\{\boldsymbol{p}_1,\boldsymbol{p}_2,\ldots,\boldsymbol{p}_N\}\subset\mathbb{R}^3$, K-近邻函数 K-$NN(\cdot)$, 距离度量$\|\cdot\|$.

1: **for** $\boldsymbol{p}_i$ in S **do**
2: 　$C_i\leftarrow MakeSet(\boldsymbol{p}_i)$　　初始化并查集的初始集合
3: $Node_i.cluster\leftarrow C_i$
4: 　$Node_i.left\_son\leftarrow\varnothing$, $Node_i.right\_son\leftarrow\varnothing$　　初始化叶子节点
5: 　K-$NN(C_i)\leftarrow HashSet(\{\{q\}|q\in K\text{-}NN(\boldsymbol{p}_i)\})$　　初始化类簇 K-近邻
6: **end for**
7: $\Omega\leftarrow DisjointSet(\{C_1,C_2,\ldots,C_N\})$　　初始化并查集Ω
8: **for** i in $[1,N-1]$ **do**
9: 　**for** j in $[i+1,N]$ **do**

10: $d(C_i,C_j)\leftarrow d(C_j,C_i)\leftarrow\|\boldsymbol{p}_i-\boldsymbol{p}_j\|$ 初始化类簇间距离

11: **end for**

12: **end for**

13: **for** t in [1,N−1] **do**

14: $C_{i^*},C_{j^*}\leftarrow\underset{C_i\in\Omega,C_j\in K\text{-}NN(C_i)}{\arg\min}\ d(C_i,C_j)$ 基于连通性约束搜索距离最近的类簇

15: $C_{N+t}\leftarrow\Omega.Unite(C_{i^*},C_{j^*})$ 并查集合并操作, Ω中的C_{i^*}与C_{j^*}被替换为C_{N+t}

16: $Node_{N+t}.cluster\leftarrow C_{N+t}$

17: $Node_{N+t}.left\_son\leftarrow Node_{i^*}$, $Node_{N+t}.right\_son\leftarrow Node_{j^*}$ 创建非叶子节点

18: $K\text{-}NN(C_{N+t})\leftarrow K\text{-}NN(C_i)\cup K\text{-}NN(C_j)\backslash\{C_i,C_j\}$ 计算类簇 K-近邻

19: **for** C_k in Ω **do**

20: **if** $C_i\in K\text{-}NN(C_k)$ or $C_j\in K\text{-}NN(C_k)$ **then**

21: $K\text{-}NN(C_k)=(K\text{-}NN(C_k)\backslash\{C_i,C_j\})\cup\{C_{N+t}\}$

22: **end if**

23: **end for** 更新受合并操作影响的类簇的 K-近邻

24: **for** C_k in $\Omega\backslash C_{N+t}$ **do**

25: $\alpha_i\leftarrow\dfrac{|C_{i^*}|+|C_k|}{|C_{i^*}|+|C_{j^*}|+|C_k|}$, $\alpha_j\leftarrow\dfrac{|C_{j^*}|+|C_k|}{|C_{i^*}|+|C_{j^*}|+|C_k|}$, $\beta\leftarrow\dfrac{-|C_k|}{|C_{i^*}|+|C_{j^*}|+|C_k|}$

26: $d(C_{N+t},C_k)\leftarrow d(C_k,C_{N+t})\leftarrow\alpha_i d(C_{i^*},C_k)+\alpha_j d(C_{j^*},C_k)+\beta d(C_{i^*},C_{j^*})$

27: **end for** 计算合并后新类簇与其他类簇的距离

28: **end for**

输出: 层次聚类树的根节点 C_{2N-1}.

2.3 层次网络结构的生成

现存的基于三维点云的层次神经网络, 其网络结构大多依赖于点云采样[16]、K 维树[7]等传统数据结构去构建层次结构. 这种网络结构的设计方式过于启发式, 因为其完全依赖于人类的先验知识, 而没有主动去学习三维点云本身所蕴含的几何结构.

在第 2.2 节中, 我们详细讨论了如何通过基于连通性约束与 Ward 链接准则的层次聚类算法去学习三维点云的几何结构, 即得到一棵记录类簇聚合全部过程的层次聚类树. 于是, 我们可以基于从三维点云中学习到的层次聚类树来自动生成层次神经网络的结构. 具体而言, 我们使用含参数的层次神经网络结构的生成方法.

这里的参数是指类簇数目的列表$[M_1,M_2,\ldots,M_T]$, 这意味着我们会将层次聚类过程分为 T 个阶段, 列表中的 M_t代表着第 t 阶段将当前的 M_{t-1} 个类簇聚合为 M_t 个类簇(令初始类簇的数目为 N=M_0). 换言之, 层次聚类算法原本会合并 N−1. 次类簇(包含 N 个点的三维点云), 但是我们通过传入的参数列表$[M_1,M_2,\ldots,M_T]$将整个合并过程划分为 T 个不同的阶段, 其中每个阶段会包含若干个合并类簇的操作, 而且每个阶段最终都会合并为参数所指定的类簇数目. 由于 T<<N, 且每个阶段中多个合并类簇的操作可以一次性完成, 所以含参数生成方法的聚合效率与可并行性均远远优于无参数的生成方法. 如图 7 所示, 我们以一个飞机的三维点云作为例子, 更加形象地展示了含参数的生成方法.

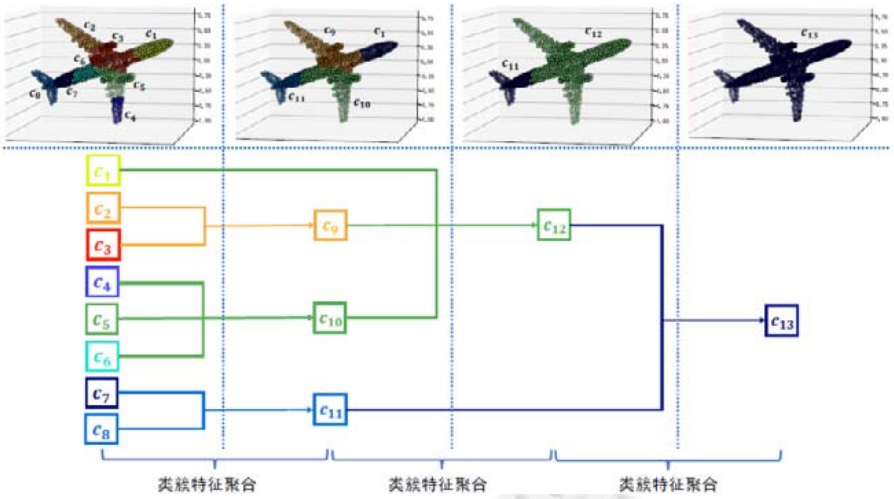


图 7 基于含参数的生成方法来生成层次神经网络结构的示意图

2.4 类簇的嵌入表达学习

2.4.1 类簇的边卷积模块

原始的边卷积层[9]只适用于学习三维点云中每个点的嵌入表达, 无法直接应用于层次聚类树中各个类簇的嵌入表达学习. 将边卷积拓展为适用于类簇特征的形式, 其难点在于如何定义类簇集合上的图, 即类簇间的边的关系. 于是, 我们提出两种方法来定义类簇之间的边集.

- 基于层次聚类算法中的类簇间距离: 如算法 2 中第 24–27 行所示, 基于 Ward 链接准则的层次聚类算法会根据 Lance-Williams 公式计算类簇间距离, 我们可以将其作为类簇间的距离度量, 建立类簇的邻域关系(例如 K-近邻关系), 然后将每个类簇 C_t 与其邻居类簇建立边的关系$\mathcal{E}_t$. 于是, 我们就得到了类簇集合上的图 $G=(\Omega,\mathcal{E})$, 其中, $\Omega=\{C_t|1\leqslant t\leqslant T\}$, $\mathcal{E}=\bigcup_{1\leqslant t\leqslant T}\mathcal{E}_t$.
- 基于类簇特征的距离: 基于层次聚类算法中的类簇间距离, 本质上还是基于输入空间$\mathbb{R}^3$ 中的欧氏距离. 而第 2 种方式是基于特征空间(feature space)中的距离作为类簇间距离. 按照凝聚型层次聚类的观点来看, 最开始时, 三维点云中的每个点都构成一个初始类簇, 所以三维点云中点的特征也可以看作是特殊的类簇特征, 只不过该类簇仅包含一个点而已. 因此, 最开始时, 我们可以将三维点云 $\{\boldsymbol{p}_i\in\mathbb{R}^3|1\leqslant i\leqslant N\}$上的$K$-近邻图作为初始类簇集合$\{\{\boldsymbol{p}_i\}|1\leqslant i\leqslant N\}$上的$K$-近邻图, 并通过边卷积层来提取初始类簇的特征 $\{f_i^{(0)}|1\leqslant i\leqslant N\}$. 然后, 假如我们通过 K 次类簇特征聚合剩下 T 个类簇特征 $\{f_i^{(k)}|1\leqslant i\leqslant T\}$, 我们就可以通过类簇特征之间的距离, 即 $|f_i^{(k)}-f_j^{(k)}|$, 作为类簇间距离. 例如, 我们可以计算两个类簇特征之间的余弦距离、欧氏距离等等. 基于类簇间的距离, 我们就可以建立类簇的邻域关系(例如 K-近邻关系), 然后将每个类簇 C_t 与其邻居类簇建立边的关系$\mathcal{E}_t$. 于是, 我们就得到了类簇集合上的图 $G=(\Omega,\mathcal{E})$, 其中, $\Omega=\{C_t|1\leqslant t\leqslant T\}$, $\mathcal{E}=\bigcup_{1\leqslant t\leqslant T}\mathcal{E}_t$.

基于上述方法确定了类簇集合上的图结构, 那么类簇集合上的边卷积就非常简单了. 假设在层次聚类树的某一层, 层次聚类算法将三维点云 $S=\{\boldsymbol{p}_i\in\mathbb{R}^3|1\leqslant i\leqslant N\}$聚类为 T 个类簇, 记为$\Omega=\{C_t|1\leqslant t\leqslant T\}$, 类簇集合$\Omega$上的图结构为 $G=(\Omega,\mathcal{E})$, 并且我们在上一层中已经提取了这 T 个类簇所对应的类簇特征 $\{f_i\in\mathbb{R}^{d_i}|1\leqslant i\leqslant T\}$, 则类簇集合$\Omega$上的边卷积层采用如下形式:

$$\begin{cases} f_i'=\max\limits_{j:(i,j)\in\mathcal{E}} e_{ij} \\ e_{ij}\triangleq(\hat{e}_{ij})=h_\Theta(f_i\oplus(f_j-f_i)) \end{cases} \tag{7}$$

其中, f_i' 为边卷积提取到的类簇 C_i 的嵌入表达; $f_i\oplus(f_j-f_i)$表示中心类簇 C_i 与其 k-近邻类簇 C_j 的特征进行作差,

得到 f_i–f_j, 然后再与中心类簇特征 f_i 进行拼接; $h_\Theta(\cdot)$是以Θ为待学习参数的神经网络.

2.4.2 类簇特征聚合

基于第 2.3 节得到的索引列表 children, 我们就可以定义一种索引池化层(indexed pooling layer)来进行类簇特征的聚合. 顾名思义, 它与一般的池化操作不同的是, 索引池化层并不是对于所有特征都进行特征聚合, 而是根据索引列表对于指定的特征进行池化操作. 具体地, 父类簇 C_k 的类簇特征, 其计算公式如下.

$$f_k = \underset{i \in children[k]}{pooling}\ f_i \tag{8}$$

其中, f_i 为类簇 C_i 所对应的类簇特征; *pooling* 为具有排列不变性的池化函数, 一般会采用最大池化或平均池化. 显然, 我们会发现: 索引池化层其实是最大池化层或平均池化层的一般性框架, 它试图以索引的形式指明池化的对象. 当索引列表包含所有特征的索引时, 索引列表则退化为普通的池化层, 即最大池化层、平均池化层均为索引池化层的一种特例. 图 8 详细展示了索引池化操作的实现细节: 第 1 行将类簇 $\{c_1,c_2,c_3\}$ 的聚合过程可视化; 第 2 行形象地展示了与第 1 行相对应的索引最大池化操作, 通过索引列表 *children*[4]=[1,2,3], 将子类簇特征 $\{f_1,f_2,f_3\}$ 聚合为父类簇特征 f_4.

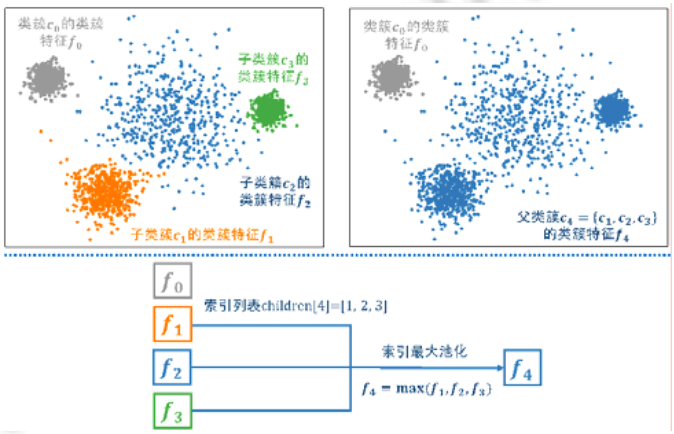


图 8 索引池化操作的示意图

2.5 模型实现细节

2.5.1 类簇特征提取模块(cluster abstraction)

基于第 2.4.1 节与第 2.4.2 节介绍的类簇边卷积层与索引池化层, 我们可以将二者相结合, 构成如图 9 所示的类簇特征提取模块(cluster abstraction), 我们将其简称为 CA 模块. 如图 9 所示, 类簇特征提取模块由边卷积层与索引最大池化层组成, 整个模块的输入是 $C\times d$ 的输入张量, 其表示三维点云中 C 个类簇的类簇特征, 特征维度为 d; 而输出的是一个 $K\times a_n$ 的张量, 即我们从 C 个子类簇的特征中学习到 K 个父类簇的特征.

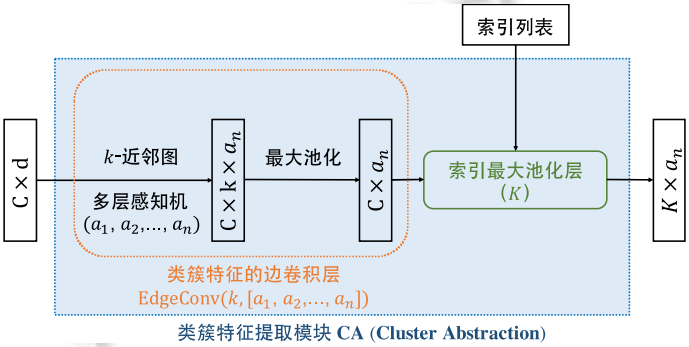


图 9 类簇特征提取模块的网络结构图

2.5.2 点云分类任务的网络结构

对于点云分类任务而言，我们的模型需要学习三维点云的全局特征，然后从全局特征中提取用于分类预测的类别权重. 具体的，从图 10 中我们可以看到，点云分类任务下的层次聚类神经网络结构可以分为以下 4 个部分.

(1) 首先，通过第 2.2 节中介绍的层次聚类算法来学习三维点云的几何结构，即得到一棵层次聚类树.

(2) 基于第 2.3 节介绍的含参数生成方法来确定层次神经网络的结构，其参数为[32,8,1]. 于是，我们就可以得到类簇特征的聚合过程，且聚合过程可以通过索引列表来实现，如图 7 所示.

(3) 基于索引列表，我们可以确定类簇特征提取模块(cluster abstraction)，然后通过不断堆叠该模块，从若干子类簇特征中提取父类簇特征，直到获得整个三维点云的全局特征.

(4) 由于全局特征综合了整个三维点云的信息，所以我们可以通过一个三层感知机，将点云的全局特征转化为用于分类预测的类别权重.

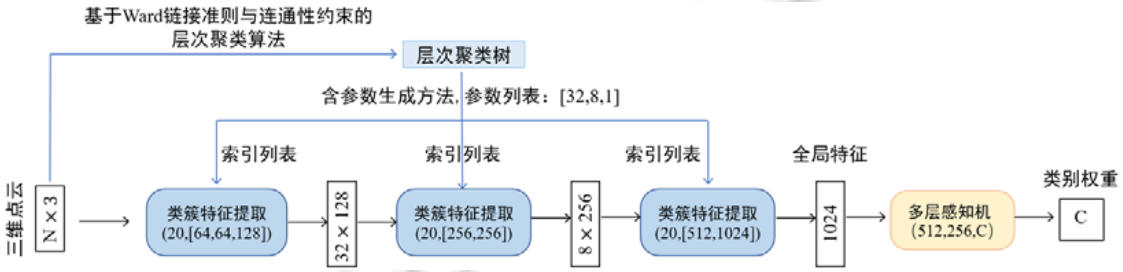


图 10 三维点云分类任务下，层次聚类神经网络的网络结构图

2.5.3 点云分割任务的网络结构

对于点云语义分割任务而言，我们的模型不仅需要学习三维点云的全局特征，还需要学习各层次下的局部特征，并且结合所有特征预测三维点云中每个点的类别权重.

如图 11 所示，层次聚类神经网络在点云语义分割任务下的网络结构建立在点云分类任务的基础之上，它把点云分类任务中由 3 个类簇特征提取模块(CA)构成的网络结构作为骨干网络，用来提取点云在不同层次下的类簇特征. 具体而言，从图 11 中我们会发现，点云语义分割任务下的层次聚类神经网络结构可以分为以下 6 个部分.

(1) 首先，我们会通过第 2.2 节中介绍的层次聚类算法来学习三维点云的几何结构，即得到一棵层次聚类树，这一步与点云分类任务一致.

(2) 基于第 2.3 节介绍的含参数生成方法来确定层次神经网络的结构，其参数与点云分类任务相同，得到表征类簇特征聚合过程的索引列表.

(3) 基于索引列表，我们可以确定类簇特征提取模块(CA)中的索引最大池化层. 在点云语义分割任务中，我们采用了更大的 K-近邻域，将点云分类任务中采用的 K=20 提升至 K=40.

(4) 与分类任务不同的是，我们还在网络最开始阶段加入了一个边卷积模块，用来提取三维点云中的点特征，它只包含每个点的 K-近邻域信息，而不包含类簇的信息. 需要注意的是：边卷积模块所采用的图结构与类簇特征提取模块保持一致，均为 40-近邻图.

(5) 我们将 3 个 CA 模块提取到的类簇特征进行上采样操作(up-sampling)，得到与三维点云中点的数目相同的类簇特征，即三维点云中的每个点都得到其所属类簇的类簇特征，然后我们就可以将边卷积提取到的点特征与经过上采样的 3 个类簇特征进行拼接(如图 11 中的⊕所示)，于是，三维点云中每个点都得到了一个 1216 维的高维嵌入表达.

(6) 最后，我们可以通过一个含有随机丢失层 Dropout[17−19]的四层感知机，将三维点云中各点的嵌入表达转化为用于分割预测的类别权重，其中，每个神经元的丢失概率为 $drop\_rate$=0.5.

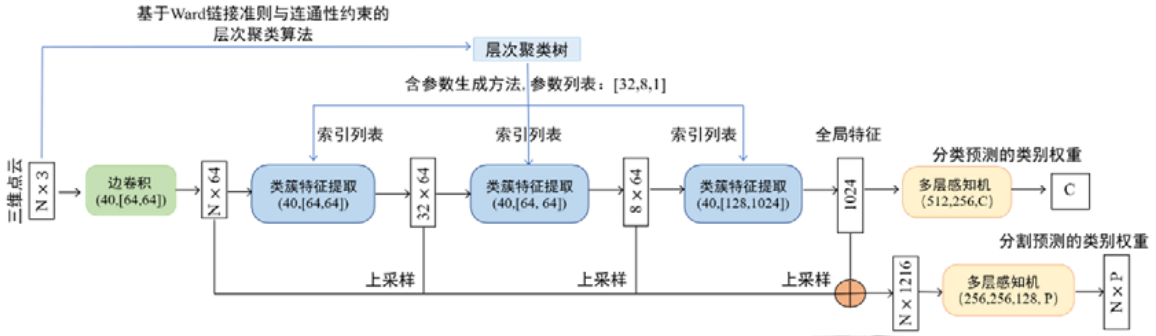


图 11　三维点云语义分割任务下, 层次聚类神经网络的网络结构图

2.5.4　ClusterNet 与严格旋转不变性表达的结合

为了使得我们的模型具有严格旋转不变性, 我们需要将本节中提出的层次聚类神经网络与第 1 节中提出的严格旋转不变性映射算法进行结合. 在第 1.3 节中, 我们介绍了严格旋转不变性表达具有非常良好的兼容性, 它可以通过如图 4 所示的兼容性网络结构, 将 $N\times(3\times K+1)$的严格旋转不变性表达(式(1))转化为 $N\times D$ 的矩阵形式, 使其能够适用于当前大多数的基于三维点云的深度学习模型. 因此, 我们可以很容易地将严格旋转不变性表达应用于本节所提出的层次聚类神经网络, 具体的网络结构如图 12 所示.

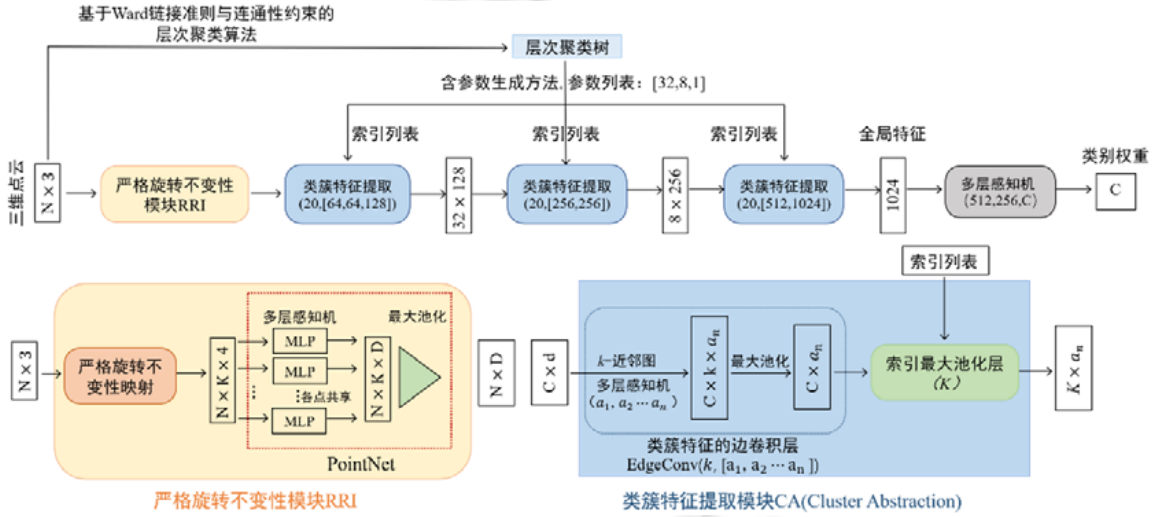


图 12　在点云分类任务下, 具有严格旋转不变性的层次聚类网络结构图

由图 12 可知, 层次聚类神经网络与严格旋转不变性表达的结合非常简单, 只需要在网络结构的初始阶段加入一个严格旋转不变性模块(RRI)即可. 具体而言: ① 在正式进行嵌入表达学习之前, 首先要根据严格旋转不变性映射算法 1, 将三维点云转化为严格旋转不变性表达(式(1)), 然后将其重新组织为 $N\times k\times 4$ 的三维矩阵形式(6); ② 然后, 我们将具有通用近似能力的 PointNet 网络作为基本模块, 即通过各点共享的全连接层与全局最大池化层来学习三维点云中各点的嵌入表达; ③ 最后则可以将具有严格旋转不变性的嵌入表达传入类簇特征提取模块(CA), 学习在不同层次下类簇的嵌入表达. 我们会注意到, 严格旋转不变性模块 RRI 得到的是关于点的特征(point feature), 所以三维点云上的 k-近邻关系可以对应于点特征之间的 k-近邻关系. 因此,严格旋转不变性表达(式(6))可以直接传入类簇特征提取模块(CA), 即能够兼容于分类任务与语义分割任务下的层次聚类神经网络.

2.5.5 模型训练与测试的方法

(1) 点云分类任务下, 模型的训练与测试方法

在点云分类任务中, 我们可以通过层次聚类神经网络的前向传播得到预测分类的类别权重, 我们不妨将其记为 $x\in \mathbb{R}^C$, 其中, C 为点云类别的数目. 基于类别权重, 我们采用交叉熵损失函数作为层次聚类神经网络的损失函数, 其具体形式如式(9)所示.

$$L(x,class)=-\log\left(\frac{\exp(x[class])}{\sum_j \exp(\boldsymbol{x}[j])}\right)=-x[class]+\log(\sum\nolimits_j \exp(\boldsymbol{x}[j])) \tag{9}$$

其中, x 为 C 维向量, 表示模型对于点云在 C 种类别下的预测权重; $class$ 为点云的正确类别. 我们采用 Adam 优化器[20]来更新网络参数, 设置初始学习率为 0.01, 每 20 轮(epoch)训练衰减为原来的 0.7 倍, 并且学习率的衰减下界为 1×10^{-5}. 整个训练阶段, 共训练 250 轮(epoch), $|Batch|$=32. 为了通过并行化计算来提高训练效率, 我们使用了一张 12G 的 NVIDIA TitanX 显卡进行训练.

在测试阶段, 我们取权重最大的类别 argmax(x)作为模型预测的类别, 然后计算模型在整个测试集上的分类准确率(accuracy), 即点云预测正确的数目与测试集大小的比值.

(2) 点云语义分割任务下, 模型的训练与测试方法

由图 11 可知, 我们可以通过层次聚类神经网络的前向传播, 既可以得到预测分类的类别权重向量 $x\in \mathbb{R}^{C_1}$, 又可以得到预测语义分割的类别权重矩阵 $X\in \mathbb{R}^{N\times C_2}$. 其中, C_1 为点云分类中的类别数目, C_2 为语义分割中的类别数目, N 为三维点云中点的数目.

对于同时包含分类与分割的真实类别的训练集, 我们采用的损失函数由分类交叉熵损失函数与分割交叉熵损失函数组成, 具体如式(10)所示.

$$\left.\begin{aligned}&\mathcal{L}=\mathcal{L}_{cls}+\mathcal{L}_{seg}\\&\mathcal{L}_{cls}\triangleq L(x,class_{cls})=-\log\left(\frac{\exp(x[class_{cls}])}{\sum_j \exp(\boldsymbol{x}[j])}\right)\\&\mathcal{L}_{seg}\triangleq \frac{1}{N}\sum\nolimits_{i=1}^{N}\mathcal{L}(X_{row_i},class_{seg}[i])=-\frac{1}{N}\sum\nolimits_{i=1}^{N}\log\left(\frac{\exp(X_{i,class_{seg}[i]})}{\sum_j \exp(X_{ij})}\right)\end{aligned}\right\} \tag{10}$$

其中, $class_{cls}$ 为点云分类的真实类别, 由一个整数表示; $class_{seg}$ 为点云分割的真实类别, 由一个 N 维整数向量表示, 因此, $class_{seg}[i]$为点云中第 i 个点的真实类别; X_{row_i} 为分割权重矩阵 X 的第 i 行, X_{ij} 为分割权重矩阵 X 中第 i 行且第 j 列的元素. 而如果训练集仅包含语义分割的真实类别, 那么此时可以令$\mathcal{L}$=$\mathcal{L}_{seg}$, 即直接将分割交叉熵函数作为损失函数.

我们同样采用 Adam 优化器[20]更新网络中的参数, 其中, 初始学习率设置为 1×10^{-2}, 并随着训练轮数增加而指数衰减, 衰减步长为 20 个 epoch, 衰减率为 0.5, 且衰减下界为 1×10^{-5}. 总计 150 个 Epoch, $|Batch|$=32. 我们采用了两块 12 GB 的 NVIDIA TitanX 显卡进行训练.

在测试阶段, 得到预测语义分割类别的权重矩阵 X, 然后对每个点 p_i 取权重最大的类别 $\arg\max(X_{row_i})$ 作为模型预测的类别, 则可以计算模型在整个测试集上的准确率($accuracy$)与平均交并比($mIOU$).

3 实验结果与分析

3.1 点云分类任务的实验

在本节中, 我们将进行与点云分类任务有关的实验, 比较我们的方法与其他先进方法在该任务下的性能差异. 首先, 我们将详细介绍外部对比实验的具体设置, 包括数据集及其预处理方法、评价方法的设计以及数据集 $SO(3)$增强的具体实现. 我们进行了外部对比实验, 比较我们的方法与其他先进方法在旋转鲁棒性上的性能表现.

3.1.1 实验设置

(1) 数据集介绍

在点云分类任务的实验中, 我们采用了被广泛使用的 ModelNet40[21]作为基准数据集. 如表 1 所示, ModelNet40 数据集包含 12 311 个 CAD 物体模型, 共分为 40 种不同的物体类别, 其中, 9 843 个模型作为训练集, 剩下的 2 468 个模型用于测试集.

表 1 ModelNet40 物体分类数据集

基准数据集	类别数目	物体数目	训练集大小	测试集大小	点云大小
ModelNet40	40	12 311	9 843	2 468	1024×3

ModelNet40 数据集中的 CAD 物体模型, 是一种 Mesh 数据形式, 由描述物体表面的若干三角形面组成, 其中记录了所有三角形面的顶点坐标. 为了从 CAD 模型中采样出三维点云, 我们按照 Charles R. Qi 等人[6]的方法, 根据每个三角形面的面积与物体表面积之比作为采样概率, 在全体三角形面上进行采样. 这意味着: 面积更大的三角形面具有更高的采样概率, 采样点的数目会更多. 为了与现存方法保持一致, 我们选取采样点数目为 1 024, 即得到 1024×3 的三维点云数据.

考虑到模型对于点云扰动的鲁棒性, 我们在训练阶段, 对训练集中的三维点云进行了预处理工作, 其依次分为 4 个步骤.

1) 点云归一化: 将三维点云的重心平移到笛卡尔坐标的原点, 然后将平移后的点云进行尺度放缩, 把点云约束为半径为 1 的球体中.
2) 点云随机扰动: 给三维点云中的每个点的笛卡尔坐标(x,y,z)加上一个随机向量$(\mathrm{d}x,\mathrm{d}y,\mathrm{d}z)$进行扰动, 其中, $\mathrm{d}x,\mathrm{d}y,\mathrm{d}z$ 均服从截断高斯分布, 均值为 0, 方差为 0.01, 截断值为 0.05, 即
$$-0.05\leqslant\max(\mathrm{d}x,\mathrm{d}y,\mathrm{d}z)\leqslant 0.05.$$
3) 点云随机放缩: 在[0.8,1.25]上的均匀分布中进行采样, 得到放缩尺度$\alpha\in\mathbb{R}$, 然后将点云中每个点 p 都乘以α, 得到放缩后的点αp.
4) 点云随机平移: 在[−0.1,0.1]上的均匀分布中进行 3 次采样得到平移向量 $m\in\mathbb{R}^3$, 然后将点云中每个点 p 都加上平移向量, 得到平移后的点 $p+m$.

(2) 模型的评价方法

我们设计了一种新的评价基准来衡量不同模型在三维旋转鲁棒性上的表现. 为了更加全面地评估模型的旋转鲁棒性, 我们采用了 3 种评价方法, 如下所示.

1) z/z: 这是目前大多数点云感知工作所采用的评价方法. 在训练阶段, 训练集中的三维点云需要经过额外的预处理, 即沿着 z 轴(笛卡尔坐标系)旋转随机角度. 在测试阶段, 我们首先在区间[0,2π]上均匀采样 60 个旋转角, 将测试集中的每个点云沿着 z 轴旋转对应的角度, 得到 60 个不同姿态下的新点云; 最后, 我们使用增强后的测试集来测试模型的分类准确率.
2) $z/SO(3)$: 在训练阶段, 与步骤 1)保持一致. 由于目前绝大多数测试集所包含的三维物体都只有一种固定的姿态, 或者只沿着固定的欧拉轴旋转, 因此在 $z/SO(3)$的测试阶段, 我们不再限制欧拉轴为 z 轴, 而是对测试集进行更全面的旋转增强, 即首先在三维旋转变换群 $SO(3)$中均匀采样得到旋转变换矩阵, 然后通过矩阵乘法对测试集进行扩充, 最后使用增强后的测试集来测试模型的分类准确率. 为了描述的准确性与简洁性, 我们将这种对于数据集的旋转增强称为 $SO(3)$增强.
3) $SO(3)/SO(3)$: 为了比较的公平性, 我们在训练阶段也加入更加全面的旋转增强, 即训练集在预处理时采用 $SO(3)$群中均匀采样得到的旋转矩阵, 让模型在训练阶段能够学习不同姿态下的三维点云. 测试阶段与步骤 2)保持一致, 采用 $SO(3)$增强后的测试集.

关于数据集的 $SO(3)$增强, 可以通过球面均匀采样或者斐波那契网格[22]来得到欧拉轴 e, 在区间[0,2π]上均匀采样得到旋转角θ, 然后将根据如下公式得到对应的旋转变换矩阵:

$$R = I_3\cos\theta + (1-\cos\theta)ee^T + [e]_\times \sin\theta,\ [e]_\times \triangleq \begin{bmatrix} 0 & -e_3 & e_2 \\ e_3 & 0 & -e_1 \\ -e_2 & e_1 & 0 \end{bmatrix} \tag{11}$$

其中, I_3 为 3×3 的单位矩阵, e 为欧拉轴, θ为旋转角.

于是, 我们根据该方法实现了三维旋转变换群 $SO(3)$上的均匀采样. 基于这种采样方法, 总共采样了 500 个欧拉轴与 60 个旋转角度, 即得到了 30 000 个在 $SO(3)$群上均匀分布的旋转变换. 然后, 我们通过这些旋转变换对 ModelNet40 的测试集进行 $SO(3)$增强, 增强后的测试集大小为 2468×30000=74040000, 即测试集中的每个点云都进行了 30 000 种不同的旋转变换, 得到同一个物体在不同姿态下的三维点云.

3.1.2 外部对比实验

为了验证层次聚类神经网络在点云分类任务中的优势, 我们通过第 3.1.1 节中介绍的 3 种评估方法来比较我们的模型与现有的先进方法在旋转鲁棒性上的性能.

(1) 基准方法

在对比实验中, 我们选取了 9 种先进方法作为基准方法, 涵盖了三维物体的多种表示形式: 体素网格、多视图、三维点云、球面表达, 而且均代表了目前最先进的方法. 具体包括 SubVolSup[23], MVCNN[24], PointNet[6], PointNet(w/STN)[6], PointNet++[16], PointNet++(w/STN)[16], DGCNN[9], DGCNN(w/STN)[9]和 Spherical CNN[25]. (w/STN)代表模型使用空间变换网络 STN[26].

(2) 对比实验结果及其分析

表 2 展示了各算法在 ModelNet40 数据集上的分类准确率. 由表 2 可知, 在目前大多数工作所采用的 z/z 评价方法下, 9 种基准方法在 ModelNet40 数据集上取得了 88.5%–92.2%的准确率. 然而一旦对测试集进行 $SO(3)$增强, 即采用 z/$SO(3)$评价方法, 我们发现, 准确率发生了断崖式下降. 因此, 如果训练阶段只采用沿 z 轴旋转增强的训练集, 前 8 种基准方法的旋转鲁棒性很差, 对于不同姿态下的三维物体泛化性能不足. 即使基于具有旋转等不变性的球形卷积, Spherical CNN 基准方法也暴露出自身的缺点, 无法适应 $SO(3)$增强的测试集, 分类准确率下降了 12 个百分点. 然而, 我们可以看到, 我们提出的算法(如表 2 最后一行所示), 在 z/$SO(3)$评价方法下依然保持 87.1%的分类准确率, 这从实验层面证明了我们的方法具有严格旋转不变性. 不仅如此, 我们的方法在 z/$SO(3)$评价方法下取得了最高准确率, 与其他基准方法相比最少提升 10.2 个百分点.

表 2 在 ModelNet40 点云分类数据集上, 各算法在 3 种评估方法下的分类准确率(%)

算法	输入(大小)	z/z	z/$SO(3)$	$SO(3)$/$SO(3)$
SubVolSup	体素网格(30^3)	88.5	36.6	82.7
MVCNN	多视图(12×224^2)	89.5	70.1	77.6
PointNet	点云(1024×3)	88.7	14.4	79.3
PointNet++	点云+法向量(5000×6)	91.9	16.0	84.7
DGCNN	点云(1024×3)	91.2	16.2	85.5
PointNet(w/ STN)	点云(1024×3)	89.2	16.4	79.8
PointNet++(w/ STN)	点云+法向量(5000×6)	91.8	18.4	85.0
DGCNN(w/ STN)	点云(1024×3)	92.2	20.6	86.2
Spherical CNN	球面表达(2×64^2)	88.9	76.9	86.9
RRI+ClusterNet(我们的算法)	点云(1024×3)	87.1	**87.1**	**87.1**

继续观察表 2 可知, 在 $SO(3)$/$SO(3)$评价方法下, 若训练阶段也加入更加全面的 $SO(3)$增强, 9 种基准方法的旋转鲁棒性得到明显的提升, 但是与 z/z 评价方法相比基准方法的分类准确率平均下降 7.1 个百分点, 即基准方法虽然提升了对于旋转变换的鲁棒性, 但是牺牲了一部分的分类准确性. 导致这一现象可能的原因是: 由于模型的输入空间大大增加, 模型变得更难以收敛. 而我们可以看到, 由于我们的方法具有严格旋转不变性, 我们在 $SO(3)$/$SO(3)$评价方法下依然保持 87.1%的最高准确率, 与其他基准方法相比平均提升 4 个百分点.

3.1.3 内部组件分析

在严格旋转不变性模块(RRI)中, K 是唯一的超参数, 控制着图 $\boldsymbol{G}$ 的连通性. 因此, 我们探究在 RRI 中不同

超参数 K 的设定下算法的性能表现. 如表 3 所示, 实验结果表明, 我们的 ClusterNet 体系结构可以适应 K 的不同值. 例如, 当 K=40 时, 存在接近 25%的点云不满足强连通条件, 但仍可达到有竞争力的分类准确率; 当 K 逐渐增加到 70 以上时, 模型的效果开始保持稳定.

表 3 在 RRI 中, 不同超参数 K 的设定下算法的性能表现

K	40	50	60	70	80	90
准确率(%)	85.6	86.4	86.8	87.0	87.1	87.1

3.2 点云部位语义分割任务的实验

在上一节中, 我们介绍了点云分类任务下的实验结果. 接下来, 本节将详细介绍我们的方法与其他先进方法在点云部位分割任务上的综合性能. 同样地, 我们将介绍实验的具体设置, 比较本文提出的方法与其他基准方法在旋转鲁棒性上的性能差异; 并对部位语义分割结果进行可视化展示, 定性地考察模型在部位语义分割任务中的性能.

3.2.1 实验设置

(1) 数据集介绍

在点云部位语义分割任务的实验中, 我们采用了被广泛使用的 ShapeNetParts 数据集[27]作为基准数据集. 如表 4 所示, ShapeNetParts 数据集包含 16 种物体类别, 总共 16 881 个三维点云数据, 其中, 12 137 个点云作为训练集, 1 870 个点云用于验证集, 而剩下的 2 874 个点云作为测试集. ShapeNetParts 数据集中的三维点云总共标注了 50 种不同的部位, 例如飞机的机翼、杯子的把手等等, 其中每个点云至少标注了 2 种部位, 最多标注了 5 种部位. 除此之外, 数据集中的每个三维点云包含 2 048 个点, 即数据大小为 2048×3.

表 4 ShapeNetParts 部位语义分割数据集

基准数据集	类别数目	物体数目	部位数目	训练集	验证集	测试集	点云大小
ShapeNetParts	16	16 881	50	12 137	1 870	2 874	2048×3

(2) 评价指标与方法

我们采用平均交并比(*mIOU*)作为部位语义分割的评价指标, 并设计了 4 种评价方法.

1) *None*/*None*: 训练集与测试集均不采用任何旋转变换上的增强处理. 该评价方法为目前大多数算法在点云部位分割下的评价方法, 为了观察旋转变换对模型的影响, 我们将其列入实验中作为比较的对象.
2) *None*/*SO*(3): 训练集不采用旋转增强, 但是在测试阶段采用 *SO*(3)增强后的测试集. 该评价方法用来测试现有的算法在常规训练方法下, 对于任意旋转变换的鲁棒性.
3) *z*/*SO*(3): 在训练阶段, 对训练集沿 z 轴旋转增强; 在测试阶段, 采用 *SO*(3)增强后的测试集. 由于部分方法只在训练阶段采用 z 轴旋转增强, 我们将其列入实验之中来分析其旋转鲁棒性.
4) *SO*(3)/*SO*(3): 在训练阶段, 采用 *SO*(3)增强后的训练集; 同样地, 在测试阶段, 也采用 *SO*(3)增强后的测试集. 为了保证比较的公平性, 我们在模型训练阶段也使用 *SO*(3)旋转增强方法, 在此基础上测试模型的效果.

3.2.2 外部对比实验

为了验证层次聚类神经网络在三维点云部位分割任务中的优势, 我们通过上一节中介绍的 4 种评估方法来比较我们的模型与现有的先进方法在旋转鲁棒性上的性能.

(1) 基准方法

在对比实验中, 我们选取了 6 种先进方法作为基准方法, 其中既有不具有旋转鲁棒性的神经网络模型, 又包括结合了严格旋转不变性模块 RRI 的网络结构, 均为该任务下最先进的算法. 具体包括 PointNet[6], PointNet (w/STN)[6], PointNet++[16], DGCNN(w/STN)[9], RRI+PointNet, RRI+DGCNN.

(2) 对比实验结果及其分析

表 5 展示了各算法在 ShapeNetParts 数据集上的平均交并比 *mIOU*(%), 其中采用了上一节所介绍的 4 种评价方法.

表 5 在 ShapeNetParts 部位分割数据集上, 各算法在 4 种评价方法下的平均交并比 *mIOU*(%)

算法	*None*/*None*	*None*/*SO*(3)	*z*/*SO*(3)	*SO*(3)/*SO*(3)
PointNet	83.2	37.7	47.1	78.4
PointNet (w/STN)	83.8	37.9	48.9	79.8
PointNet++	81.1	50.3	67.4	83.4
DGCNN (w/STN)	84.7	43.8	54.8	78.0
RRI+PointNet	81.3	81.3	81.3	81.3
RRI+DGCNN	83.6	83.6	83.6	83.6
RRI+ClusterNet (我们的算法)	84.5	**84.5**	**84.5**	**84.5**

由表 5 可知:

- 在目前大多数工作所采用的 *None*/*None* 评价方法下, 前 4 种基准方法在 ShapeNetParts 数据集上取得了 81.1%–84.7%的平均交并比(*mIOU*).
- 然而一旦对测试集进行 *SO*(3)增强, 即采用 *None*/*SO*(3)评价方法, *mIOU* 均发生了断崖式下降. 这说明不具有旋转不变性的基准方法无法适应不同姿态下的三维物体, 其泛化能力很差.
- 即使在训练阶段加入 *z* 轴旋转增强, 即采用 *z*/*SO*(3)评价方法, 我们会发现, 依然无法改变 *mIOU* 的断崖式下降. 这说明对训练集进行 *z* 轴旋转增强, 对于提升模型的旋转鲁棒性收效甚微.
- 而如果在训练阶段采用 *SO*(3)增强的训练集, 即采用 *SO*(3)/*SO*(3)评价方法, 基准方法 PointNet, PointNet(w/STN)和 DGCNN(w/STN)的 *mIOU* 依然分别下降了 4.8, 4.0 和 6.7 个百分点. 这从实验层面说明: 基于数据集旋转增强的方法只能在一定程度上缓解旋转灾难, 而无法彻底解决这一问题, 其旋转鲁棒性无法得到保证.

基于本文提出的严格旋转不变性模块 RRI、基准方法 RRI+PointNet 与 RRI+DGCNN 在 4 种评价方法下均保持了相同的 *mIOU*, 这从实验层面证明了本文提出的点云表达有良好的兼容性, 能够十分有效地提升模型的旋转鲁棒性, 使得原本旋转鲁棒性较差的算法具有严格旋转不变性. 此外, 我们提出的算法 RRI+CluterNet 不仅具有旋转不变性, 而且在 *None*/*SO*(3), *z*/*SO*(3), *SO*(3)/*SO*(3)这 3 种基准方法下都取得了最好的表现.

3.2.3 可视化结果分析

如图 13–图 15 所示, 我们分别展示了在 *SO*(3)/*SO*(3)评价方法下, 我们的方法与其他 3 种基准方法对于吉他、椅子、笔记本电脑 3 种不同点云的部位语义分割结果. 图 13 中第 1 行展示了点云部位语义分割的结果, 其中, Groud Truth 为标准答案, 展示了每个点的真实的部位类别; 图中第 2 行展示了三维点云中部位类别预测正确的点(用蓝色表示)以及预测错误的点(用红色表示).

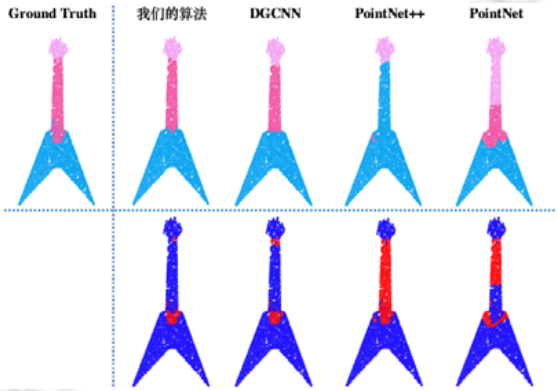


图 13 在 *SO*(3)/*SO*(3)评估方法下, 各算法对吉他所对应的点云进行部位分割的可视化结果

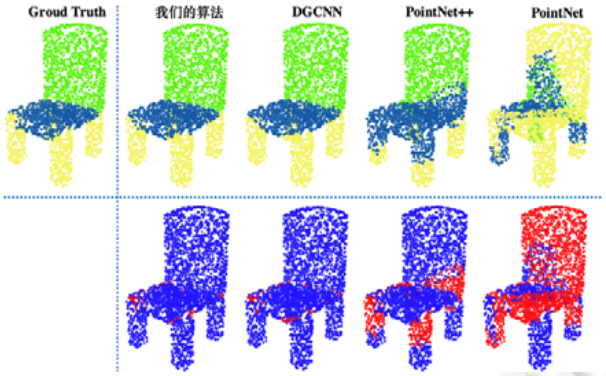


图 14 在 $SO(3)/SO(3)$评估方法下, 各算法对椅子所对应的点云进行部位分割的可视化结果

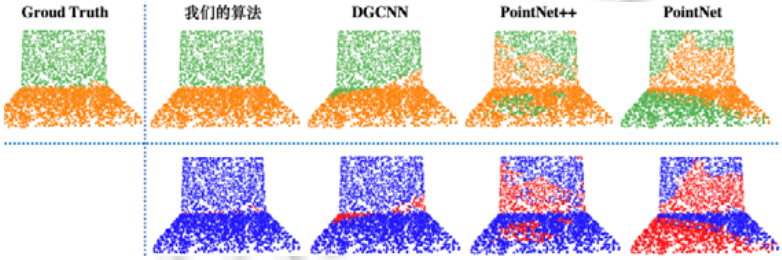


图 15 在 $SO(3)/SO(3)$评估方法下, 各算法对笔记本电脑所对应的点云进行部位分割的可视化结果

观察图 13–图 15, 我们可以进行定性的分析: 我们提出的算法在 3 个点云上均取得了部位语义分割的最佳性能; 其他 3 种基准方法即使在训练阶段采用 $SO(3)$增强的训练集, 即采用 $SO(3)/SO(3)$评价方法, 它们的表现依然十分欠佳. 例如, PointNet++算法对吉他点云进行部位分割时, 明显缺失了吉他的颈部; PointNet 算法对椅子点云进行部位分割时, 出现了大面积的预测错误, 椅背缺失严重; DGCNN 算法对笔记本电脑点云进行部位分割时, 显示器面与键盘面的交界处发生了较多的误分类.

3.3 点云场景语义分割任务的实验

本节将详细介绍我们的方法与其他先进方法在点云场景语义分割任务上的综合性能, 包括实验设置、外部对比实验以及可视化结果分析.

3.3.1 实验设置

(1) 数据集介绍

在点云场景语义分割任务的实验中, 我们采用了被广泛使用的斯坦福大规模三维室内空间数据集 S3DIS[28]作为基准数据集. 由表 6、表 7 可知, S3DIS 数据集包含 6 种不同类型的室内场景, 总共 272 个室内房间, 得到了带有语义标注的 23 585 个三维点云数据, 其中, 点云中的每个点可以被标注为 13 种不同的语义类别, 例如椅子、地面、墙壁等等. 每个三维点云的大小为 4096×9.

表 6 S3DIS 场景语义分割数据集

基准数据集	场景类型数目	房间数目	语义类别数目	点云大小	点的属性
S3DIS	6	272	13	4096×9	XYZ, RGB, $\hat{X}\hat{Y}\hat{Z}$

表 7 S3DIS 数据集中 6 个场景的详细情况

S3DIS 数据集	场景 1	场景 2	场景 3	场景 4	场景 5	场景 6	总计
点云数目	3 687	4 440	1 650	3 662	6 852	3 294	23 585

(2) 评价指标与方法

由表 7 可知, S3DIS 数据集包含 6 种不同类型的室内场景, 每个场景下均采样得到了不同数目的三维点云. 为了综合评价模型的性能, 我们按照 Charles R. Qi 等人[6]采用的 6-fold 交叉验证的策略, 根据场景类型将整个 S3DIS 数据集分为 6 个部分, 按照第 i 个模型 $Model_i$ 采用除场景 i 之外的全体场景作为训练集、剩下的场景 i 作为测试集的方法, 分别训练 6 个模型, 并依次计算各模型的平均交并比 *mIOU*(%)以及平均分类准确率 *mAC*(%), 然后对 6 个模型的平均交并比与平均分类准确率进行平均, 从而得到 6-fold 交叉验证策略下的总体平均交并比 *mIOU* 与总体平均分类准确率 *mAC*, 我们将其作为本节实验的评价指标.

为了更全面地评估模型在点云语义分割任务中的综合性能, 我们基于总体平均交并比 *mIOU* 与总体平均分类准确率 *mAC*, 采用了 *None*/*None*, *None*/*SO*(3), *SO*(3)/*SO*(3)这 3 种评价方法.

3.3.2 外部对比实验

(1) 基准方法

在对比实验中, 我们选取了 5 种先进方法作为基准方法, 包括 PointNet[6], PointNet++[16], DGCNN[9], RRI+PointNet, RRI+DGCNN.

(2) 对比实验结果及其分析

表 8 展示了各算法在 S3DIS 数据集上关于场景语义分割的总体平均交并比 *mIOU*(%)和总体平均分类准确率 *mAC*(%), 其中采用了上节介绍的 3 种评价方法. 由表 8 可知, 实验结果与点云部位语义分割任务保持一致, 再一次有力地验证了 RRI 模块具有旋转不变性, 我们提出的算法 RRI+CluterNet 可以取得最佳的综合性能.

表 8 在 S3DIS 场景语义分割数据集上, 各算法在 3 种评价方法下的总体平均交并比 *mIOU* (%)以及总体平均分类准确率 *mAC* (%)

算法	*None*/*None*		*None*/*SO*(3)		*SO*(3)/*SO*(3)	
	mIOU	*mAC*	*mIOU*	*mAC*	*mIOU*	*mAC*
PointNet	47.71	78.62	8.39	25.82	38.73	72.28
PointNet++	56.83	83.70	13.60	37.01	43.07	74.31
DGCNN	56.10	84.10	11.49	33.41	46.41	77.91
RRI+PointNet	47.62	78.23	47.62	78.23	47.62	78.23
RRI+DGCNN	52.16	82.49	52.16	82.49	52.16	82.49
RRI+ClusterNet(我们的算法)	53.38	83.57	**53.38**	**83.57**	**53.38**	**83.57**

3.3.3 可视化结果分析

如图 16 所示, 根据可视化结果, 我们可以进行定性分析: 我们的算法在场景语义分割上具有非常良好的性能, 能够十分有效地对点云场景进行语义层面上的理解.

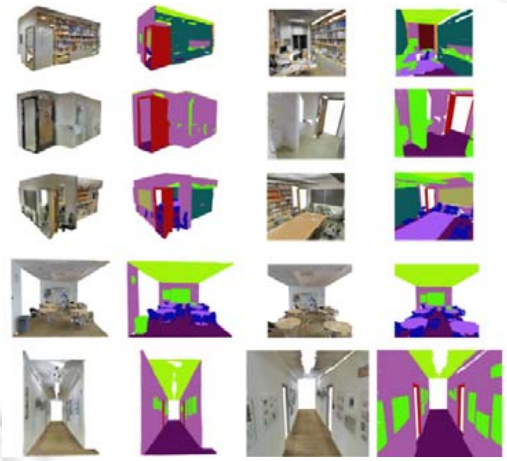


图 16 我们提出的算法 RRI+ClusterNet 在 S3DIS 数据集上进行场景语义分割的可视化结果

4 未来展望

本文虽然在三维点云分析问题上取得了一些成果, 但依然存在不少改进空间及进一步拓展的研究方向.

(1) 目前, 计算机三维视觉领域中, 基于三维点云的模型往往需要稠密点云(点云中包含的点的数目较大)作为输入, 然而在无人驾驶等实际应用中, 往往获取到的是十分稀疏的点云, 这种苛刻的条件对于目前绝大多数先进方法而言, 其识别精度都会不可避免地发生大幅下降. 因此, 对于点云疏密程度具有鲁棒性的方法将会是未来的主流. 在后续的工作中, 我希望将点云密度鲁棒性作为进一步探索的研究方向.

(2) 在本文的实验部分, 我们进行了点云识别、点云部位语义分割以及点云场景语义分割三大任务的实验. 但是在无人驾驶等实际应用中, 三维物体检测也是一种非常重要的任务需求, 它对避障、紧急制动等动作的规划与执行具有深刻的意义. 因此在未来的工作中, 我们计划将现有的 ClusterNet 网络结构拓展为可应用于物体检测的版本, 并且增加三维物体检测任务的对比实验与定性分析.

(3) 由于三维点云其实可以看作一种没有边集的特殊图结构, 因此本文提出的 ClusterNet 其实可以很自然地应用于图数据. 换言之, 我们可以将 ClusterNet 拓展为图神经网络, 用来学习图数据中的各个节点的嵌入表达. 在后续的工作中, 我希望将我们的工作从计算机三维视觉领域迁移到几何深度学习领域, 在更加通用的非欧式数据中学习数据所蕴含的关联信息.

References:

[1] Goodfellow IJ, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. arXiv:1412.6572, 2014.

[2] Kurakin A, Goodfellow I, Bengio S. Adversarial machine learning at scale. arXiv:1611.01236, 2016.

[3] Tramèr F, Kurakin A, Papernot N, Goodfellow I, Boneh D, McDaniel P. Ensemble adversarial training: Attacks and defenses. arXiv: 1705.07204, 2017.

[4] Zhou H, Chen KJ, Zhang WM, Fang H, Zhou WB, Yu NH. DUP-net: Denoiser and upsampler network for 3D adversarial point clouds defense. In: Proc. of the IEEE Int'l Conf. on Computer Vision. 2019. 1961−1970.

[5] Zhang Q, Yang JC, Fang RY, Ni BB, Liu JX, Tian Q. Adversarial attack and defense on point sets. arXiv:1902.10899, 2019.

[6] Qi CR, Su H, Mo KC, Guibas LJ. PointNet: Deep learning on point sets for 3D classification and segmentation. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. 2017. 652−660.

[7] Klokov R, Lempitsky V. Escape from cells: Deep Kd-networks for the recognition of 3D point cloud models. In: Proc. of the IEEE Int'l Conf. on Computer Vision. 2017. 863−872.

[8] Riegler G, Ulusoy AO, Geiger A. OctNet: Learning deep 3D representations at high resolutions. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. 2017. 3577−3586.

[9] Wang Y, Sun YB, Liu ZW, Sarma SE, Bronstein MM, Solomon JM. Dynamic graph CNN for learning on point clouds. ACM Trans. on Graphics (TOG), 2019, 38(5): 1−12.

[10] Ward JH. Hierarchical grouping to optimize an objective function. Journal of the American Statistical Association, 1963, 58(301): 236−244.

[11] Murtagh F, Legendre P. Ward's hierarchical agglomerative clustering method: Which algorithms implement ward's criterion? Journal of Classification, 2014, 31(3): 274−295.

[12] Müllner D. Modern hierarchical, agglomerative clustering algorithms. arXiv:1109.2378, 2011.

[13] Lance GN, Williams WT. A general theory of classificatory sorting strategies: 1. hierarchical systems. The Computer Journal, 1967, 9(4): 373−380.

[14] Galler BA, Fisher MJ. An improved equivalence algorithm. Communications of the ACM, 1964, 7(5): 301−303.

[15] Hopcroft JE, Ullman JD. Set merging algorithms. SIAM Journal on Computing, 1973, 2(4): 294−303.

[16] Qi CR, Yi L, Su H, Guibas LJ. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In: Proc. of the Advances in Neural Information Processing Systems. 2017. 5099−5108.

[17] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. In: Proc. of the Advances in Neural Information Processing Systems. 2012. 1106−1114.

[18] Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR. Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580, 2012.

[19] Srivastava N, Hinton GE, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 2014, 15(1): 1929−1958.

[20] Kingma DP, Ba J. Adam: A method for stochastic optimization. arXiv:1412.6980v9, 2017.

[21] Wu ZR, Song S, Khosla A, Yu F, Zhang LG, Tang XO, Xiao JX. 3D ShapeNets: A deep representation for volumetric shapes. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. 2015. 1912−1920.

[22] González Á. Measurement of areas on a sphere using Fibonacci and latitude—Longitude lattices. Mathematical Geosciences, 2010, 42(1): 49−64.

[23] Qi CR, Su H, Niessner M, Dai A, Yan MY, Guibas LJ. Volumetric and multi-view cnns for object classification on 3D data. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition. 2016. 5648−5656.

[24] Su H, Maji S, Kalogerakis E, Learned-Miller E. Multi-view convolutional neural networks for 3D shape recognition. In: Proc. of the IEEE Int'l Conf. on Computer Vision. 2015. 945−953.

[25] Esteves C, Allen-blanchette C, Makadia A, Daniilidis K. Learning SO(3) equivariant representations with spherical CNNs. In: Proc. of the European Conf. on Computer Vision. 2018. 52−68.

[26] Jaderberg M, Simonyan K, Zisserman A, Kavukcuoglu K. Spatial transformer networks. In: Advances in Neural Information Processing Systems. 2015.

[27] Yi L, Kim VG, Ceylan D, Shen IC, Yan MY, Su H, Lu CW, Huang QX, Sheffer A, Guibas L. A scalable active framework for region annotation in 3D shape collections. ACM Trans. on Graphics (ToG), 2016, 35(6): 1−12.

[28] Armeni I, Sener O, Zamir AR, Jiang H, Brilakis I, Fischer M, Savarese S. 3D semantic parsing of large-scale indoor spaces. In: Proc. of the IEEE Int'l Conf. on Computer Vision and Pattern Recognition. 2016. 1534−1543.

李冠彬(1986－), 男, 博士, 副教授, 博士生导师, CCF 高级会员, 主要研究领域为计算机视觉, 机器学习.

陈超(1994－), 男, 硕士, 主要研究领域为计算机视觉.

张锐斐(1998－), 男, 学士, 主要研究领域为计算机视觉.

林倞(1981－), 男, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为计算机视觉, 机器学习.