Points

陈乐偲

Abstract

在数据集ModelNet中实现了基于点云的3D物体分类模型

- PointNet
- PointNet++
- PPF
- DGCNN
- PointCNN

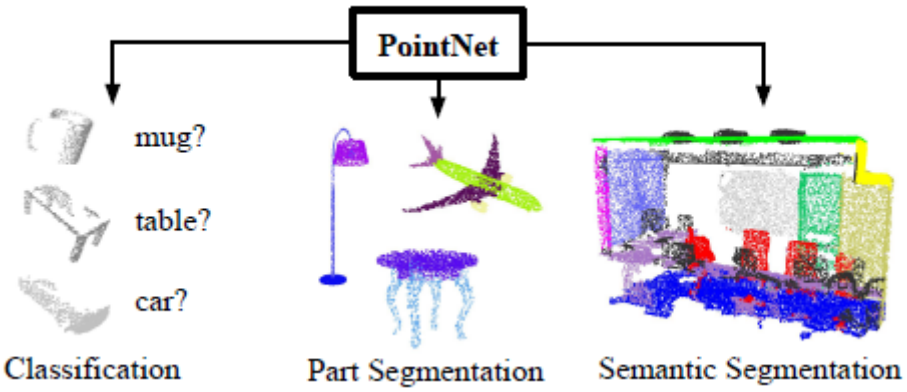
PointNet

3d点云是重要的3d数据表示形式，先前将点云数据转化为体素表示加3d卷积的方法非常消耗时间和空间。PointNet直接学习点云数据，并且学习到点云数据的排列不变性质。

之前基于深度学习的方式处理3d物体有几种方式，

- 基于体素，但只适用于稀疏情况
- 基于多视角，将3d问题转化为2d问题，但对于场景理解等问题不直接
- 基于流行学习，但流形应用于家具等物体上存在困难
- 将3d特征转化为向量表示，但模型能力取决于向量表示本身

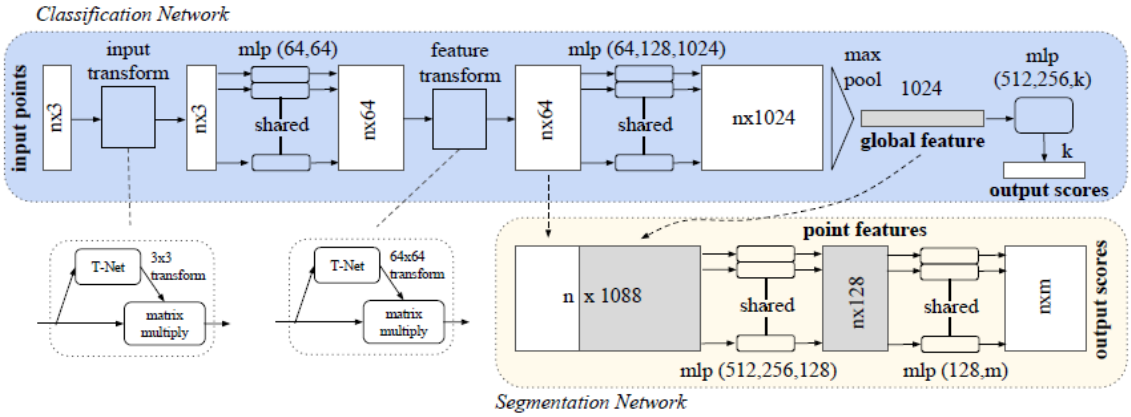
文章提出PointNet可以胜任分类分割等任务，



模型结构如下，其中T-Net为变换网络,结合多个共享权重的MLP提取到排列不变特征，再使用最大值池化提取全局特征。

对于分类任务，将全局特征经过最后一层mlp分类。

对于分割任务，将全局特征和局部特征结合，使用卷积得到每个点的分割结果。



点云数据具有以下特征，

- 无序性
- 结点交互性
- 平移旋转不变性

PoinyNet采用对称函数，最大值池化学习数据的无序性。

文章对比了其他几种无序性学习的方式，如排序，乱序RNN等，均指出其弊端，对比说明最大值池化的好处。

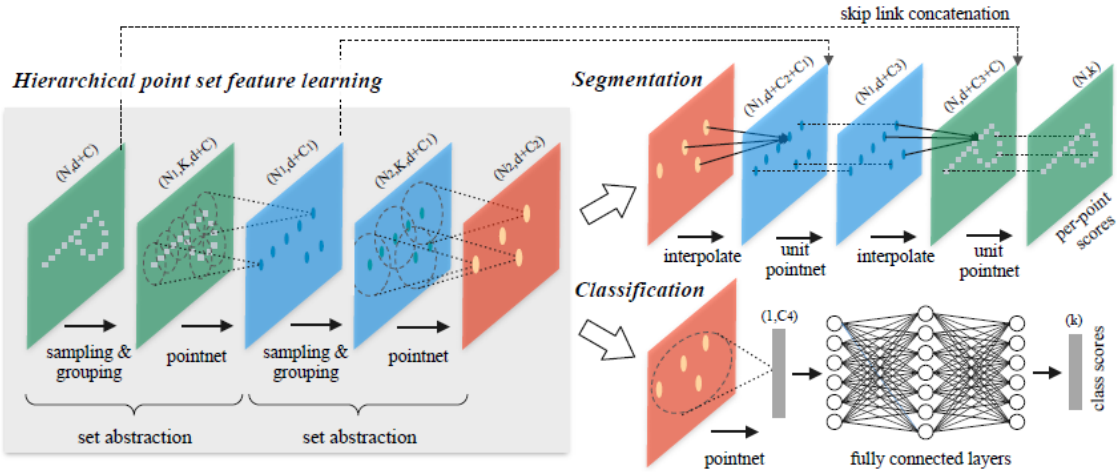
文章使用共享权重的MLP学习局部特征，从而学习结点交互性。对于旋转不变性，比起其他类型的数据，点云数据的平移旋转实现简单，只需要使用T-Net对每个点进行仿射变换。

实验与3dCNN进行对比。在准确率和时间上均大部分超过。

PointNet++

改进PointNet，主要解决PointNet并没有很好地学习到点云在度量空间中的局部特性的问题。并且针对点云数据非均匀分布的特性，提出更为鲁棒的策略。

模型结构如下，下采样部分使用采样和卷积交替进行，对于分类任务将下采样提取的特征经过三层全连接层得到最后每一类的分数，对于分割任务使用上采样得到每个点的分数。



层级特征提取，每一层由采样、分组、MLP层三个步骤组成

采样采用最远点采样（FPS），该方法相对于随机采样更能从样本中得到点与点之间的依存关系。

Sampling layer. Given input points $\{x_1, x_2, ..., x_n\}$, we use iterative farthest point sampling (FPS) to choose a subset of points $\{x_{i_1}, x_{i_2}, ..., x_{i_m}\}$, such that x_{i_j} is the most distant point (in metric distance) from the set $\{x_{i_1}, x_{i_2}, ..., x_{i_{j-1}}\}$ with regard to the rest points. Compared with random sampling, it has better coverage of the entire point set given the same number of centroids. In contrast to CNNs that scan the vector space agnostic of data distribution, our sampling strategy generates receptive fields in a data dependent manner.

根据采样点作为中心点分组，分组有两种方式，基于球半径和K近邻，也即寻找位于以中心点为球心的球内的点或者寻找k近邻点。文章建议用球半径的方式以更好地保持局部空间结构。

Grouping layer. The input to this layer is a point set of size $N \times (d + C)$ and the coordinates of a set of centroids of size $N' \times d$. The output are groups of point sets of size $N' \times K \times (d + C)$, where each group corresponds to a local region and K is the number of points in the neighborhood of centroid points. Note that K varies across groups but the succeeding *PointNet layer* is able to convert flexible number of points into a fixed length local region feature vector.

MLP层，相比于PointNet直接将每个点的坐标作为MLP的2输入，PointNet++中的MLP每个点相对于中心点的相对坐标作为输入，以此捕获点之间的局部相对关系。

PointNet layer. In this layer, the input are N' local regions of points with data size $N' \times K \times (d+C)$. Each local region in the output is abstracted by its centroid and local feature that encodes the centroid's neighborhood. Output data size is $N' \times (d + C')$.

The coordinates of points in a local region are firstly translated into a local frame relative to the centroid point: $x_i^{(j)} = x_i^{(j)} - \hat{x}^{(j)}$ for $i = 1, 2, ..., K$ and $j = 1, 2, ..., d$ where $\hat{x}$ is the coordinate of the centroid. We use PointNet [20] as described in Sec. 3.1 as the basic building block for local pattern learning. By using relative coordinates together with point features we can capture point-to-point relations in the local region.

对于点云数据非均匀分布的特点，采用多尺度采样MSG和多分辨率采样MRG。两种方法的示意图见下，相当于用不同尺度/不同分辨率采样，然后将多次采样结果拼接。文章同时对比了两种方法分别的利弊。

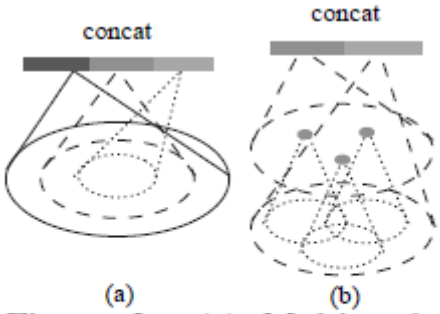
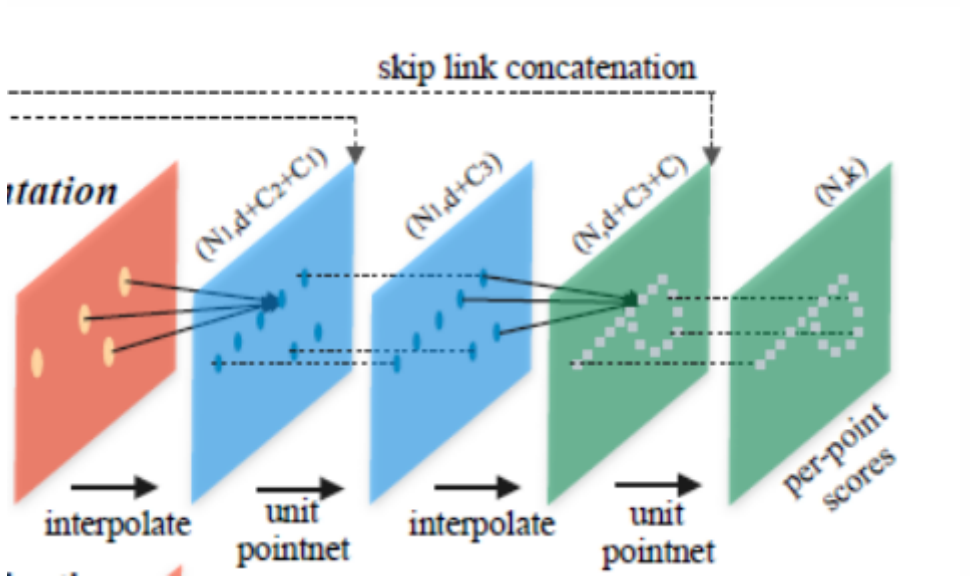


Figure 3: (a) Multi-scale grouping (MSG); (b) Multi-resolution grouping (MRG).

对于分割任务，需要进行上采样，即从采样的部分点的特征推广到每个点的特征，文章采用了特征传播的方法，传播时进行类似1x1卷积的网络层unit pointnet进行卷积操作，然后用如下公式进行插值。



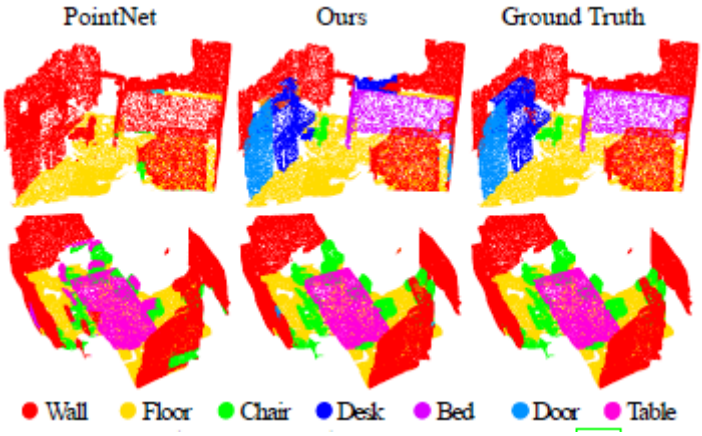
插值公式如下，

$$f^{(j)}(x) = \frac{\sum_{i=1}^{k} w_i(x) f_i^{(j)}}{\sum_{i=1}^{k} w_i(x)} \quad \text{where} \quad w_i(x) = \frac{1}{d(x, x_i)^p}, \; j = 1, ..., C \tag{2}$$

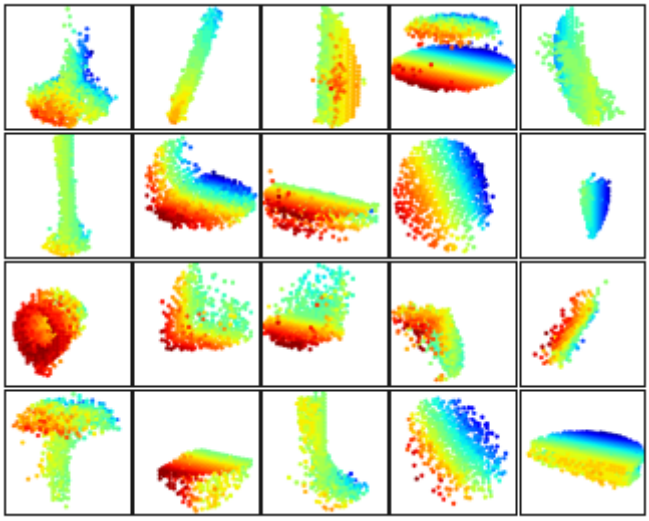
同时采用了跳层连接，以同时获取不同尺度的特征。

该方法可用于体素表示、图片（如MNIST手写数据集)、点云数据（ModelNet40），文章对此与简单的MLP、卷积神经网络、PointNet进行对比。并探究了MSG、MRG等不同技术的作用。

下图展示了PointNet++相对于PointNet不能很好分割的例子的效果。



可视化隐藏层的激活情况，

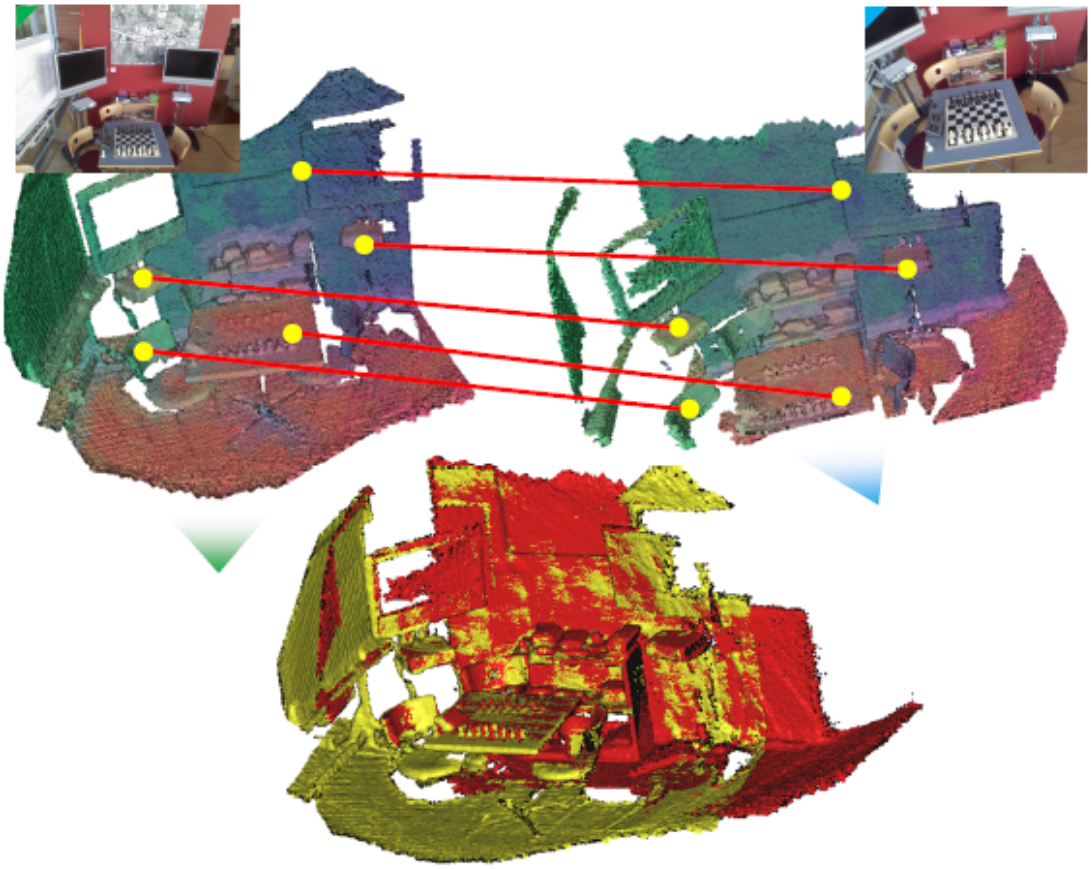


文章同时提及该方法也可以推广到非欧空间上，对流形等结构数据也同样适用。

PPF

PPF(Point Pair Feature) ,尝试获取点对之间的特征对点云数据进行学习。文章指出，PointNet很好地学习到了点云数据的排列不变性、PointNet++学习到了点云密度分布的特征。但PPF学到了点云数据的更多结构性特征。

文章动机可以用下图阐释，学习到了点与点之间的相互关系特征更有利于点云学习。



点云数据具有在刚体变换（平移、旋转）不变的性质，

PPF特征如下定义，其中n为每个点相对于物体表面地法向量，可以证明如下方式定义得到的PPF具有刚体变换下的不变性。

Point Pair Features (PPF) Point pair features are anti-symmetric 4D descriptors, describing the surface of a pair of oriented 3D points $\mathbf{x}_1$ and $\mathbf{x}_2$, constructed as:

$$\psi_{12} = (\|\mathbf{d}\|_2, \angle(\mathbf{n}_1, \mathbf{d}), \angle(\mathbf{n}_2, \mathbf{d}), \angle(\mathbf{n}_1, \mathbf{n}_2)) \quad (4)$$

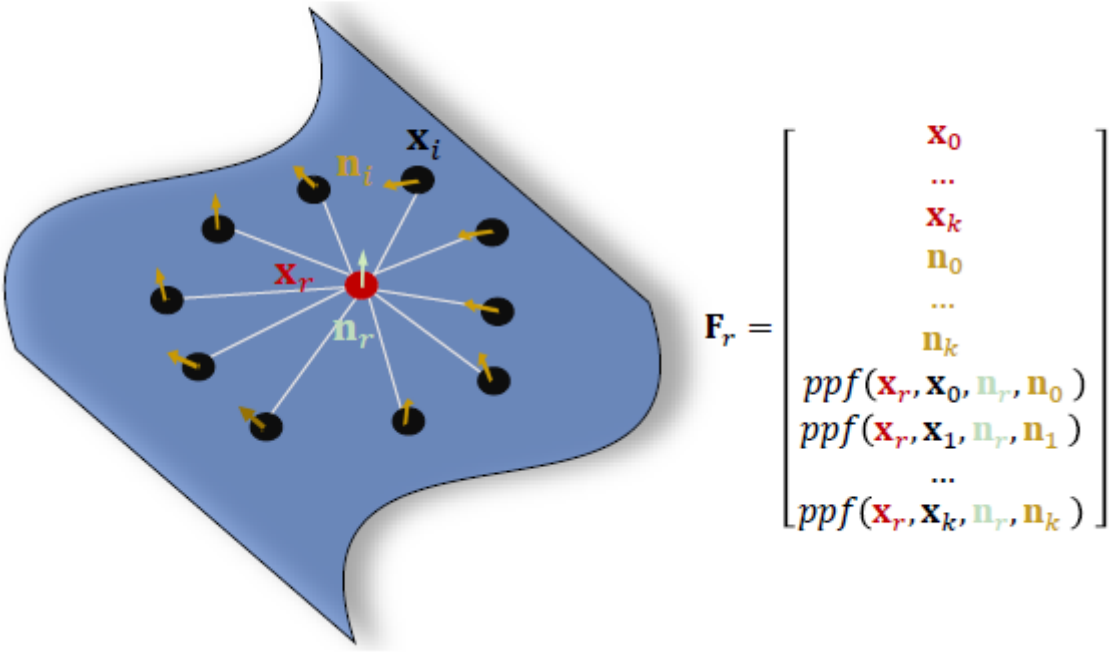
where $\mathbf{d}$ denotes the difference vector between points, $\mathbf{n}_1$ and $\mathbf{n}_2$ are the surface normals at $\mathbf{x}_1$ and $\mathbf{x}_2$. $\|\cdot\|$ is the Euclidean distance and $\angle$ is the angle operator computed in a numerically robust manner as in [2]:

$$\angle(\mathbf{v}_1, \mathbf{v}_2) = \text{atan2}\left(\|\mathbf{v}_1 \times \mathbf{v}_2\|, \mathbf{v}_1 \cdot \mathbf{v}_2\right) \quad (5)$$

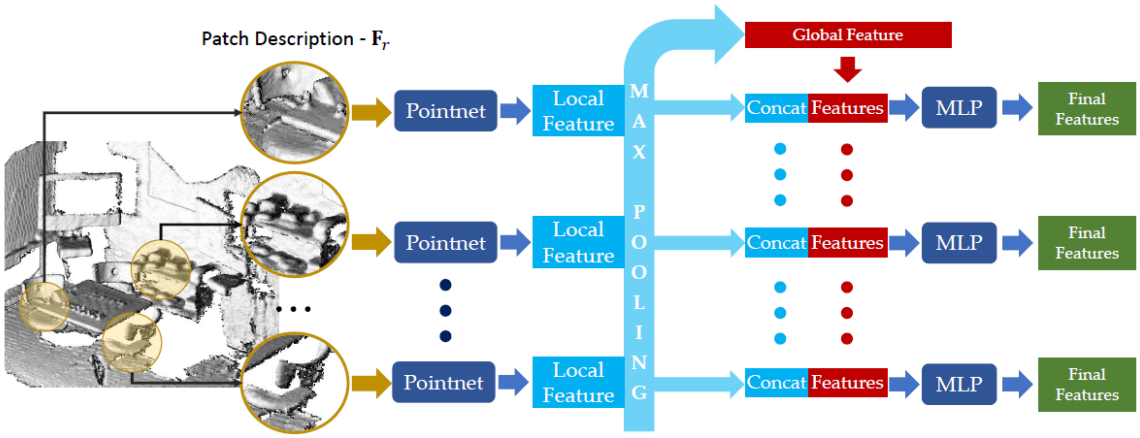
网络将每个结点的位置信息、法向量信息、点对之间的PPF信息作为MLP的输入。

$$\mathbf{F}_r = \{\mathbf{x}_r, \mathbf{n}_r, \mathbf{x}_i, \cdots, \mathbf{n}_i, \cdots, \psi_{ri}, \cdots\} \qquad (6)$$

用下图表示，



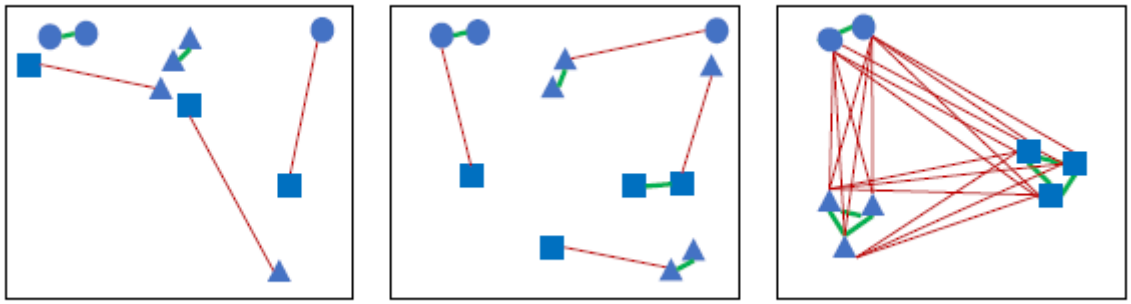
结构如下，与PointNet相同地采用了先提取局部特征，再提取全局特征的方式，区别在于输入的为每个局部的Patch Description，每个Patch Description定义为上述F_R，



网络应当学习到Patch的局部特征，因此希望邻近的patch具有相似的特征，而较远的patch具有差别较大的特征，

常用的有2元组损失、3元组损失等，但文章指出这都可以归结为N元组损失，

NOTE:下图中绿色边表示相近的patch，应该鼓励特征的相似性，而红色相反，



(a) Pair Samples (b) Triplet Samples (c) N-Tuple Configuration

借助哈达玛乘积等数学符号，N元组损失可以简单表达，也即在计算中，将实际的距离通过阈值t二进制化为M，与预测的距离进行计算，

Generalizing these losses to N-patches, we propose N-tuple loss, an N-to-N contrastive loss, to correctly learn to solve this combinatorial problem by catering for the many-to-many relations as depicted in Fig. 4. Given the ground truth transformation $\mathbf{T}$, N-tuple loss operates by constructing a correspondence matrix $\mathbf{M} \in \mathbb{R}^{N \times N}$ on the points of the aligned fragments. $\mathbf{M} = (m_{ij})$ where:

$$m_{ij} = \mathbb{1}(\|\mathbf{x}_i - \mathbf{T}\mathbf{y}_j\|_2 < \tau) \tag{7}$$

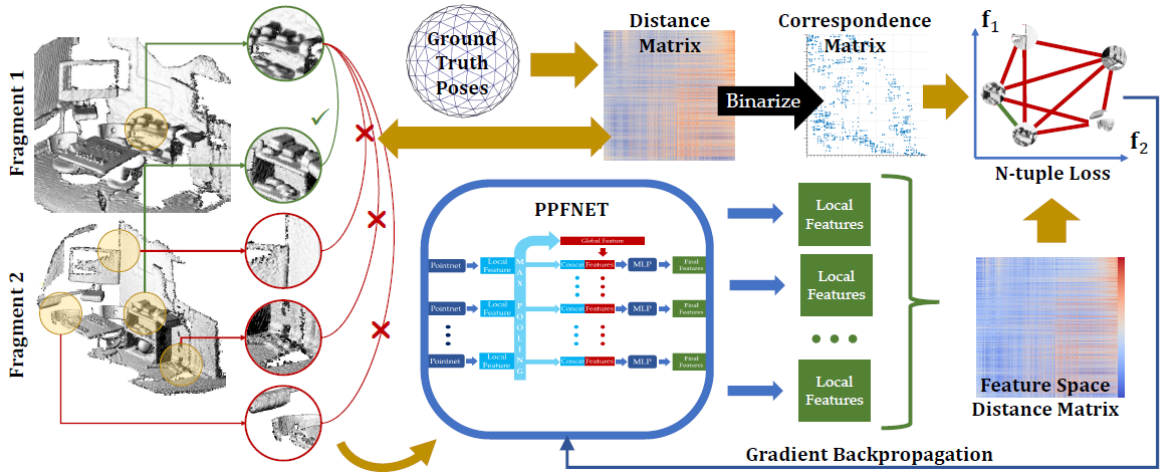
$\mathbb{1}$ is an indicator function. Likewise, we compute a feature-space distance matrix $\mathbf{D} \in \mathbb{R}^{N \times N}$ and $\mathbf{D} = (d_{ij})$ where

$$d_{ij} = \|f(\mathbf{x}_i) - f(\mathbf{y}_j)\|_2 \tag{8}$$

The N-tuple loss then functions on the two distance matrices solving the correspondence problem. For simplisity of expression, we define an operation $\sum^*(\cdot)$ to sum up all the elements in a matrix. N-tuple loss can be written as:

$$L = \sum{}^* \left(\frac{\mathbf{M} \circ \mathbf{D}}{\|\mathbf{M}\|_2^2} + \alpha \frac{max(\theta - (1 - \mathbf{M}) \circ \mathbf{D}, 0)}{N^2 - \|\mathbf{M}\|_2^2} \right) \tag{9}$$

整体结构表示如下，

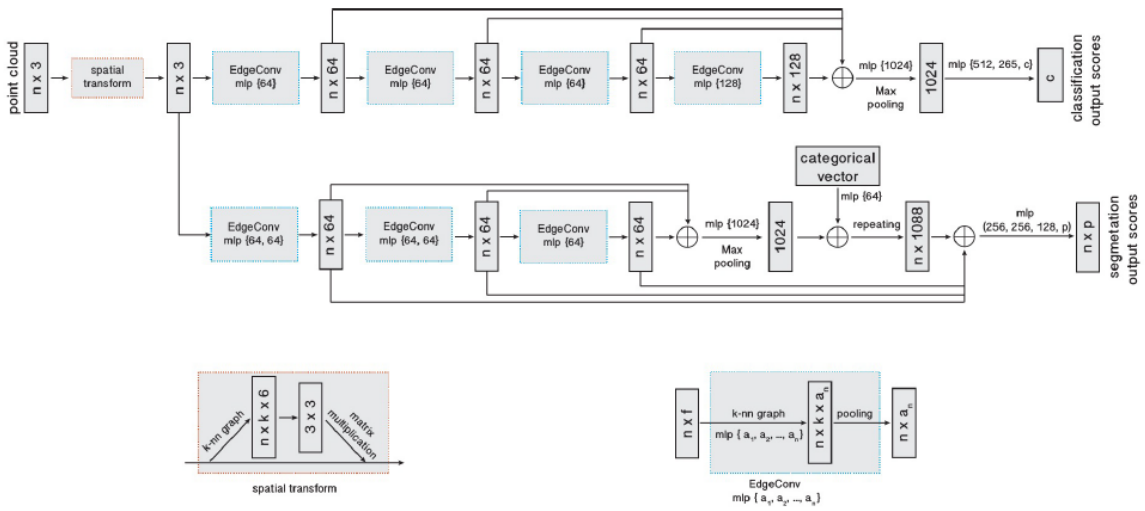


总结，PFF利用PPF特征获取局部信息，通过N元组损失获取全局信息，并用实验证明，将两者运用于简单的PointNet上均能提升模型的效果，而最好的模型是结合使用所有技术的模型。

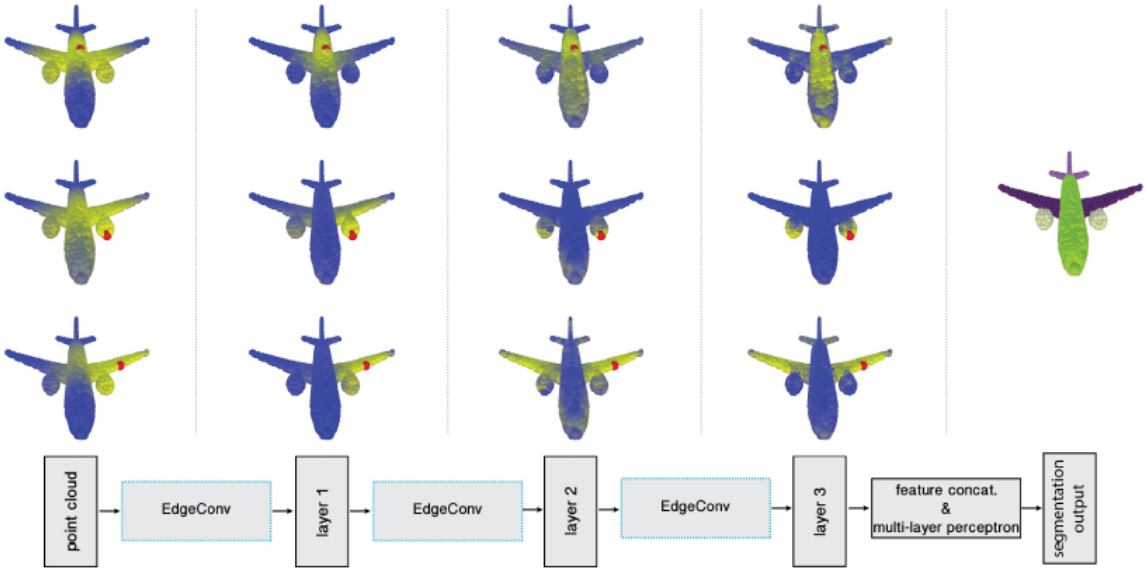
DGCNN

动态图卷积模型，总结前人的方法提出通用性的EdgeConv，指出该方法可以很好地学习局部特征，同时堆叠起来学习全局特征。

模型结构和PointNet基本一致，区别是用EdgeConv取代了PointNet中的MLP。与PointNet使用全部节点、PointNet++使用采样的部分结点的静态图方法不同，DGCNN基于每一层的特征动态构建k近邻图，进行空间变换和边卷积操作。



下图形象展示了动态近邻图，红色结点为中心点，黄色结点表示在特征空间上与中心点更接近的结点。



该图展示了EdgeConv，

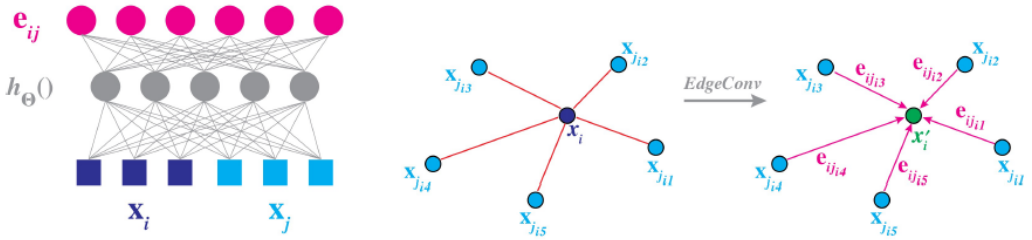


Fig. 2. **Left**: Computing an edge feature, $\mathbf{e}_{ij}$ (top), from a point pair, $\mathbf{x}_i$ and $\mathbf{x}_j$ (bottom). In this example, $h_\Theta()$ is instantiated using a fully connected layer, and the learnable parameters are its associated weights. **Right**: The EdgeConv operation. The output of EdgeConv is calculated by aggregating the edge features associated with all the edges emanating from each connected vertex.

EdgeConve可以写为如下公式，方框可为Sum或Max操作，

$$\mathbf{x}_i' = \square_{j:(i,j)\in\mathcal{E}} h_\Theta(\mathbf{x}_i, \mathbf{x}_j). \qquad (1)$$

而PointNet，PointNet++中使用的方法其实均为EdgeConv的特例，

DGCNN is related to two classes of approaches, PointNet and graph CNNs, which we show to be particular settings of our method. We summarize different methods in Table 1.

PointNet is a special case of our method with $k = 1$, yielding a graph with an empty edge set $\mathcal{E} = \varnothing$. The edge function used in PointNet is $h_{\Theta}(\mathrm{x}_i, \mathrm{x}_j) = h_{\Theta}(\mathrm{x}_i)$, which considers global but not local geometry. PointNet++ tries to account for local structure by applying PointNet in a local manner. In our parlance, PointNet++ first constructs the graph according to the Euclidean distances between the points, and in each layer applies a graph coarsening operation. For each layer, some points are selected using farthest point sampling (FPS); only the selected points are preserved while others are directly discarded after this layer. In this way, the graph becomes smaller after the operation applied on each layer. In contrast to DGCNN, PointNet++ computes pairwise distances using point input coordinates, and hence their graphs are fixed during training. The edge function used by PointNet++ is $h_{\Theta}(\mathrm{x}_i, \mathrm{x}_j) = h_{\Theta}(\mathrm{x}_j)$, and the aggregation operation is also a max.

其他的一些方法也可覆盖在EdgeConv的框架下，文章建议使用如下地EdgeConv，同时获取结点绝对特征和相对特征，

$$h_{\Theta}(\mathrm{x}_i, \mathrm{x}_j) = \bar{h}_{\Theta}(\mathrm{x}_i, \mathrm{x}_j - \mathrm{x}_i). \tag{7}$$

对于分割任务，与PointNet进行了对比，部分结果可视化，

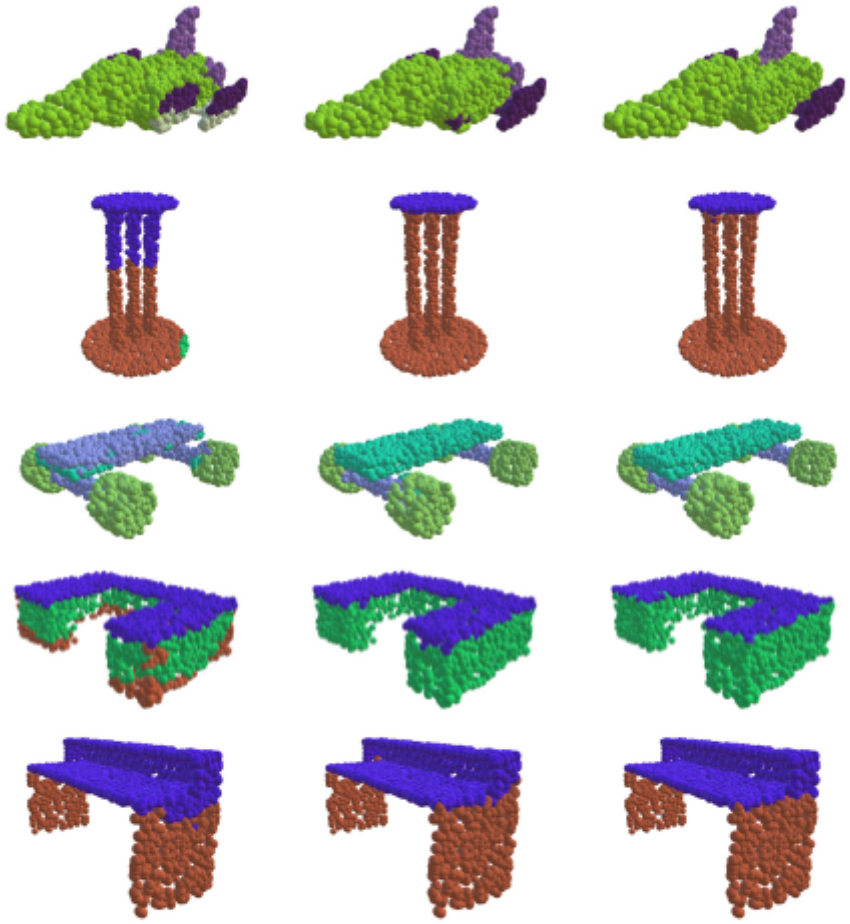
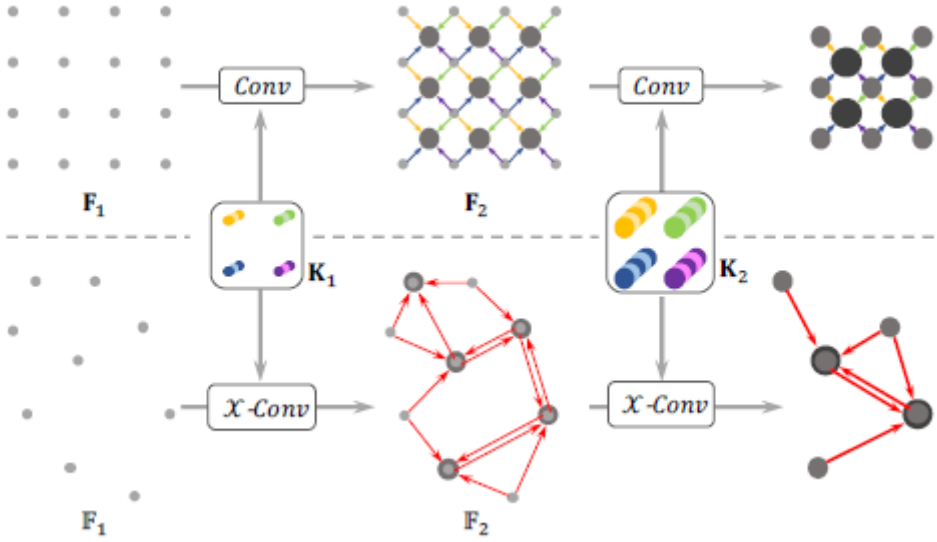


Fig. 7. Compare part segmentation results. For each set, from left to right: PointNet, ours and ground truth.

PointCNN

提出X-Conv，基于X变换的卷积操作，用于处理点云数据。证明了X-变换可以起到空间转换网络的作用，同时可以同时起到分配权重和排列的作用。

将图像卷积推广到图结构卷积，



X-Conv考虑了结点的特征和相对位置，

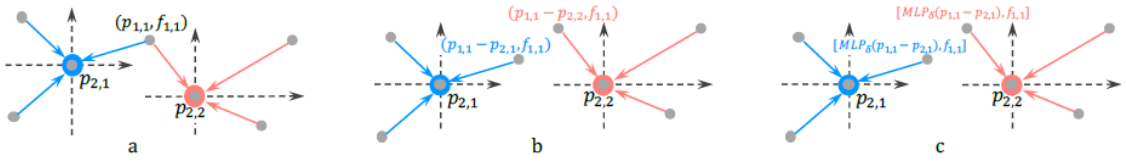
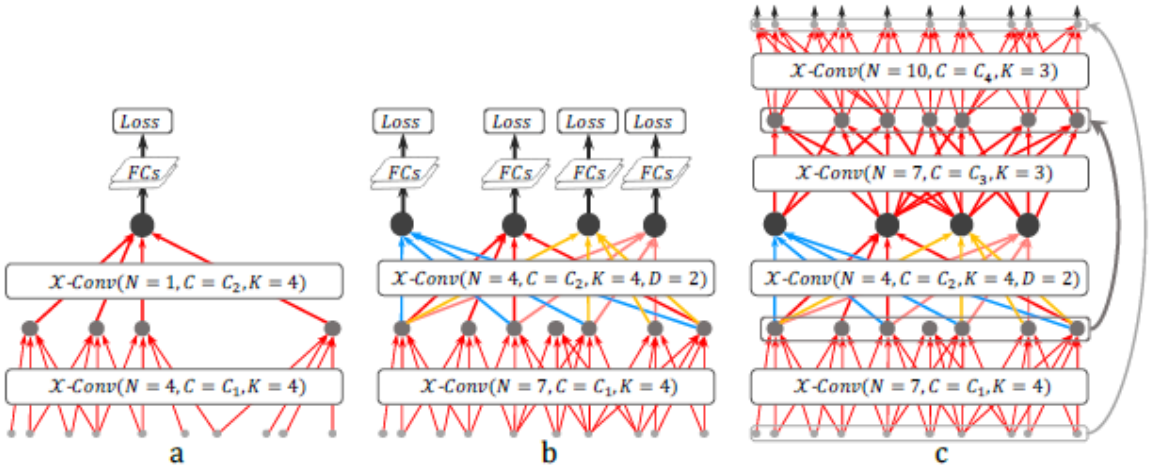


Figure 3: The process for converting point coordinates to features. Neighboring points are transformed to the local coordinate systems of the representative points (a and b). The local coordinates of each point are then individually lifted and combined with the associated features (c).

原文公式，K为卷积核、MLP为多层感知机、P为点位置、p为卷积的结点、F为结点特征。

$$\mathbf{F}_p = \mathcal{X}\text{-}Conv(\mathbf{K}, p, \mathbf{P}, \mathbf{F}) = Conv(\mathbf{K}, MLP(\mathbf{P} - p) \times [MLP_\delta(\mathbf{P} - p), \mathbf{F}]), \qquad (2)$$

网络整体框架图，



文章针对ModelNet40做了实验，证明方法优于PointNet等方法。

PyG代码中类定义为，

CLASS **XConv** (**in_channels**: int, **out_channels**: int, **dim**: int, **kernel_size**: int, **hidden_channels**: Optional[int] = None, **dilation**: int = 1, **bias**: bool = True, **num_workers**: int = 1) [source]

The convolutional operator on $\mathcal{X}$-transformed points from the "PointCNN: Convolution On X-Transformed Points" paper

$$\mathbf{x}_i' = \mathrm{Conv}\left(\mathbf{K}, \gamma_{\mathbf{\Theta}}(\mathbf{P}_i - \mathbf{p}_i) \times \left(h_{\mathbf{\Theta}}(\mathbf{P}_i - \mathbf{p}_i) \,\|\, \mathbf{x}_i\right)\right),$$

where $\mathbf{K}$ and $\mathbf{P}_i$ denote the trainable filter and neighboring point positions of $\mathbf{x}_i$, respectively. $\gamma_{\mathbf{\Theta}}$ and $h_{\mathbf{\Theta}}$ describe neural networks, *i.e.* MLPs, where $h_{\mathbf{\Theta}}$ individually lifts each point into a higher-dimensional space, and $\gamma_{\mathbf{\Theta}}$ computes the $\mathcal{X}$- transformation matrix based on *all* points in a neighborhood.