【CVPR 2020】用于点云中 3D 对象检测的图神经网络

论文原文：https://arxiv.org/pdf/2003.01Point-GNN251.pdf

论文背景

本文提出了一个 GNN 用于从 LiDAR 电云中发现对象，为此，本文在固定半径的近邻图中有效地编码了点云，使用 Point-GNN 预测每个点的对象的类别和形状。在 Point-GNN 中，本文提出了一种自动注册机制来减少平移差异，并且还设计了一种盒合并和计分操作，以准确地组合来自多个顶点的检测。 本文在 KITTI 基准上进行的实验表明，所提出的方法仅使用点云即可达到领先的准确性，甚至可以超越基于融合的算法。 我们的结果证明了使用图神经网络作为 3D 对象检测的新方法的潜力。

理解 3D 环境对于机器人感知十分重要，从点云中识别物体对于如自动驾驶之类的应用很有帮助。

CNN 依赖卷积操作识别物体，卷积操作虽然有效，但需要网格化的输入，但点云相较于图片来说更加稀疏并且在网格中分布不均。将点云放置在常规网格上会在网格单元中生成数量不均匀的点。 在这样的网格上应用相同的卷积运算会导致拥挤的单元中潜在的信息丢失或空单元中的计算浪费。

最近的一些工作尝试使用无序点集作为输入，这样无需将点云转换为网格。但这样做通常需要迭代采样和分组来创建点集。在大的点云上重复分组和采样可能会在计算上造成高昂的成本，因此最近的一些 3D 检测方法通常采用混合的方法在不同阶段分别使用网格或集合表示，但这种混合的策略可能会同时受到这两种表示方法的限制。

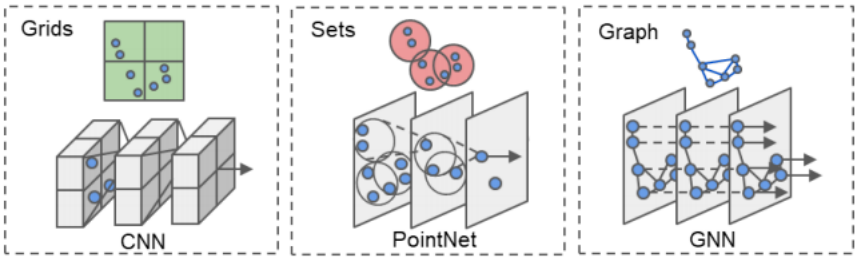

Figure 1. Three point cloud representations and their common processing methods.

本文提出用图的形式来表示点云，并且设计了 Point-GNN 来进行目标检测。具体来说，将点云中的点作为图的顶点，并与周围一定半径内的其他点进行连边，从而允许特征在邻居之间流动。这样的图表示可以直接适应点云的结构而无需将其转化为其他的形式。GNN 在每层中会重新使用图中的边，以避免重复对点进行分组和采样。本文提出的 Point-GNN 将点云作为输入，输出每个顶点所属的对象类别和边界框，从而一次性检出多个物体，同时引入了一种自动注册机制，以根据特征自动对齐坐标，设计了框合并以及积分操作，以准确的组合来自多个顶点的检测结果。

本文的主要贡献有以下几个方面：

1. 提出了一种使用 GNN 的点云物体检测方法
2. 使用带有自动注册机制的 Point-GNN 实现一次检测多个物体
3. 在 KITTI benchmark 上达到了 SOTA 的 3D 对象检测精确度，并深入分析了每个组件的有效性

论文模型

模型的整体结构如下图所示，分为三个组件：1)图构建；2) T 次迭代的 GNN；3)边界框合并和评分

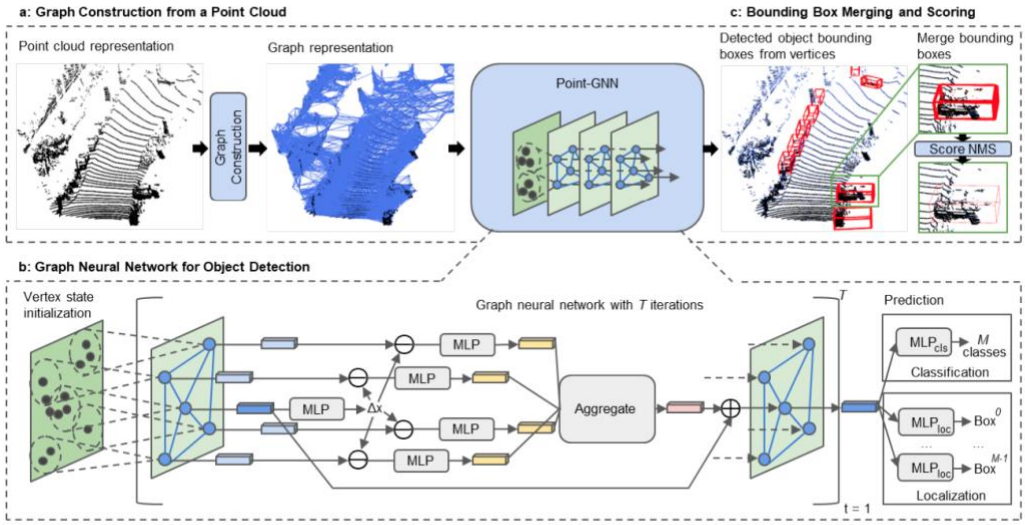

Figure 2. The architecture of the proposed approach. It has three main components: (a) graph construction from a point cloud, (b) a graph neural network for object detection, and (c) bounding box merging and scoring.

图构建

定义一个包括 N 个点的点云 $P = \{p_1, \ldots, p_N\}$，其中 $p_i = (x_i, s_i)$ 是点的三维坐标以及用于表示点的属性的长度为 k 的向量 s。

给定一个点云 P，我们可以使用这些顶点 ，并将在这些点半径为 r 的以内点连接起来得到图 $G=(P,E)$，其中 $E=\{(p_i,p_j)\mid \|x_i-x_j\|_2<r\}$

实际上，一个点云通常包含数以万计的点。以所有点为顶点构造图会带来很大的计算负担。因此，本文将体素降采样点云 $\hat{P}$ 用于图的构建。 必须注意的是，此处的体素仅用于降低点云的密度，而不用作点云的表示形式。

带自注册机制的 GNN

GNN 通过边聚合特征来更新顶点特征，在第 t+1 次迭代中，顶点特征更新方式如下：

$$v_i^{t+1}=g^t(\rho(\{e_{ij}^t\mid(i,j)\in E\}),v_i^t)$$
$$e_{ij}^t=f^t(v_i^t,v_j^t)$$

其中 eˆt 和 vˆt 分别是第 t 次迭代的边和顶点特征，$\widetilde{f^t(.)}$ 用于通过所连接的两个节点计算边的特征，$\rho(.)$ 用于为每个顶点聚合边的特征，$g^t(.)$ 将每个顶点聚合后的边的特征用于更新这个节点的特征。

对于目标检测这个任务来说，本文提出的 GNN 可以将顶点所属的物体信息也用于更新顶点表示：

$$s_i^{t+1}=g^t(\rho(\{f^t(x_j-x_i,s_j^t)\mid(i,j)\in E\}),s_i^t)$$

对于边的特征提取，本文使用邻居的相对坐标作为 $f^t(.)$ 的输入。相对坐标保证了点云的平移不变性，但其仍然对邻域内的平移敏感，当顶点产生较小的平移时，其邻居的局部结构保持相似，但邻居的相对坐标都产生了变化，这会影响到 $f^t(.)$ 的值，为了减少这种变化，本文按照邻居的结构特征而不是中心顶点的坐标对齐邻居的坐标。因为中心顶点已经包含了先前迭代中的一些结构特征，我们可以使用它来预测对齐便宜，并提出一种自动注册机制：

$$\Delta x_i{}^t=h^t(s_i^t)$$
$$s_i^{t+1}=g^t(\rho(\{f(x_j-x_i+\Delta x_i{}^t,s_j^t)\},s_i^t)$$

Δx_i^t 是顶点注册坐标的坐标偏移量，$h^t(.)$ 使用上一次迭代的中心顶点状态值来计算偏移量，通过设置 $h^t(.)$ 使得输出为 0，GNN 可以在必要的时候防止偏移量。

作者将上诉的函数设置为多层感知机(MLP)和 max，具体表示为:

$$\Delta x_i{}^t = MLP_h^t(s_i^t)$$
$$e_{ij}^t = MLP_f^t([x_j - x_i + \Delta x_i{}^t, s_j^t])$$
$$s_i^{t+1} = MLP_g^t(Max(\{e_{ij} \mid (i,j) \in E\})) + s_i^t$$

可以看出边的特征和顶点的属性都是采用的 MLP 去学习的

Loss

对于目标分类，每个顶点都会计算多累概率分布 $\{p_{c_1}, ..., p_{c_M}\}$，M 是目标类的总数，包括背景类，如果顶点在目标的边界框内，则将目标类分配给该顶点，否则将其分配为背景类，这部分使用交叉熵作为 loss。

$$l_{cls} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{c_j}^i log(p_{c_j}^i)$$

对于目标边界框，作者使用 $\tilde{b} = (x, y, z, l, h, w, \theta)$ 对其进行预测，其中(x, y, z)表示目标框的中心位置，(l, h, w)表示框的长度高度和宽度，θ 表示偏航角，使用顶点坐标 (x_v, y_v, z_v) 将目标框编码为：

$$\delta_x = \frac{x - x_v}{l_m},\ \delta_y = \frac{y - y_v}{h_m},\ \delta_z = \frac{z - z_v}{w_m}$$
$$\delta_l = log(\frac{l}{l_m}),\ \delta_h = log(\frac{h}{h_m}),\ \delta_w = log(\frac{w}{w_m})$$
$$\delta_\theta = \frac{\theta - \theta_0}{\theta_m}$$

如果一个顶点在框中，那么采用 Huber loss，否则将其 loss 设置为 0，最后将所有 loss 求一个平均：

$$l_{loc} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(v_i \in b_{interest}) \sum_{\delta \in \delta_{b_i}} l_{huber}(\delta - \delta^{gt})$$

为了防止过拟合，作者加入了 L1 正则化：

$$l_{total} = \alpha l_{cls} + \beta l_{loc} + \gamma l_{reg}$$

这里的 l_{reg}应该就是以往的 smooth l1

边界框合并和评分

由于多个顶点可以位于同一对象上，因此神经网络可以输出同一对象的多个边界框。 必须将这些边界框合并为一个，并分配一个置信度分数。 非最大抑制（NMS）已被广泛用于此目的。 通常的做法是选择具有最高分类分数的框，然后隐藏其他重叠的框。 但是，分类分数并不总是反映定位质量。 明显地，部分被遮挡的物体可能具有指示该物体的类型的强烈线索，但是缺乏足够的形状信息。 标准 NMS 可能仅基于分类分数就选择了不准确的边界框。所以作者在合并的过程中同时考虑到了重叠边界框的中位数位置和大小：

Algorithm 1: NMS with Box Merging and Scoring

Input: $\mathcal{B} = \{b_1, ..., b_n\}, \mathcal{D} = \{d_1, ..., d_n\}, T_h$
$\mathcal{B}$ is the set of detected bounding boxes.
$\mathcal{D}$ is the corresponding detection scores.
T_h is an overlapping threshold value.
Green color marks the main modifications.

$\mathcal{M} \leftarrow \{\}, \mathcal{Z} \leftarrow \{\}$
while $\mathcal{B} \neq empty$ **do**
  $i \leftarrow argmax\ D$
  $\mathcal{L} \leftarrow \{\}$
  for b_j *in* $\mathcal{B}$ **do**
    if $iou(b_i, b_j) > T_h$ **then**
      $\mathcal{L} \leftarrow \mathcal{L} \cup b_j$
      $\mathcal{B} \leftarrow \mathcal{B} - b_j, \mathcal{D} \leftarrow \mathcal{D} - d_j$
    end
  end
  $m \leftarrow median(\mathcal{L})$
  $o \leftarrow occlusion(m)$
  $z \leftarrow (o+1) \sum_{b_k \in \mathcal{L}} IoU(m, b_k) d_k$
  $\mathcal{M} \leftarrow \mathcal{M} \cup m, \mathcal{Z} \leftarrow \mathcal{Z} \cup z$
end
return $\mathcal{M}, \mathcal{Z}$

$$o_i = \frac{1}{l_i w_i h_i} \prod_{v \in \{v_i^l, v_i^w, v_i^h\}} \max_{p_j \in b_i} (v^T x_j) - \min_{p_j \in b_i} (v^T x_j)$$

实验

Method	Modality	Car			Pedestrian			Cyclist		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
UberATG-ContFuse[12]	LiDAR + Image	82.54	66.22	64.04	N/A	N/A	N/A	N/A	N/A	N/A
AVOD-FPN[8]	LiDAR + Image	81.94	71.88	66.38	50.80	42.81	40.88	64.00	52.18	46.61
F-PointNet[13]	LiDAR + Image	81.20	70.39	62.19	51.21	44.89	40.23	71.96	56.77	50.39
UberATG-MMF[11]	LiDAR + Image	86.81	76.75	68.41	N/A	N/A	N/A	N/A	N/A	N/A
VoxelNet[23]	LiDAR	81.97	65.46	62.85	**57.86**	**53.42**	**48.87**	67.17	47.65	45.11
SECOND[19]	LiDAR	83.13	73.66	66.20	51.07	42.56	37.29	70.51	53.85	53.85
PointPillars[10]	LiDAR	79.05	74.99	68.30	52.08	43.53	41.49	75.78	59.07	52.92
PointRCNN[16]	LiDAR	85.94	75.76	68.32	49.43	41.78	38.63	73.93	59.60	53.59
STD[21]	LiDAR	86.61	77.63	**76.06**	53.08	44.24	41.97	**78.89**	62.53	55.77
Our Point-GNN	LiDAR	**88.33**	**79.47**	72.29	51.92	43.77	40.14	78.60	**63.48**	**57.08**

Table 1. The Average Precision (AP) comparison of 3D object detection on the KITTI *test* dataset.

Method	Modality	Car			Pedestrian			Cyclist		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
UberATG-ContFuse[12]	LiDAR + Image	88.81	85.83	77.33	N/A	N/A	N/A	N/A	N/A	N/A
AVOD-FPN[8]	LiDAR + Image	88.53	83.79	77.9	58.75	51.05	47.54	68.06	57.48	50.77
F-PointNet[13]	LiDAR + Image	88.70	84 .00	75.33	58.09	50.22	47.20	75.38	61.96	54.68
UberATG-MMF[11]	LiDAR + Image	89.49	87.47	79.10	N/A	N/A	N/A	N/A	N/A	N/A
VoxelNet[23]	LiDAR	89.60	84.81	78.57	**65.95**	**61.05**	**56.98**	74.41	52.18	50.49
SECOND[19]	LiDAR	88.07	79.37	77.95	55.10	46.27	44.76	73.67	56.04	48.78
PointPillars[10]	LiDAR	88.35	86.10	79.83	58.66	50.23	47.19	79.14	62.25	56.00
STD[21]	LiDAR	89.66	87.76	**86.89**	60.99	51.39	45.89	81.04	65.32	57.85
Our Point-GNN	LiDAR	**93.11**	**89.17**	83.9	55.36	47.07	44.61	**81.17**	**67.28**	**59.67**

Table 2. The Average Precision (AP) comparison of Bird's Eye View (BEV) object detection on the KITTI *test* dataset.

	Box Merge	Box Score	Auto Reg.	BEV AP (Car)			3D AP (Car)		
				Easy	Moderate	Hard	Easy	Moderate	Hard
1	-	-	-	89.11	87.14	86.18	85.46	76.80	74.89
2	-	-	✓	89.03	87.43	86.39	85.58	76.98	75.69
3	✓	-	✓	89.33	87.83	86.63	86.59	77.49	76.35
4	-	✓	✓	89.60	88.02	86.97	87.40	77.90	76.75
5	✓	✓	-	90.03	88.27	87.12	88.16	78.40	77.49
6	✓	✓	✓	89.82	88.31	87.16	87.89	78.34	77.38

Table 3. Ablation study on the *val.* split of KITTI data.

结论

本文提出了一种名为 Point-GNN 的图神经网络，用于从点云的图形表示中检测 3D 对象。 通过图表示紧凑地对点云进行编码，而无需映射到网格或重复采样和分组。Point-GNN 在 KITTI benchmark 的 3D 和鸟瞰图对象检测方面均达到领先的精度。实验表明，提出的自动配准机制减少了过渡方差，并且在框合并和评分操作提高了检测精度。