Dynamic Graph CNN for learning on point clouds
Wang Yue, et al.

Otakar Jašek

March 25, 2019
Point cloud learning – history

- Hand-crafted features
- View-based methods
- Voxel representation
- PointNet
- Geometric deep learning
Point cloud characteristics

- **Unordered** set of points
- Varying number of points
- Invariant to rigid transformations
- Often no features, location only
To combat first two points, we need a function, that is:

- invariant to permutations
- capable of taking any number of parameters

---


What to do

To combat first two points, we need a function, that is:

- invariant to permutations
- capable of taking any number of parameters

For example $\sum$ or max

Spatial transformer network\(^1\) should solve rigid transformation invariance

---


What to do

To combat first two points, we need a function, that is:

- invariant to permutations
- capable of taking any number of parameters

For example $\sum$ or $\max$

Spatial transformer network\(^1\) should solve rigid transformation invariance

And if we combine these two important ingredients together, we get a PointNet\(^2\)

---


It does not capture neighborhood information
Sample $n$ points at coarse scale, as far from each other as possible

Generate "global" features on local neighborhood of selected points by PointNet

Concatenate these features and repeat for finer scale and larger $n$
PointNet++

- Sample $n$ points at coarse scale, as far from each other as possible
- Generate "global" features on local neighborhood of selected points by PointNet
- Concatenate these features and repeat for finer scale and larger $n$
- Really slow due to sampling of neighborhood and running multiple PointNets
EdgeConv

Generalization of convolution on graph

$$\text{edge\_conv}(x_i) = \bigoplus_{j: (j, i) \in \varepsilon} \theta_j h(x_i, x_j)$$

where

- $\varepsilon$ is a set of edges of the graph
- $\theta_j$ are parameters of the EdgeConv layer
- $\bigoplus$ is aggregation function (usually $\Sigma$, $\max$)
- $h(x_i, x_j)$ is a feature function
Resulting operation largely depends on choice of $\square$ and $h(\cdot, \cdot)$

- $\square = \sum, h(x_i, x_j) = x_j$ is classical convolution (sum over weighted neighbors)
- $h(x_i, x_j) = x_i$ is PointNet-style convolution. It captures only global information
- $h(x_i, x_j) = x_j - x_i$ is another reasonable choice, encoding locality (and disregarding global information)
- Authors of the papers decided, that $h(x_i, x_j) = (x_i, x_j - x_i), \square = \max$
Dynamic graph computation

- Second key idea is to recompute graph using kNN after every EdgeConv
- First EdgeConv is operating on spatially near points, while subsequent on semantically near points.
Full architecture

Dynamic Graph CNN for learning on point clouds

Otakar Jařík
### Classification results

<table>
<thead>
<tr>
<th>CENT</th>
<th>DYN</th>
<th>XFORM</th>
<th>Mean Class Accuracy (%)</th>
<th>Overall Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td></td>
<td></td>
<td>88.8</td>
<td>91.2</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td></td>
<td>88.8</td>
<td>91.5</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>x</td>
<td>89.6</td>
<td>91.9</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>89.8</td>
<td>91.9</td>
</tr>
<tr>
<td>x</td>
<td>x</td>
<td>x</td>
<td>90.2</td>
<td>92.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of nearest neighbors (K)</th>
<th>Mean Class Accuracy (%)</th>
<th>Overall Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>88.0</td>
<td>90.5</td>
</tr>
<tr>
<td>10</td>
<td>88.8</td>
<td>91.4</td>
</tr>
<tr>
<td>20</td>
<td>90.2</td>
<td>92.2</td>
</tr>
<tr>
<td>40</td>
<td>89.2</td>
<td>91.7</td>
</tr>
</tbody>
</table>
### Classification results continued

<table>
<thead>
<tr>
<th>Model</th>
<th>Mean Class Accuracy</th>
<th>Overall Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3DShapeNets [54]</td>
<td>77.3</td>
<td>84.7</td>
</tr>
<tr>
<td>VoxNet [30]</td>
<td>83.0</td>
<td>85.9</td>
</tr>
<tr>
<td>Subvolume [35]</td>
<td>86.0</td>
<td>89.2</td>
</tr>
<tr>
<td>ECC [45]</td>
<td>83.2</td>
<td>87.4</td>
</tr>
<tr>
<td>PointNet [34]</td>
<td>86.0</td>
<td>89.2</td>
</tr>
<tr>
<td>PointNet++ [36]</td>
<td>-</td>
<td>90.7</td>
</tr>
<tr>
<td>KD-Net (depth 10) [20]</td>
<td>-</td>
<td>90.6</td>
</tr>
<tr>
<td>KD-Net (depth 15) [20]</td>
<td>-</td>
<td>91.8</td>
</tr>
<tr>
<td>Ours (baseline)</td>
<td>88.8</td>
<td>91.2</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>90.2</strong></td>
<td><strong>92.2</strong></td>
</tr>
</tbody>
</table>
### Segmentation results

<table>
<thead>
<tr>
<th>Method</th>
<th>Mean IoU</th>
<th>Overall Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PointNet (Baseline) [34]</td>
<td>20.1</td>
<td>53.2</td>
</tr>
<tr>
<td>PointNet [34]</td>
<td>47.6</td>
<td>78.5</td>
</tr>
<tr>
<td>MS + CU(2) [12]</td>
<td>47.8</td>
<td>79.2</td>
</tr>
<tr>
<td>G + RCU [12]</td>
<td>49.7</td>
<td>81.1</td>
</tr>
<tr>
<td>Ours</td>
<td>56.1</td>
<td>84.1</td>
</tr>
</tbody>
</table>

Otakar Jašek

Dynamic Graph CNN for learning on point clouds
Part segmentation

PointNet  
Ours  
Ground truth

Otakar Jašek  Dynamic Graph CNN for learning on point clouds
Semantic nearness

Otakar Jašek

Dynamic Graph CNN for learning on point clouds
Semantic nearness across objects

Figure 9. Visualize the Euclidean distance (yellow: near, blue: far)

Source points  Other point clouds from the same category

Otakar Jašek  Dynamic Graph CNN for learning on point clouds
Generating normals

Otakar Jašek

Dynamic Graph CNN for learning on point clouds