Towards Sparse Hierarchical Graph Classifiers

Cătălina Cangea*, Petar Veličković*, Nikola Jovanović, Thomas Kipf, Pietro Liò

NIPS 2018 Relational Representation Learning Workshop [arxiv.org/abs/1811.01287]
Graph classification

- Typical machine learning task aimed at graph-structured data (e.g. want to classify molecular structures, citation networks, social circles)

- Graphs represent the *generalization* of images (grid-like graphs) → aim to extend readily established *image classification* techniques

- Typical image classification pipeline: alternate between convolution (feature detection) and pooling (down-sampling)—want this for graphs!
Existing work

- **Graph convolutional layers: plenty!**

- **Pooling mechanisms: comparatively few...**
  - Globally pool after each/the final message-passing step
  - *Progressively coarsen* the graph—most approaches assume a pre-defined assignment of nodes to clusters
  - Learn a differentiable pooling mechanism based on the *structure of the data*: Ying et al.(2018), Anon.(2018)
Hierarchical (pooling) strategies

- **DiffPool**: compute *soft clustering* assignments of nodes from the original graph to nodes from the resulting graph
- End-to-end trainable architecture with SOTA results
- Main issue: soft clustering assignments require storing the assignment matrix and therefore $O(kV^2)$ memory!

- **Graph U-Net**: does not suffer from the issue above, but the architecture was not evaluated on classification task benchmarks
Our contributions

- An intuitive architecture for graph classification that closely resembles the computation model for image classification
- Results comparable to state-of-the-art on classification benchmark tasks
- A drastic reduction in the GPU memory requirement (from $O(V^2)$ to $O(V + E)$)
Model

- Input graph → a matrix of node features, $X \in \mathbb{R}^{N \times F}$, and an adjacency matrix, $A \in \mathbb{R}^{N \times N}$

- $A$ is binary and symmetric

- If the graph is featureless, use 1-hot encoding of node degree information to construct $X$

- A CNN-inspired network for graph classification should contain the following layers: convolutional, pooling, readout (i.e. flattening layer in an image CNN used for the final prediction)
Figure 1: The full pipeline of our model (for $k = 0.5$), leveraging several stacks of interleaved convolutional/pooling layers (that, unlike DiffPool, drop rather than aggregate nodes), as well as a JK-net-style summary, combining information at different scales.
Convolutional layer

- Apply the *mean-pooling* propagation rule (as in GCN or Const-GAT):

\[
MP(X, A) = \sigma \left( \hat{D}^{-1} \hat{A}X\Theta + X\Theta' \right)
\]

- \( \hat{A} = A + I_N \) is the adjacency matrix with self-loops
- \( \hat{D} \) is the degree matrix: \( \hat{D}_{ii} = \sum_j \hat{A}_{ij} \)
Pooling layer

- Use Graph U-Net architecture to reduce the number of nodes $N$ via pooling ratio $k \in (0, 1]$

- Layer outputs $\lceil kN \rceil$ nodes

- Learn projection score $\tilde{p}$ used as gating values → ensure lower scoring nodes retain comparatively less features
Pooling layer—cont’d

▶ Obtain pooled graph \((X', A')\) as:

\[
\tilde{y} = \frac{X\tilde{\rho}}{\|\tilde{\rho}\|},
\]

\[
\tilde{i} = \text{top-}k(\tilde{y}, k),
\]

\[
X' = (X \cdot \tanh(\tilde{y}))_{\tilde{i}},
\]

\[
A' = A_{\tilde{i}, \tilde{i}}
\]

▶ This only requires a pointwise projection operation and slicing into the original matrices!
Readout layer

- Want “flattening” operation analogous to image CNNs
- Take the average and max of all learnt node embeddings for output graph \((X^k, A^k)\) of the \(k\)-th block:

\[
\bar{s}^k = \frac{1}{N^k} \sum_{i=1}^{N^k} \bar{x}^k_i \parallel \max_{i=1}^{N^k} \bar{x}^k_i 
\]

- Inspired by JK-net, summarise the graph by summing:

\[
\tilde{s} = \sum_{k=1}^{K} \bar{s}^k 
\]

- An MLP predicts the class for \(\tilde{s}\) at the “tail” of the model
Experiments

- Benchmark datasets: Enzymes, Proteins, DD, Collab
- 10-fold cross-validation, compare to Ying et al.(2018)

- Three \{GCN layer, pooling layer\} blocks
- Preserve $k = 80\%$ of the nodes on each pool
## Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Enzymes</th>
<th>D&amp;D</th>
<th>Collab</th>
<th>Proteins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphlet</td>
<td>41.03</td>
<td>74.85</td>
<td>64.66</td>
<td>72.91</td>
</tr>
<tr>
<td>Shortest-path</td>
<td>42.32</td>
<td>78.86</td>
<td>59.10</td>
<td>76.43</td>
</tr>
<tr>
<td>1-WL</td>
<td>53.43</td>
<td>74.02</td>
<td>78.61</td>
<td>73.76</td>
</tr>
<tr>
<td>WL-QA</td>
<td>60.13</td>
<td>79.04</td>
<td>80.74</td>
<td>75.26</td>
</tr>
<tr>
<td>PatchySAN</td>
<td>–</td>
<td>76.27</td>
<td>72.60</td>
<td>75.00</td>
</tr>
<tr>
<td>GraphSAGE</td>
<td>54.25</td>
<td>75.42</td>
<td>68.25</td>
<td>70.48</td>
</tr>
<tr>
<td>ECC</td>
<td>53.50</td>
<td>74.10</td>
<td>67.79</td>
<td>72.65</td>
</tr>
<tr>
<td>Set2Set</td>
<td>60.15</td>
<td>78.12</td>
<td>71.75</td>
<td>74.29</td>
</tr>
<tr>
<td>SortPool</td>
<td>57.12</td>
<td>79.37</td>
<td>73.76</td>
<td>75.54</td>
</tr>
<tr>
<td>DiffPool-Det</td>
<td>58.33</td>
<td>75.47</td>
<td><strong>82.13</strong></td>
<td>75.62</td>
</tr>
<tr>
<td>DiffPool-NoLP</td>
<td>62.67</td>
<td>79.98</td>
<td>75.63</td>
<td>77.42</td>
</tr>
<tr>
<td>DiffPool</td>
<td><strong>64.23</strong></td>
<td><strong>81.15</strong></td>
<td>75.50</td>
<td><strong>78.10</strong></td>
</tr>
<tr>
<td>Ours</td>
<td>64.17</td>
<td>78.59</td>
<td>74.54</td>
<td>75.46</td>
</tr>
</tbody>
</table>
Memory usage

Figure 2: GPU memory usage of our method (with no pooling; \(k = 1.0\)) and DiffPool (\(k = 0.25\)) during training on Erdős-Rényi graphs [9] of varying node sizes (and \(|E| = 2|V|\)). Both methods ran with 128 input and hidden features, and three Conv-Pool layers. “OOM” denotes out-of-memory.
Qualitative analysis
Future directions

- Apply the computation model to large datasets (particularly relevant for the neuroscience domain, where brain meshes are encoded by graphs with +100K nodes).

- Improve the pooling mechanism (e.g. by making the pooling ratio $k$ learnable).

- Make the “mirroring” operation (unpooling) independent of the pooling results.
Thank you!

Questions?

Catalina.Cangea@cst.cam.ac.uk
www.cst.cam.ac.uk/~ccc53/