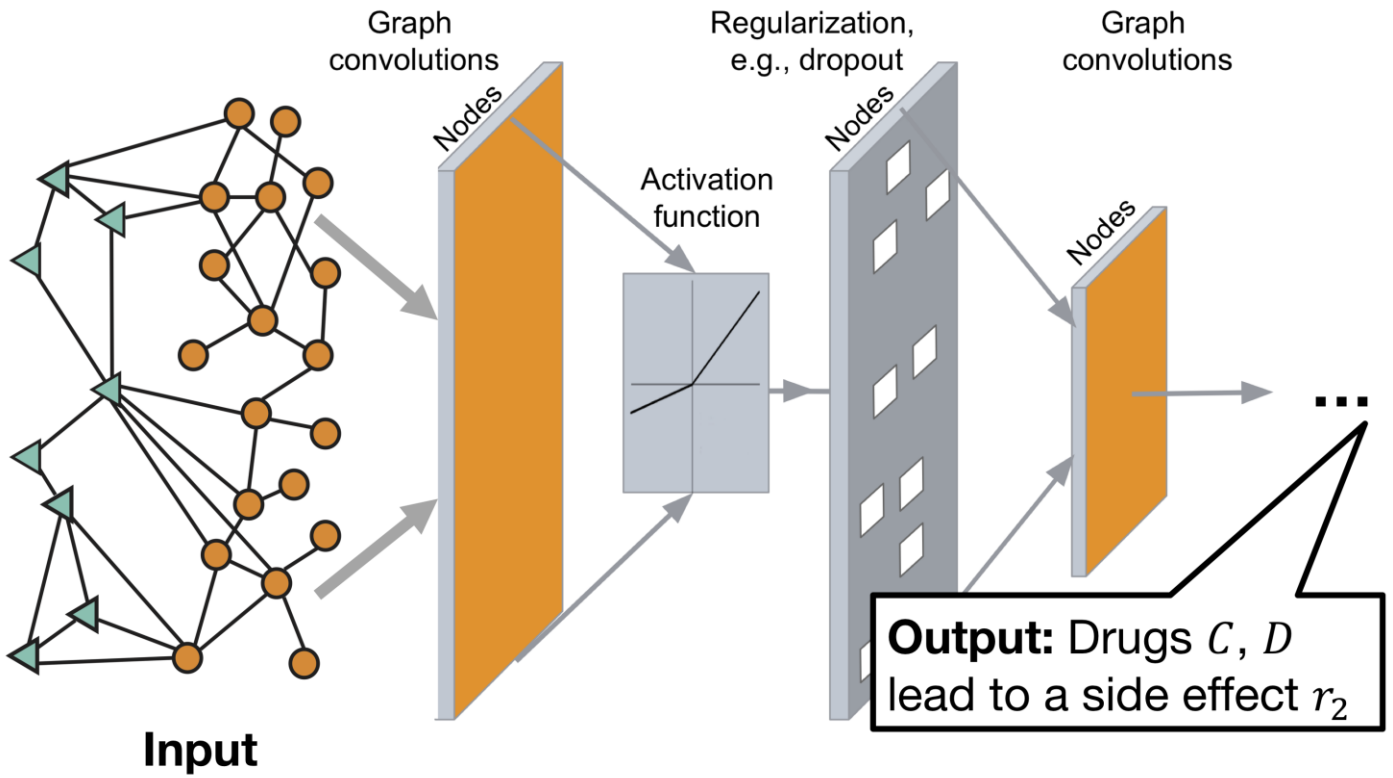
Spatial graph convolutions

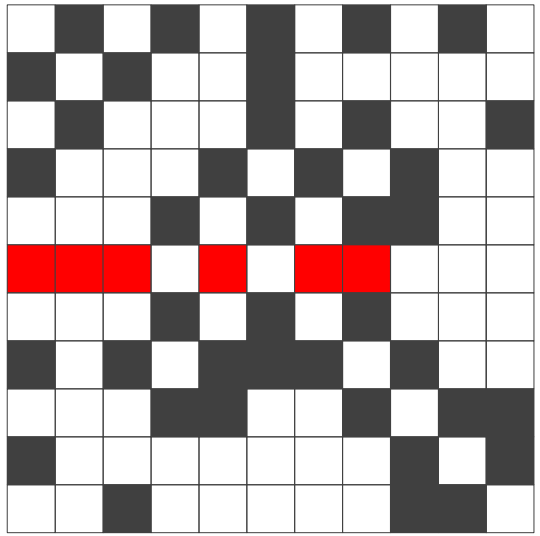


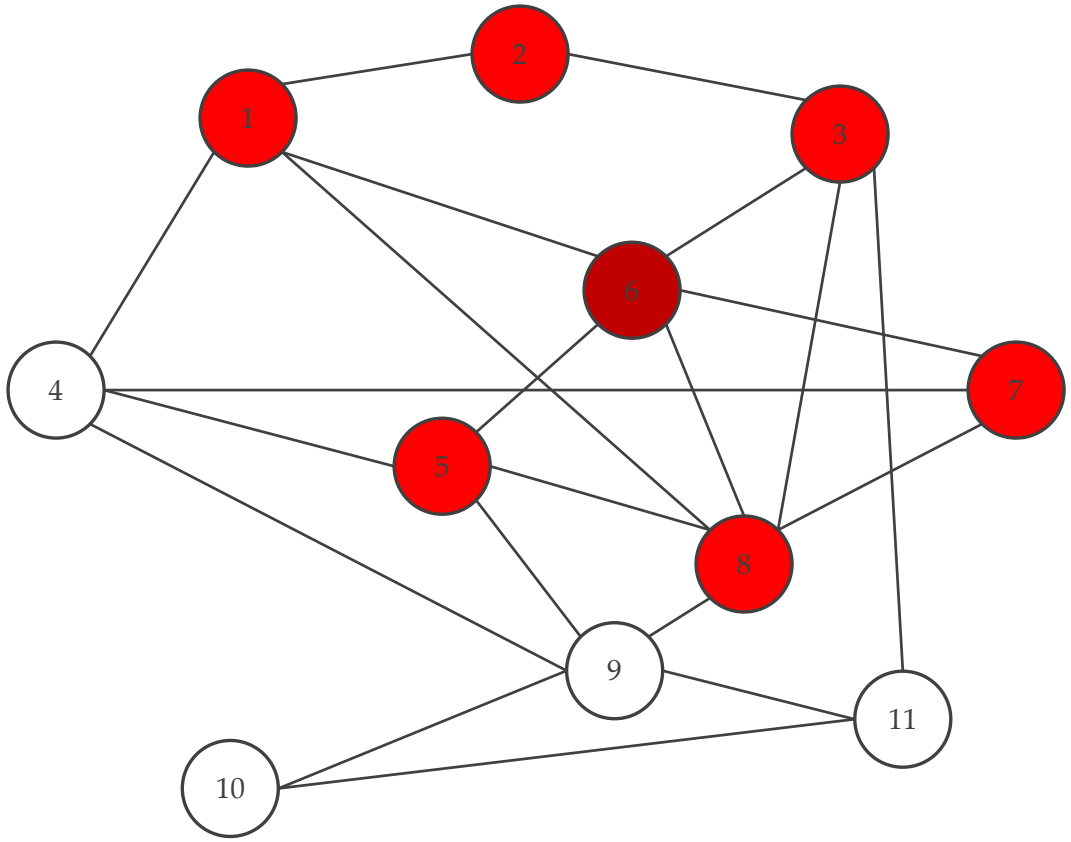
Graph convolutions on spatial domain

- Convolutions as (“local”) matrix multiplications

Adjacency matrix

Neighbors of “6”

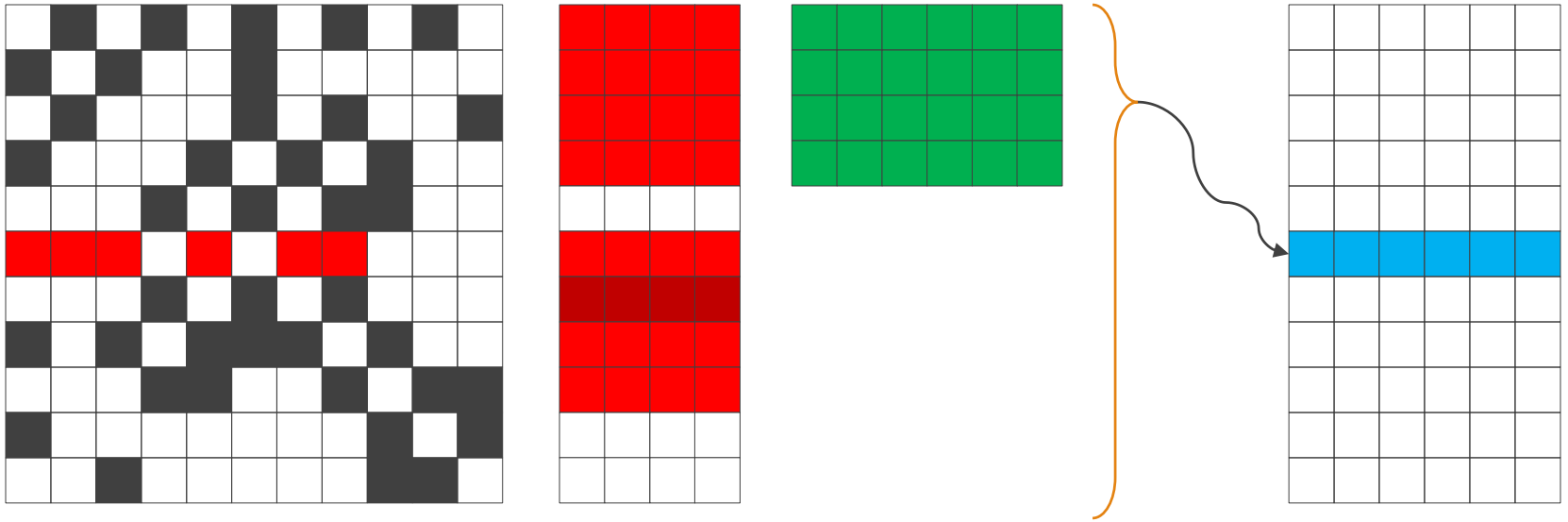


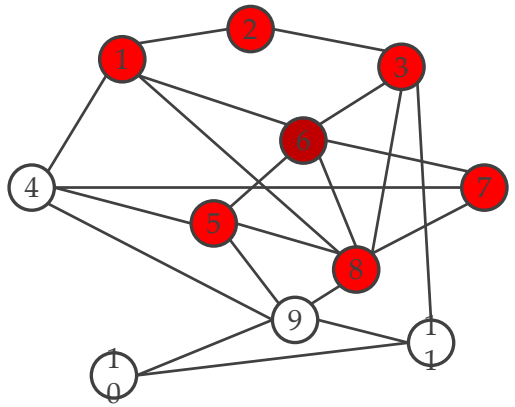


Graph Convolutional Networks (GCN)

- Each node has a feature vector (row-wise)
  - Left-multiplying with adjacency, we recover the features in neighborhood
  - Right-multiplying with a weight matrix, we "convolve" on neighborhood

$$\boldsymbol{y} = \mathrm{ReLU}(\boldsymbol{AXW})$$



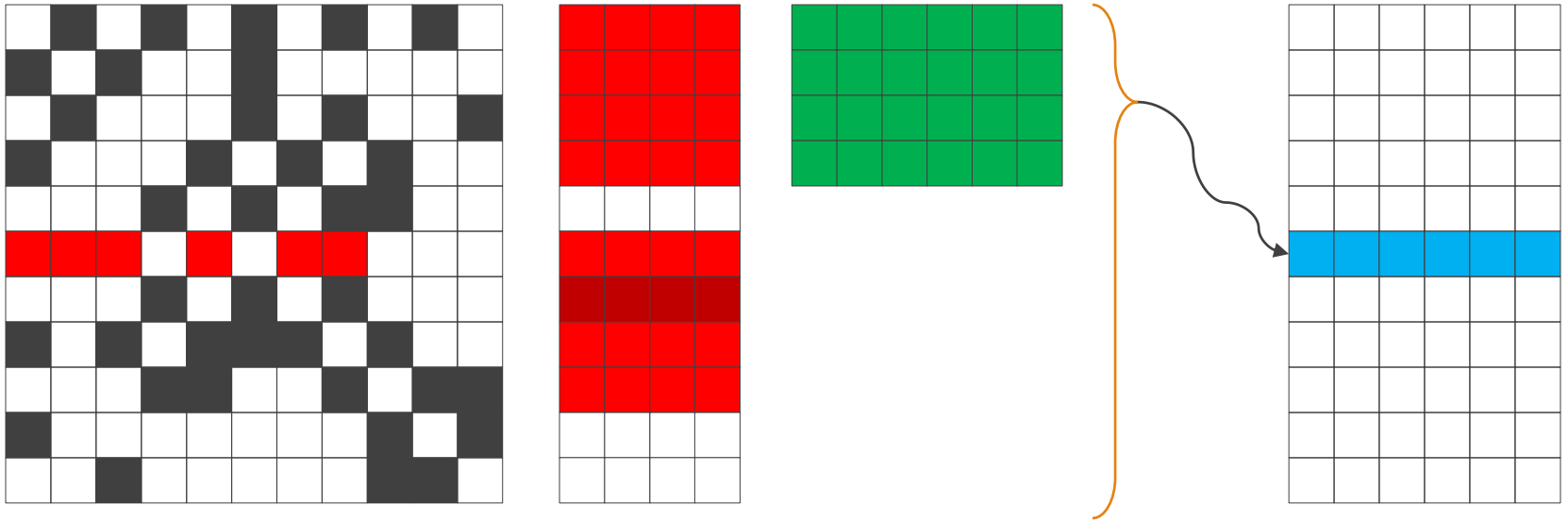


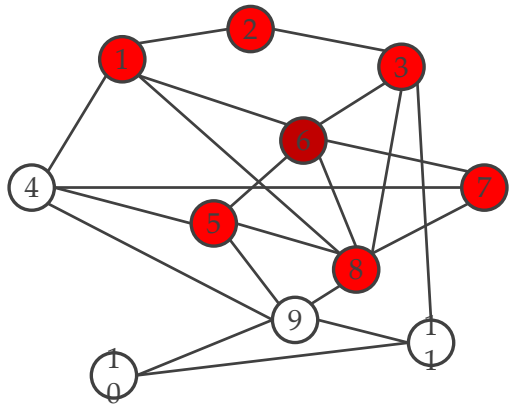
T. Kipf, M. Welling, 2016

Graph Convolutional Networks (GCN)

- Each node has a feature vector (row-wise)
  - Left-multiplying with adjacency, we recover the features in neighborhood
  - Right-multiplying with a weight matrix, we "convolve" on neighborhood
  - We can also stack multiple convolution layers

$$\boldsymbol{y} = \text{softmax}(\boldsymbol{A}\ \text{ReLU}(\boldsymbol{AXW}_1)\boldsymbol{W}_2)$$





T. Kipf, M. Welling, 2016

Graph Attention Networks (GAT)

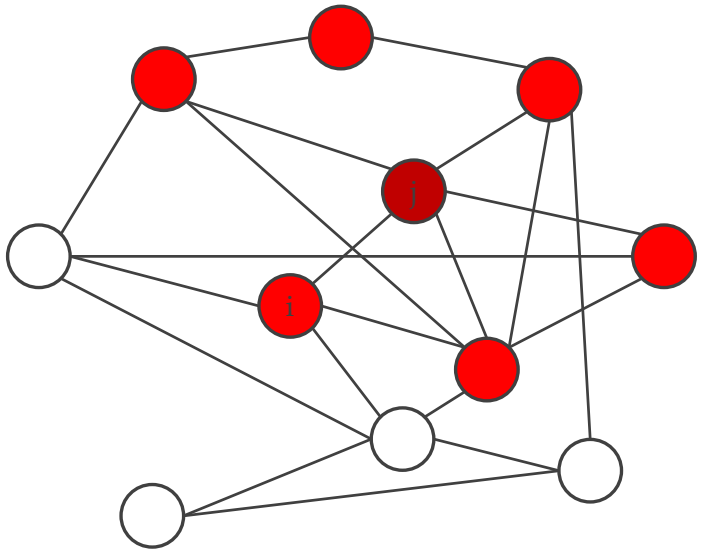
- Similar but including attention: $y_j = h(\sum_{j \in \mathcal{N}(i)} a_{ij} \boldsymbol{x_j})$
- Attention with using self-attention and graph convolutions

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k \in \mathcal{N}(i)} \exp(\text{LeakyRELU}\left([\boldsymbol{x}_i \boldsymbol{W}, \boldsymbol{x}_j \boldsymbol{W}] \cdot u\right))}$$

- e_{ij} are the self-attention weights

$$e_{ij} = \text{LeakyRELU}\left([\boldsymbol{x}_i \boldsymbol{W}, \boldsymbol{x}_j \boldsymbol{W}] \cdot u\right)$$

- u is a weight vector



F. Monti et al., 2017; P. Velickovic et al. 2018

Self-attention for graph convolutions

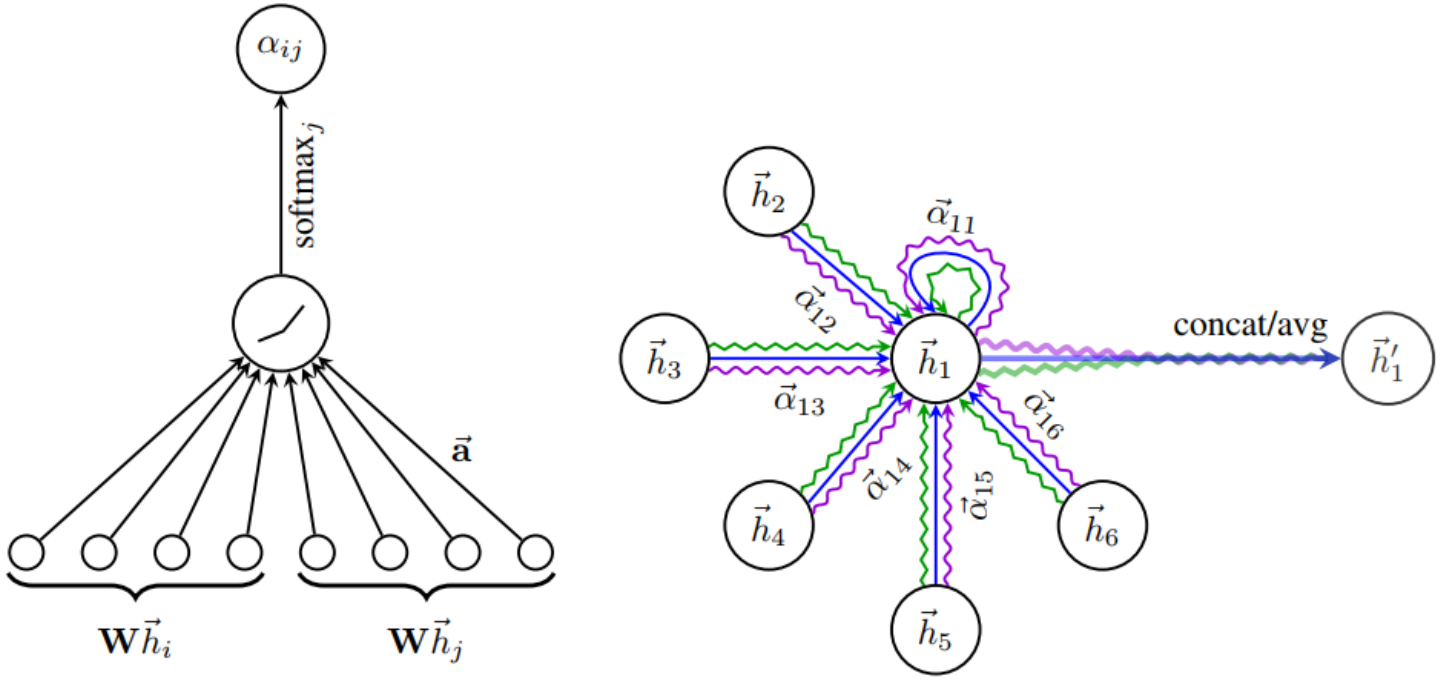
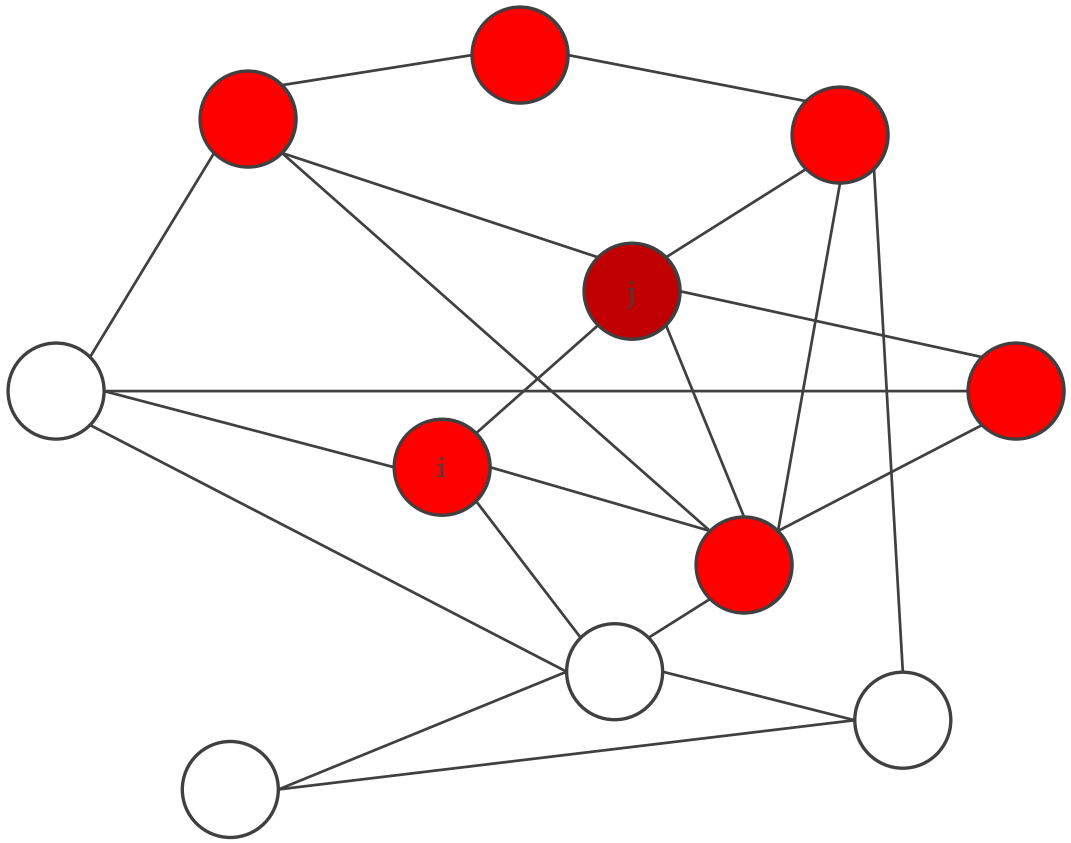


Figure 1: **Left:** The attention mechanism $a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$ employed by our model, parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$, applying a LeakyReLU activation. **Right:** An illustration of multi-head attention (with $K = 3$ heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or averaged to obtain $\vec{h}'_1$.

P. Velickovic et al. 2018

Message Passing Neural Network (MPNN)

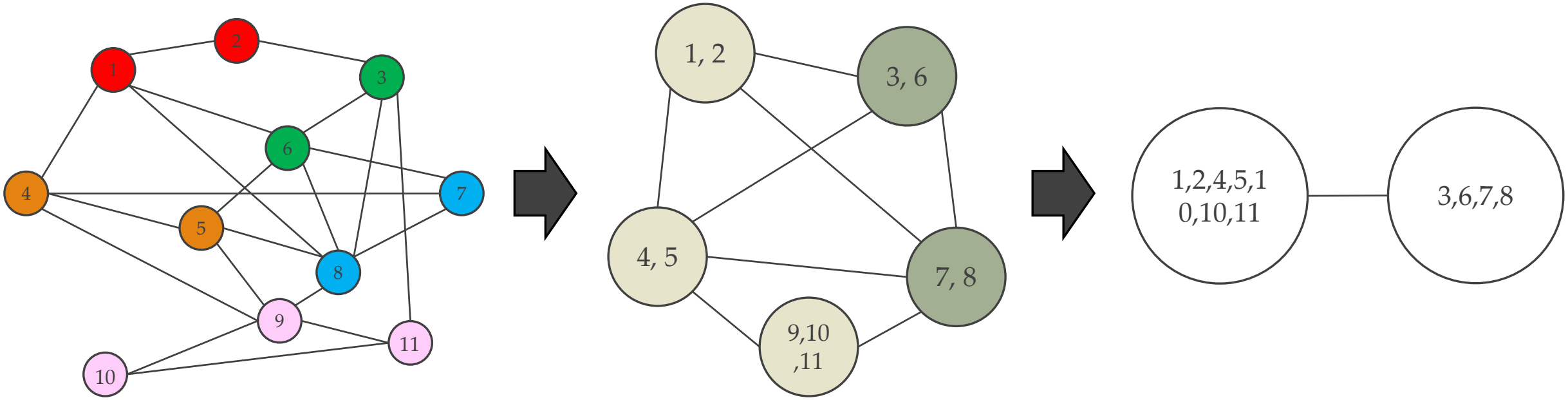
- General aggregation function
  - $y_j = g(\sum_{j \in \mathcal{N}(i)} h(x_i, x_j, e_{ij}, W))$



Gilmer et al. 2017

Coarsening graphs

- Group nodes together
- Learnable pooling
- Adjacency matrices get updated
- Average or max pool the node features



Differential Graph Pooling

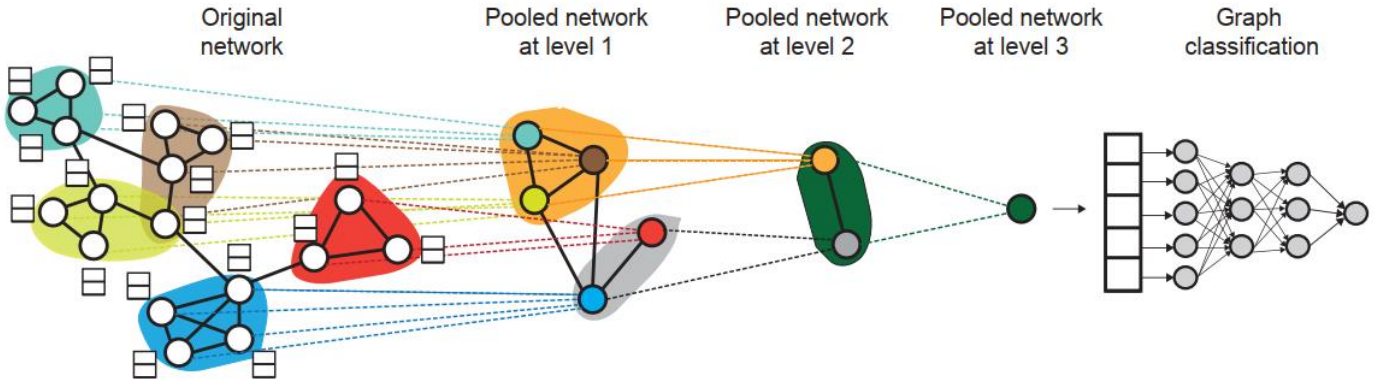


Figure 1: High-level illustration of our proposed method DIFFPOOL. At each hierarchical layer, we run a GNN model to obtain embeddings of nodes. We then use these learned embeddings to cluster nodes together and run another GNN layer on this coarsened graph. This whole process is repeated for L layers and we use the final output representation to classify the graph.

Summary

- What makes graphs special?
- Revisiting graphs
- Revisiting convolutions
- Spectral graph convolutions
- Spatial graph convolutions

Extra reading material:

- All papers mentioned in the slides