

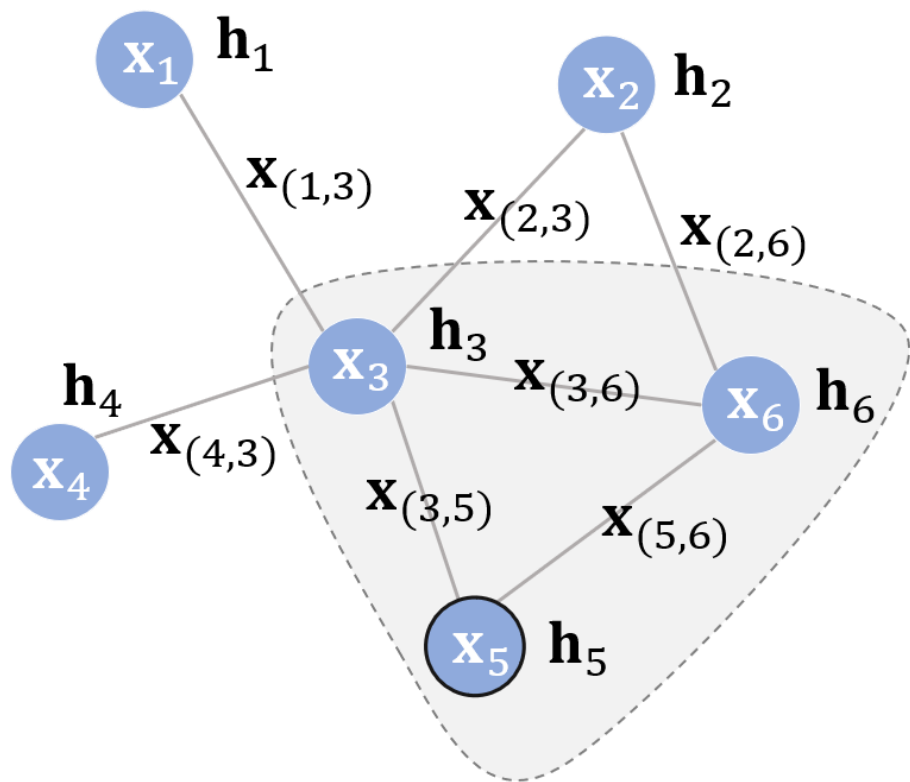
Graph Neural Networks

程豪

Contents

- Vanilla Graph Neural Networks
 - Limitations
- Graph Convolutional Networks
 - Spectral-based GCN
 - Limitations
 - Spatial-based GCN
 - MPNN
 - PATCHY-SAN
- Graph Attention Networks
- Pooling & Readout
 - Differentiable Pooling
- Graph Embedding and Graph Neural Network
- Reference

Vanilla GNN



$$\mathbf{h}_5 = f(\mathbf{x}_5, \mathbf{x}_{(3,5)}, \mathbf{x}_{(5,6)}, \mathbf{h}_3, \mathbf{h}_6, \mathbf{x}_3, \mathbf{x}_6)$$

$$\mathbf{h}_v^{(t)} = \sum_{u \in N(v)} f(\mathbf{x}_v, \mathbf{x}_{(v,u)}^e, \mathbf{x}_u, \mathbf{h}_u^{(t-1)}),$$

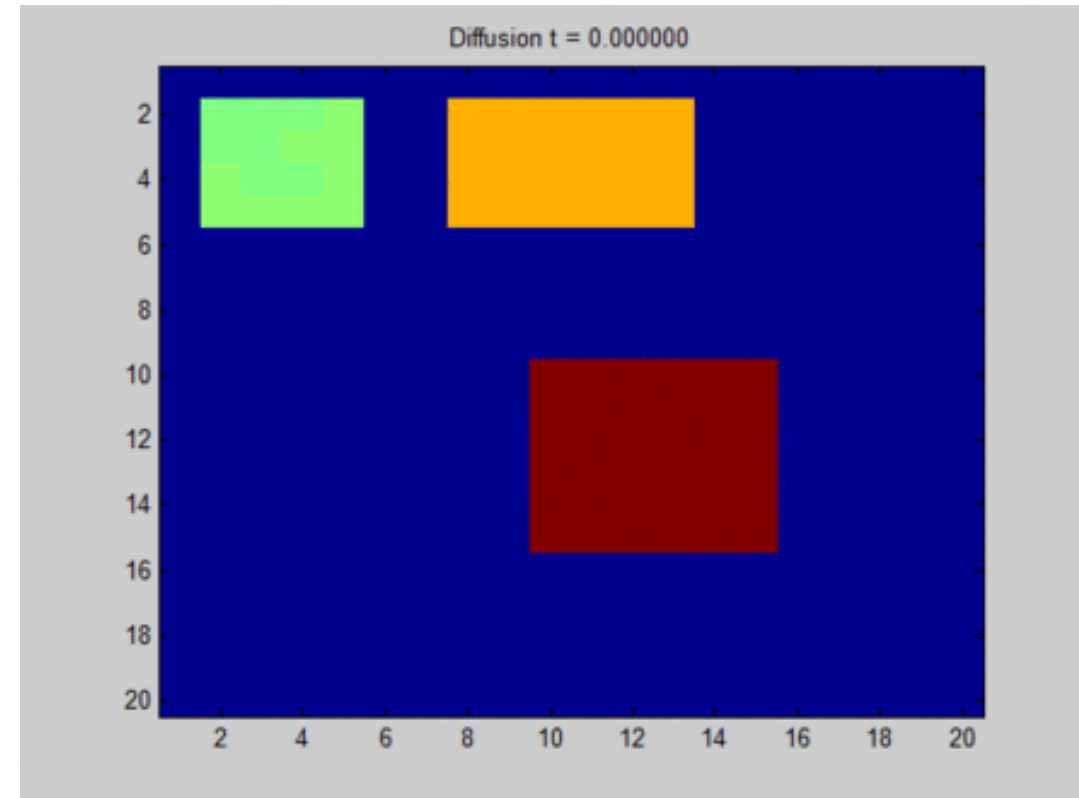
$$\mathbf{o}_v = g(\mathbf{h}_v, \mathbf{x}_v),$$

$$loss = \sum_{i=1}^p (\mathbf{t}_i - \mathbf{o}_i),$$

Limitations

1. Computationally inefficient
2. Features on the edges
3. Same parameters in the iteration
4. Over smooth

Improvement: GGNN, R-GCN



Graph Convolutional Networks


Spectral-based GCN

$$(f * g)(t) = F^{-1}[F[f(t)] \odot F[g(t)]]$$

$$\hat{f}(t) = \int f(x) \exp^{-2\pi i x t} dx$$

$$\Delta \exp^{-2\pi i x t} = \frac{\partial^2}{\partial t^2} \exp^{-2\pi i x t} = -4\pi^2 x^2 \exp^{-2\pi i x t}$$



Labeled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

$$D-A=L$$

$$L=U\Lambda U^T$$

$$U=(u_1,u_2,\cdots,u_n)$$

$$\Lambda=\begin{bmatrix}\lambda_1&\cdots&0\\ \cdots&\cdots&\cdots\\ 0&\cdots&\lambda_n\end{bmatrix}$$

$$\hat{f}(t)=\sum_{n=1}^Nf(n)u_t(n)$$

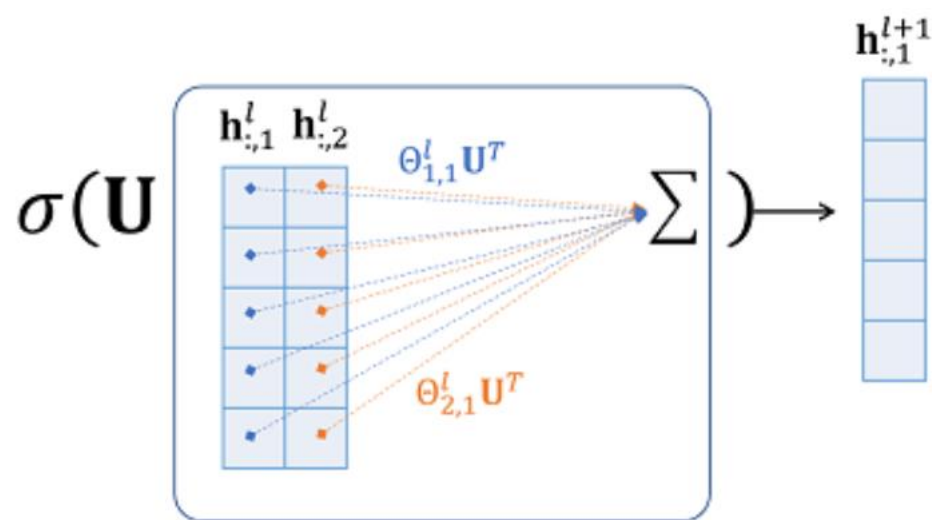
$$\hat{f}=\begin{bmatrix}\hat{f}(1)\\ \cdots\\ \hat{f}(N)\end{bmatrix}=U^Tf$$

$$(f\ast g)=F^{-1}[F[f]\odot F[g]]$$

$$(f\ast_Gg)=U(U^Tf\odot U^Tg)=U(U^Tg\odot U^Tf)$$

$$o=(f\ast_Gg)_\theta=Ug_\theta U^Tf$$

$$\mathbf{H}_{:,j}^{(k)} = \sigma\left(\sum_{i=1}^{f_{k-1}} \mathbf{U} \boldsymbol{\Theta}_{i,j}^{(k)} \mathbf{U}^T \mathbf{H}_{:,i}^{(k-1)}\right) \quad (j = 1, 2, \dots, f_k)$$



Limitations

- Computational complexity
 - Learned filters are domain dependent
 - Any perturbation to a graph results in a change of eigenbasis
-
- Improvement: ChebNet, GCN, AGCN

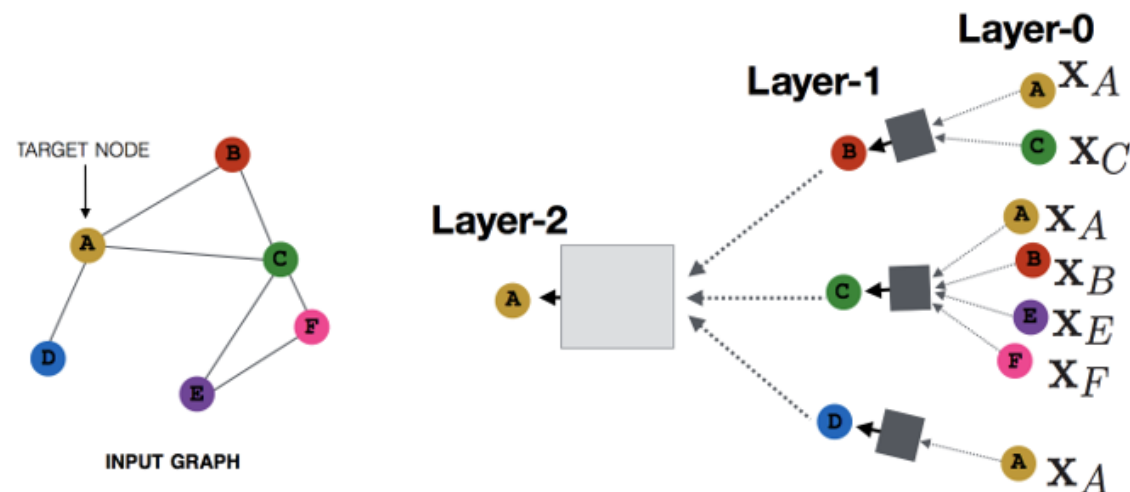
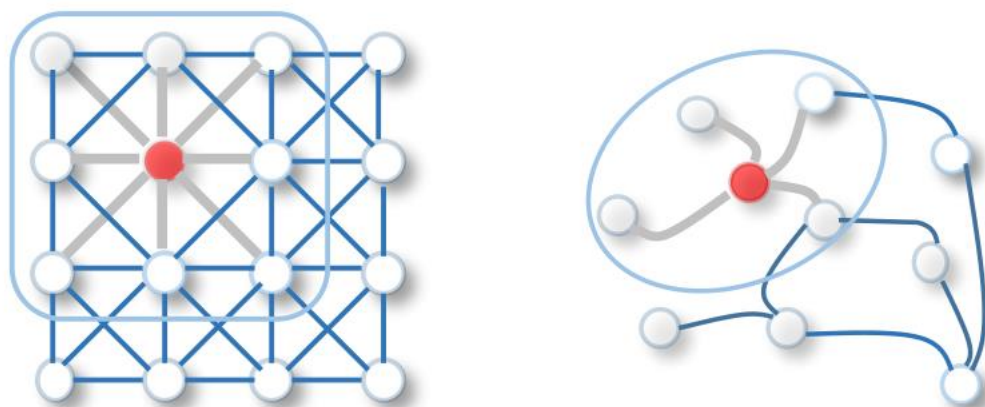
Graph Convolutional Networks

Spatial-based GCN

Message Passing Neural Network (MPNN)

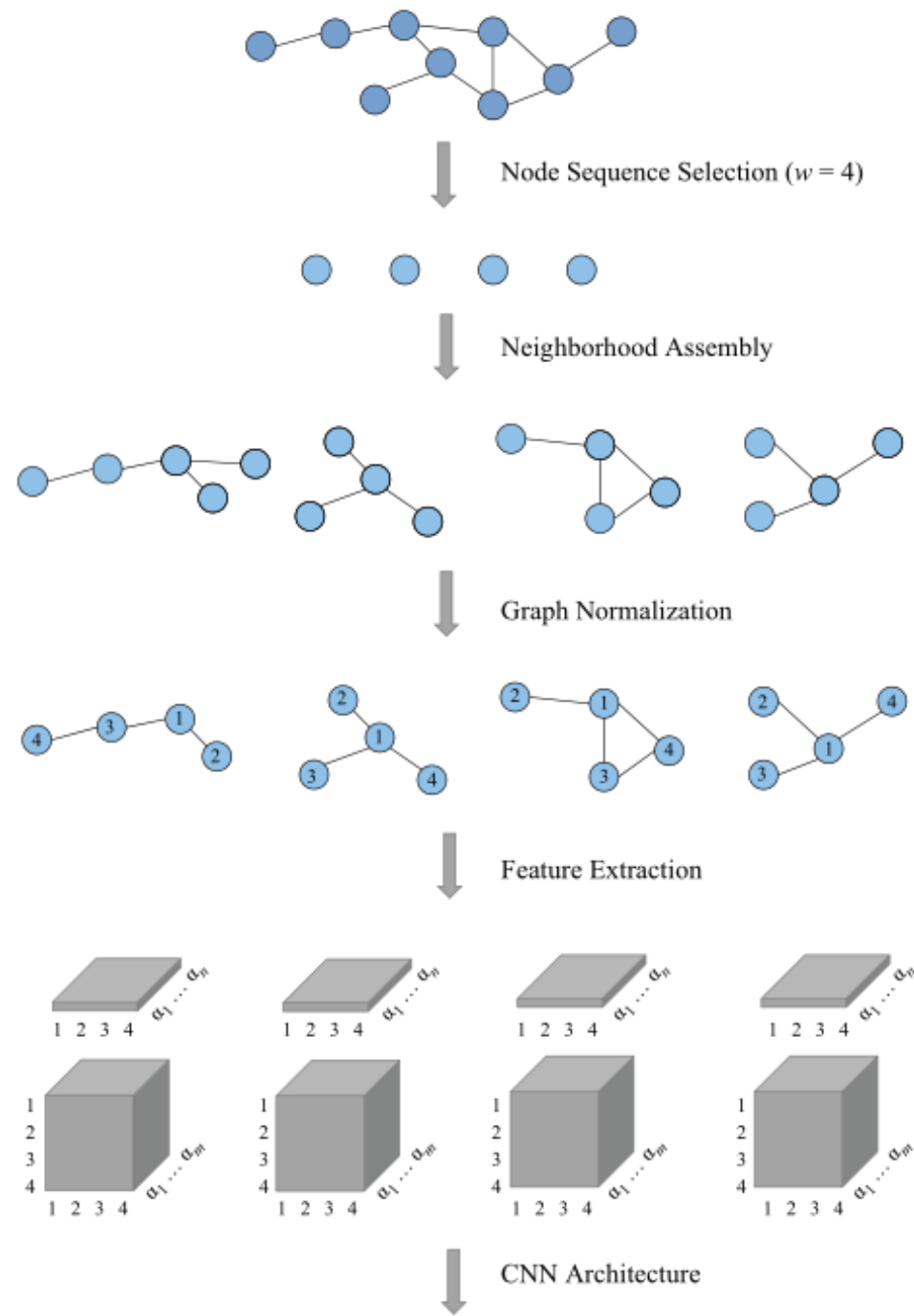
$$\mathbf{h}_v^{(k)} = U_k(\mathbf{h}_v^{(k-1)}, \sum_{u \in N(v)} M_k(\mathbf{h}_v^{(k-1)}, \mathbf{h}_u^{(k-1)}, \mathbf{x}_{vu}^e)),$$

$$\mathbf{h}_G = R(\mathbf{h}_v^{(K)} | v \in G),$$



PATCHY-SAN

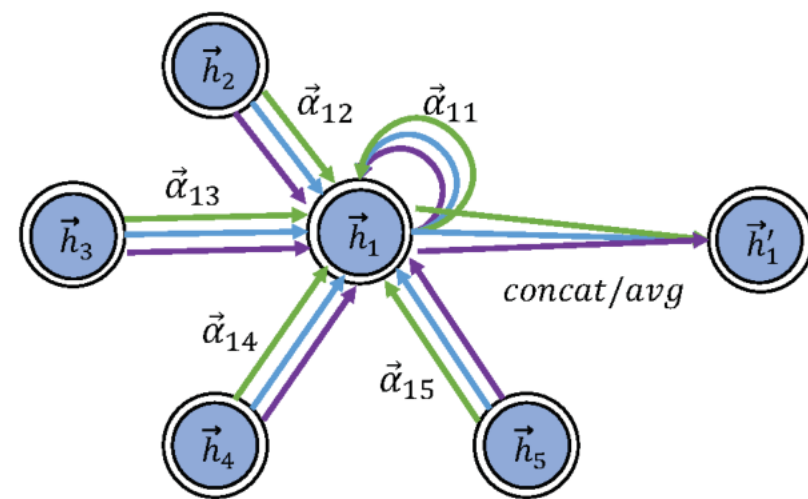
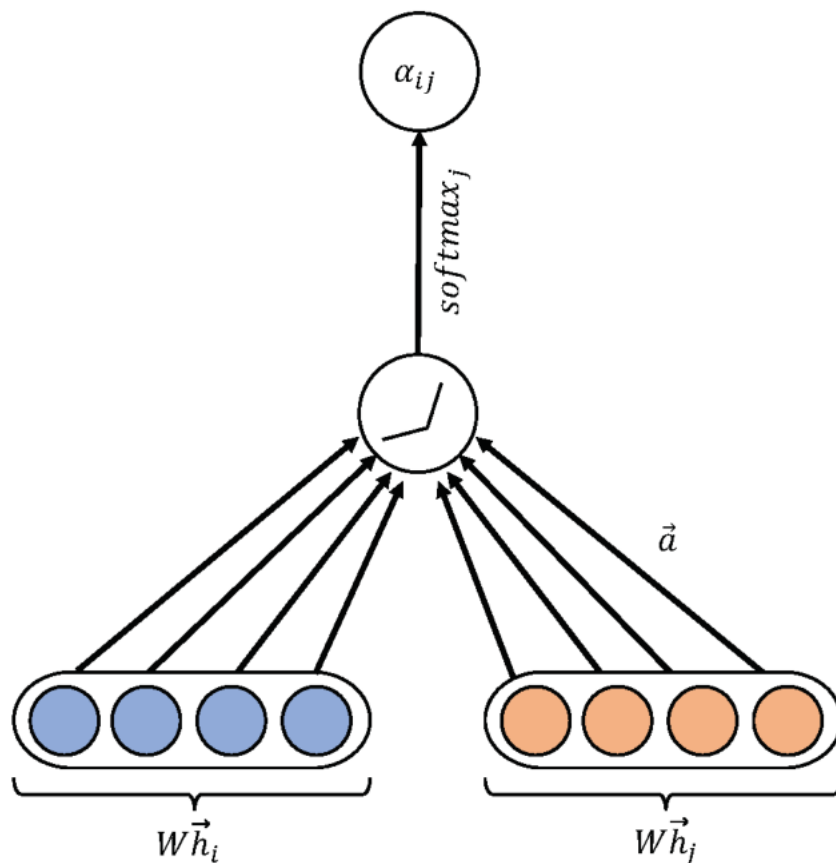
1. Node Sequence Selection
2. Neighborhood Assembly
3. Graph Normalization
4. Convolutional Architecture



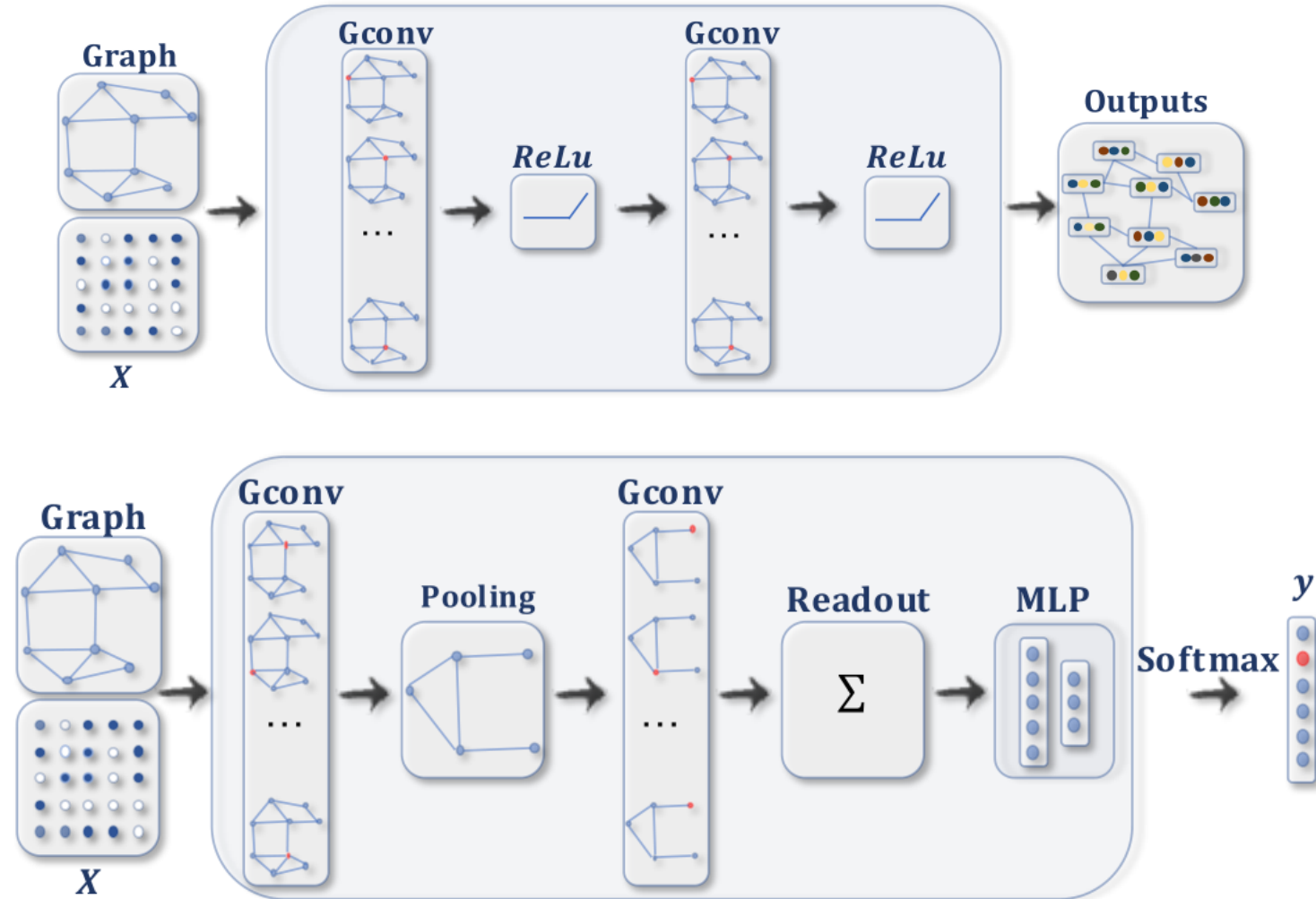
Graph Attention Networks

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^T [\mathbf{W}\mathbf{h}_i \parallel \mathbf{W}\mathbf{h}_j]))}{\sum_{k \in N_i} \exp(\text{LeakyReLU}(\mathbf{a}^T [\mathbf{W}\mathbf{h}_i \parallel \mathbf{W}\mathbf{h}_k]))},$$

$$\mathbf{h}'_i = \sigma\left(\sum_{j \in N_i} \alpha_{ij} \mathbf{W}\mathbf{h}_j\right).$$



Pooling & Readout



Differentiable Pooling

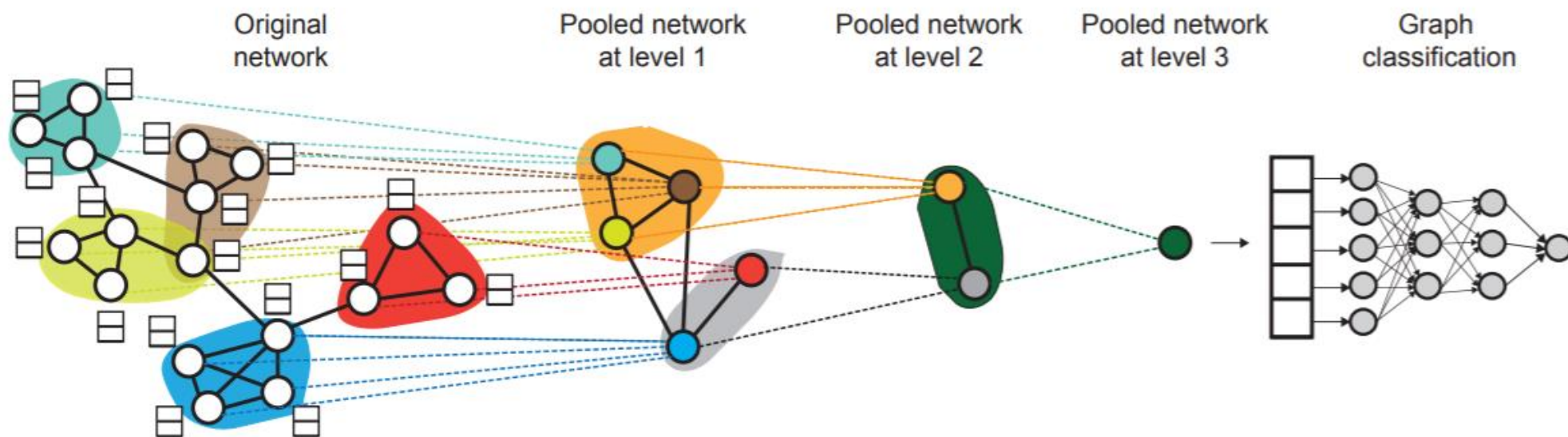


Figure 1: High-level illustration of our proposed method DIFFPOOL. At each hierarchical layer, we run a GNN model to obtain embeddings of nodes. We then use these learned embeddings to cluster nodes together and run another GNN layer on this coarsened graph. This whole process is repeated for L layers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

1. Soft Clustering(SC)
2. Node Representation(NR)

$$\mathbf{S}^l = SC(\mathbf{A}^l, \mathbf{H}^l), \text{ 其中 } \mathbf{S}^l \in \mathbb{R}^{(N_l \times N_{l+1})}$$

$$\tilde{\mathbf{H}}^l = NR(\mathbf{A}^l, \mathbf{H}^l)$$

$$\mathbf{H}^{l+1} = (\mathbf{S}^l)^T \tilde{\mathbf{H}}^l$$

$$\mathbf{A}^{l+1} = (\mathbf{S}^l)^T \mathbf{A}^l \mathbf{S}^l$$

\mathbf{A} : adjacency matrix

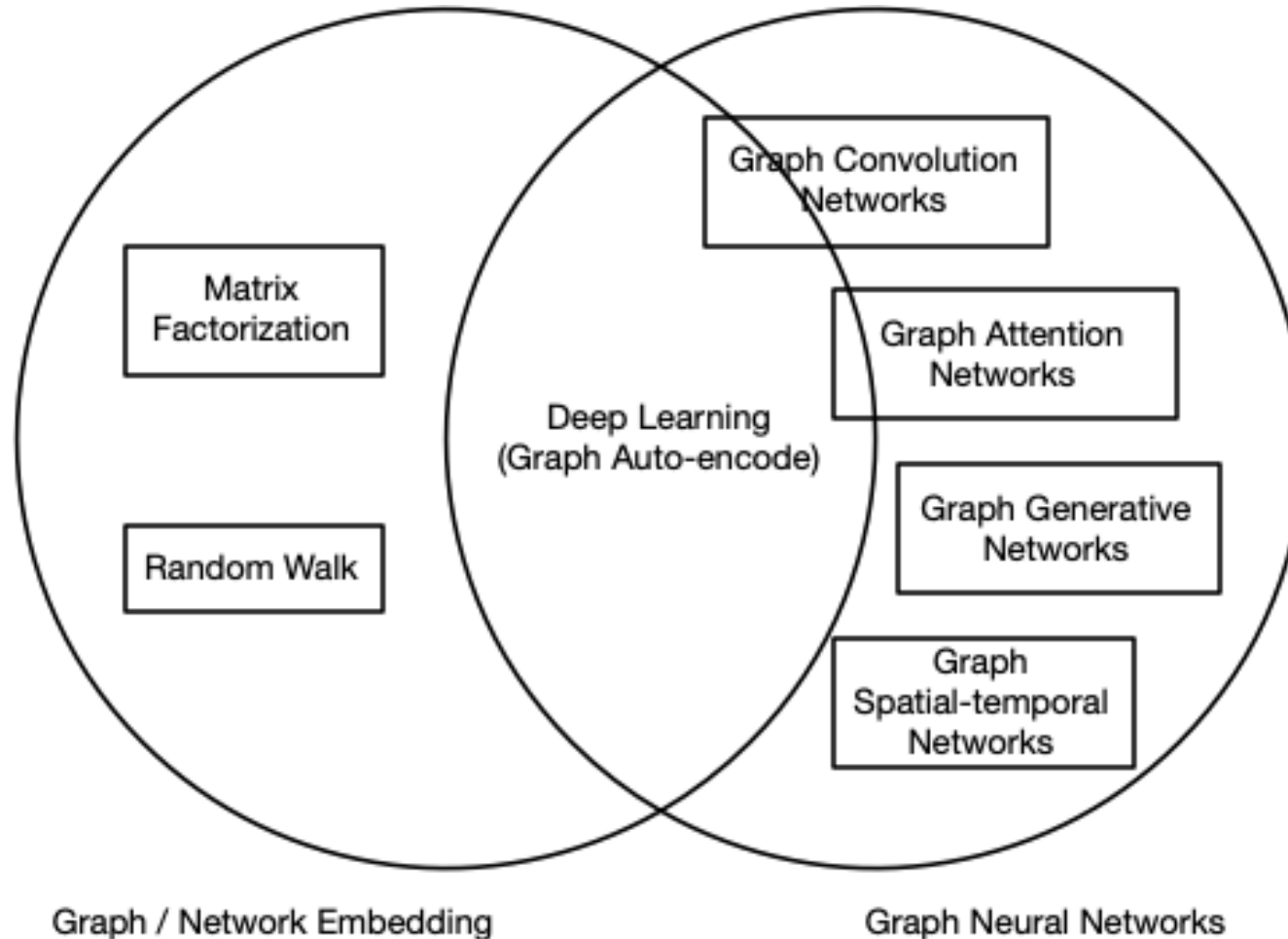
N_l : nodes of layer l

\mathbf{H} : embedding matrix

●	0	1	0
●	1	0	0
●	0	0	1
●	1	0	0
●	0	1	0
●	0	1	0

$$\mathbf{S}^l \in \mathbb{R}^{6 \times 3}$$

Graph Embedding and Graph Neural Network



Reference

- Introduction to Graph Neural Networks. Liu, Z., & Zhou, J. (2020).
- A Comprehensive Survey on Graph Neural Networks. Zonghan Wu. (2019).
- Deep Learning on Graphs: A Survey. Ziwei Zhang. (2015)
- Neural Message Passing for Quantum Chemistry. Justin Gilmer. (2017)
- Learning Convolutional Neural Networks for Graphs. Niepert et al. (2016)
- Hierarchical Graph Representation Learning with Differentiable Pooling. Rex Ying. (2019)
- [图嵌入 \(Graph Embedding\) 和图神经网络 \(Graph Neural Network\)](#)
- [从图\(Graph\)到图卷积\(Graph Convolution\): 漫谈图神经网络模型系列](#)
- [Paper笔记-- 《Learning Convolutional Neural Networks for Graphs》](#)

THANKS

程豪
2020/08/08