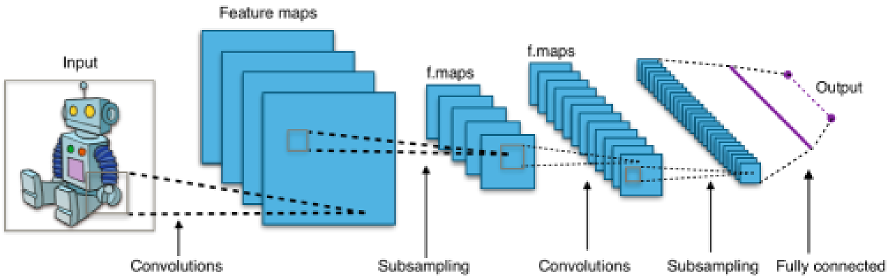
Graph Convolutional Neural Networks

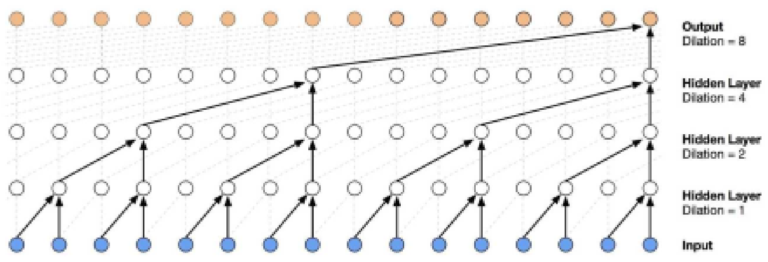
Huawei Shen
Institute of Computing Technology,
Chinese Academy of Sciences
2019.08.11

Convolutional Neural Network

- Convolutional neural network (CNN) gains great success on Euclidean data, e.g., image, text, audio, and video
  - Image classification, object detection, machine translation



Convolutional neural networks on image



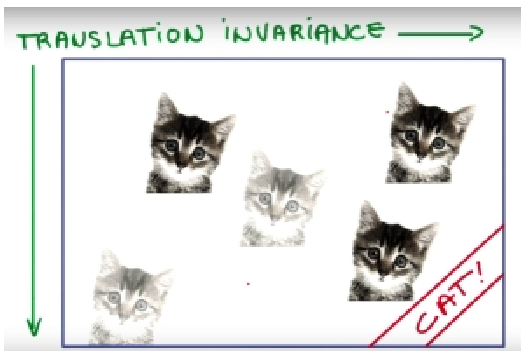
Temporal convolutional network

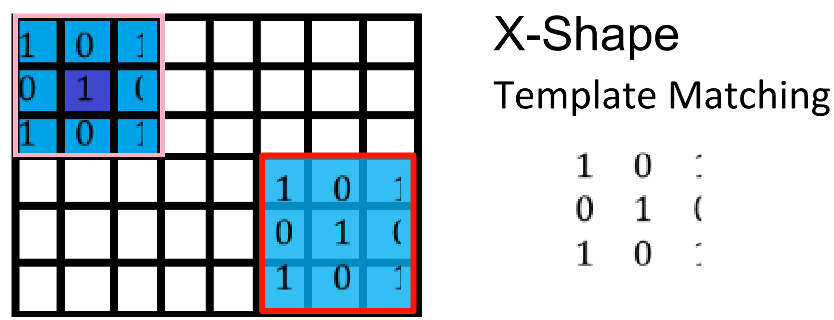
- The power of CNN lies in
  - its ability to learn **local stationary structures**, via **localized convolution filter,** and compose them to form **multi-scale hierarchical patterns**

M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, P. Vandergheynst. Geometric deep learning: going beyond Euclidean data. IEEE Signal Processing Magazine, 18-42, 2017.

Convolutional Neural Network

- **Localized convolutional filters are translation- or shift-invariant**
  - **Which are able to recognize identical features independently of their spatial locations**



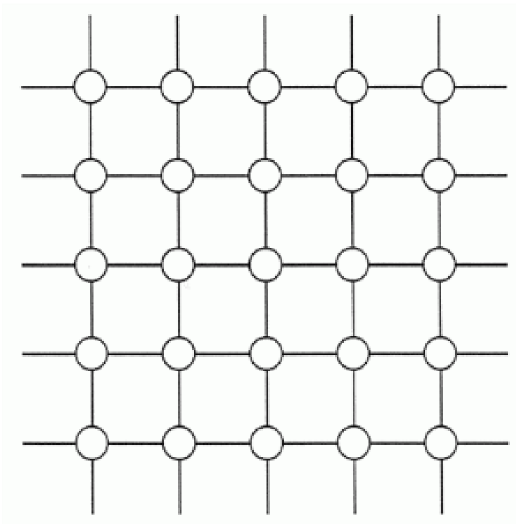


- **One interesting problem is how to generalize convolution to non-Euclidean domain, e.g., graph?**
  - **Irregular structure of graph poses challenges for defining convolution**

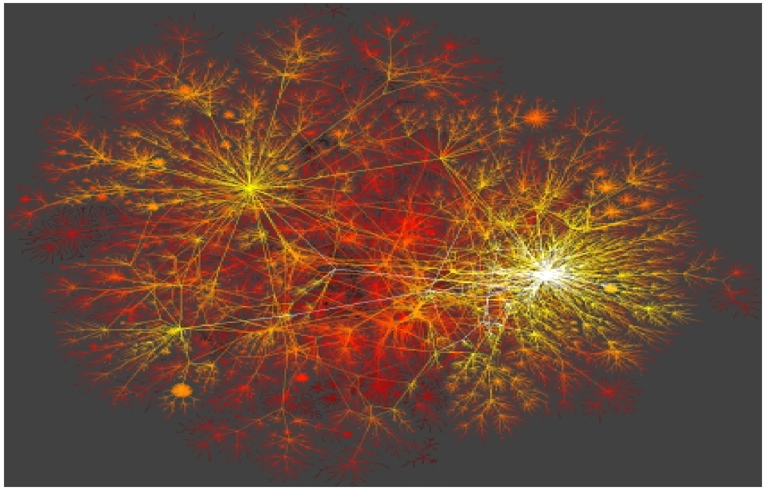
LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. Nature, 521(7553):436, 2015

From CNN to graph CNN

- Convolution is well defined in Euclidean data, grid-like network
- Not straightforward to define convolution on irregular network, widely observed in real world



Grid-like network



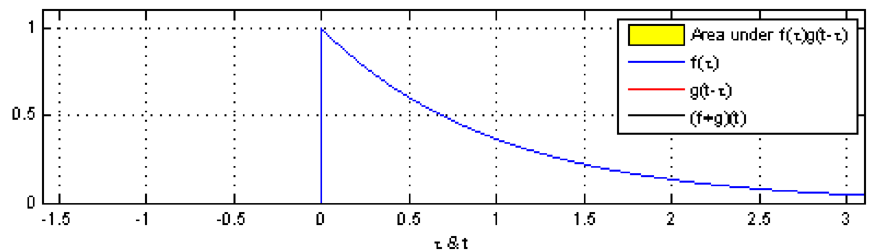
Irregular networks

Convolution

- **Convolution is a mathematical operation on two functions, f and g, to produce a third function h.**
  - **Defined as the integral, in continuous case, or sum, in discrete case, of**

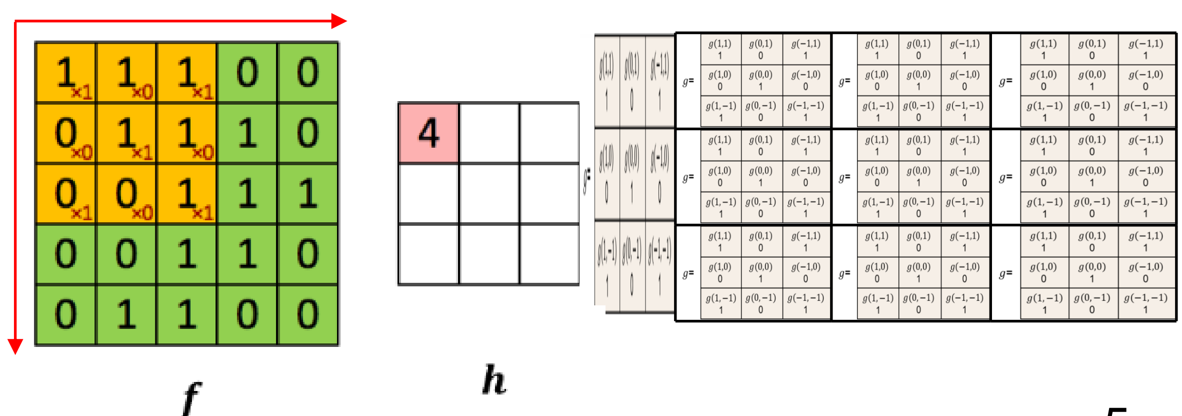
Continuous case

$$h(t) = (f * g)(t) \stackrel{\text{def}}{=} \int f(t)g(t-\tau)\,d\tau$$



Discrete case

$$h(x,y) = (f * g)(x,y) \stackrel{\text{def}}{=} \sum_{m,n} f(x-m,y-n)g(m,n)$$



Existing methods to define convolution

- **Spectral methods: define convolution in spectral domain**
  - **Convolution is defined via graph Fourier transform and convolution theorem.**
  - **The main challenge is that convolution filter defined in spectral domain is not localized in vertex domain.**

- **Spatial methods: define convolution in the vertex domain**
  - **Convolution is defined as a weighted average function over all vertices located in the neighborhood of target vertex.**
  - **The main challenge is that the size of neighborhood varies remarkably across nodes, e.g., power-law degree distribution.**

Spectral methods for graph convolutional neural networks

Spectral methods

- Given a graph $G = (V, E, W)$
  - V is node set with $n = |V|$, E is edge set, and $W \in R^{n \times n}$ is the weighted adjacency matrix
  - Each node is associated with d features, and $X \in R^{n \times d}$ is the feature matrix of nodes, each column of X is a signal defined over nodes

- Graph Laplacian
  - $L = D - W$, where is a diagonal matrix with $D_{ii} = \sum_j W_{ij}$
  - Normalized graph Laplacian

$$L = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$$

Graph Fourier Transform

- Fourier basis of graph G
  - The complete set of orthonormal eigenvectors $\{u_l\}_{l=0}^{n-1}$ of L, ordered by its non-negative eigenvalues $\{\lambda_l\}_{l=0}^{n-1}$
  - Graph Laplacian could be diagonalized as

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

where $U = [u_0, \cdots, u_{n-1}]$, and $\Lambda = \mathrm{diag}([u_0, \cdots, u_{n-1}])$

- Graph Fourier transform
  - Graph Fourier transform of a signal $x \in R^n$ is defined as

$$\hat{x} = U^T x$$

  - Graph Fourier inverse transform is

$$x = U \hat{x}$$

Define convolution in spectral domain

- Convolution theorem
  - The Fourier transform of a convolution of two signals is the point-wise product of their Fourier transforms

- According to convolution theorem, given a signal x as input and the other signal y as filter, graph convolution $*_G$ could be written as

$$x *_G y = U\left(\left(\boldsymbol{U}^T \boldsymbol{x}\right) \odot \left(\boldsymbol{U}^T y\right)\right)$$

Here, the convolution filter in spectral domain is $\boldsymbol{U}^T y$.

Define convolution in spectral domain

- **Graph convolution in spectral domain**
  - **Let $U^T y = [\theta_0, \cdots, \theta_{n-1}]^T$ and $g_\theta = \mathrm{diag}([\theta_0, \cdots, \theta_{n-1}])$, we have**

$$\boldsymbol{x} *_G \boldsymbol{y} = \boldsymbol{U}\left(\left(\boldsymbol{U}^T\boldsymbol{x}\right)\odot\left(\boldsymbol{U}^T\boldsymbol{y}\right)\right)$$

$$\boldsymbol{x} *_G \boldsymbol{y} = \boxed{\boldsymbol{U}}\,\boldsymbol{g}_\theta\,\boxed{\boldsymbol{U}^T\boldsymbol{x}}$$

Step 3
Graph Fourier
Inverse Transform

$\boldsymbol{U}\boldsymbol{g}_\theta\boldsymbol{U}^T\boldsymbol{x}$

Step 2:
Convolution
in spectral domain

$\boldsymbol{g}_\theta\boldsymbol{U}^T\boldsymbol{x}$

Step 1:
Graph
Fourier Transform

$\boldsymbol{U}^T\boldsymbol{x}$

Spectral Graph CNN

- **Spectral Graph CNN**

$$x_{k+1,j} = h\left(\sum_{i=1}^{f_k} U \boxed{F_{k,i,j}} U^T \boxed{x_{k,i}}\right)$$

$j = 1, \cdots, f_{k+1}$

Signals in the k-th layer

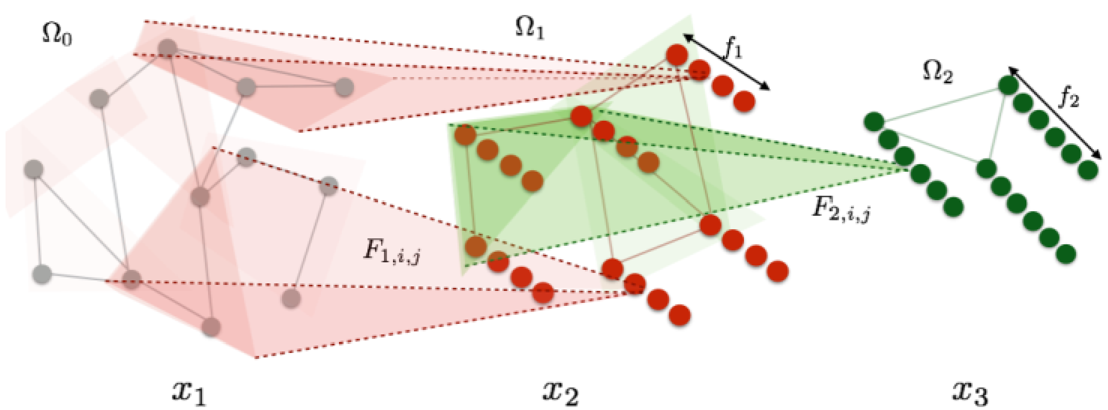
Filter in the k-th layer

Graph Fourier Transform

$$x^* = U^T x$$

Graph Fourier Inverse Transform

$$x = U x^*$$



J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. ICLR, 2014.

Shortcomings of Spectral graph CNN

- **Requiring eigen-decomposition of Laplacian matrix**
  - Eigenvectors are explicitly used in convolution
- **High Computational cost**
  - Multiplication with graph Fourier basis $\boldsymbol{U}$ is $\boldsymbol{O}(\boldsymbol{n}^2)$
- **Not localized in vertex domain**

ChebyNet: parameterizing filter

- **Parameterizing convolution filter via polynomial approximation**

$$\boldsymbol{g_\theta} = \mathbf{diag}([\boldsymbol{\theta_0}, \cdots, \boldsymbol{\theta_{n-1}}])$$

$$\Downarrow$$

$$g_\beta(\Lambda) = \sum_{k=0}^{K-1} \beta_k \Lambda^k \qquad \Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_n)$$

- **ChebyNet**

$$\boldsymbol{x} *_G \boldsymbol{y} = \boldsymbol{U} g_\beta(\Lambda) \boldsymbol{U}^T \boldsymbol{x} = \sum_{k=0}^{K-1} \beta_k L^k \boldsymbol{x}$$

The number of free parameters reduces from n to K

M. Defferrard, X. Bresson, P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. NeuralPS, 2016.

ChebyNet vs. Spectral Graph CNN

- **Eigen-decomposition is not required**

- **Computational cost is reduced from $O(n^2)$ to $O(|E|)$**

$$\boldsymbol{x} *_G \boldsymbol{y} = \boldsymbol{U} g_\beta(\Lambda) \boldsymbol{U}^T \boldsymbol{x} = \sum_{k=0}^{K-1} \beta_k L^k \boldsymbol{x}$$

- **Convolution is localized in vertex domain**
  - Convolution is strictly localized in a ball of radius K, i.e., K hops from the central vertex

Is this method good enough? What could we do more?

M. Defferrard, X. Bresson, P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. NeuralPS, 2016.

Our method:
Graph Wavelet Neural Network
(ICLR 2019)

Graph wavelet neural network

- ChebyNet achieves localized convolutional via restricting the space of graph filters as a polynomial function of eigenvalue matrix Λ

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k$$

- We focus on the Fourier basis to achieve localized graph convolution

$$x *_G y = \boxed{U} g_\theta \boxed{U^T} x$$

- We propose to replace Fourier basis with wavelet basis

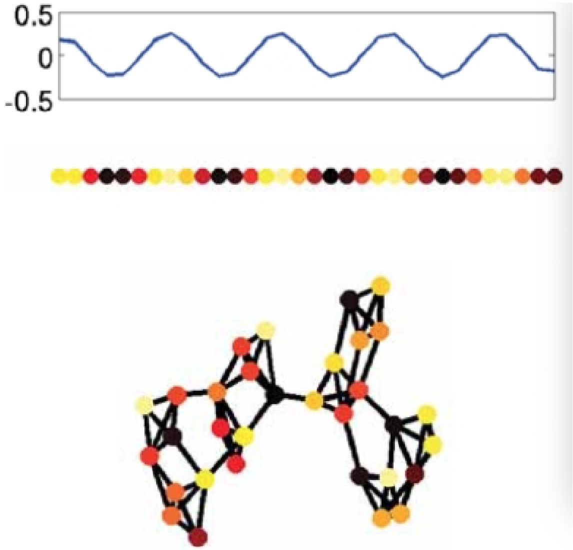
Fourier vs. Wavelet

Fourier Basis

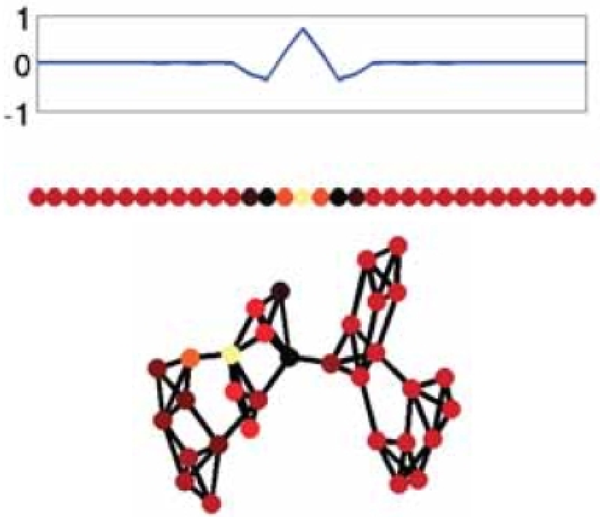
- Dense
- Not localized
- High Computational cost

Wavelet Basis

- Sparse
- Localized
- Low Computational cost



Fourier basis: U



Wavelet basis: $\psi_s = Ue^{\lambda s}U^T$

B. Xu, H. Shen, Q. Cao, Y. Qiu, X. Cheng. Graph wavelet neural network. ICLR 2019.

Graph wavelet neural network

- **Graph Wavelet Neural Network**
  - **Replace graph Fourier transform with graph wavelet transform**

Graph Fourier transform

$$x^* = U^T x$$

Inverse Fourier transform

$$x = U x^*$$

Graph Wavelet transform

$$x^* = {\psi_s}^{-1} x$$

Inverse Wavelet transform

$$x = \psi_s x^*$$

Graph wavelet neural network (GWNN)

- **Graph convolution via wavelet transform**

$$\boldsymbol{x} *_{\mathcal{G}} \boldsymbol{y} = \boldsymbol{U}\left((\boldsymbol{U}^{\top}\boldsymbol{y}) \odot (\boldsymbol{U}^{\top}\boldsymbol{x})\right),$$

Replacing basis

$$\boldsymbol{x} *_{\mathcal{G}} \boldsymbol{y} = \psi_s((\psi_s^{-1}\boldsymbol{y}) \odot (\psi_s^{-1}\boldsymbol{x}))$$

- **Graph wavelet neural network**

$$x_{k+1,j} = h\left(\sum_{i=1}^{p} U F_{k,i,j} U^T x_{k,i}\right) \Rightarrow x_{k+1,j} = h\left(\sum_{i=1}^{p} \psi_s F_{k,i,j} \psi_s^{-1} x_{k,i}\right)$$

$j = 1, \cdots, q$

Parameter complexity: $O(n * p * q)$

Reducing parameter complexity

- **Key idea:**
  - **Detaching graph convolution from feature transformation**

$$x_{k+1,j} = h\left(\sum_{i=1}^{p} \psi_s F_{k,i,j} \psi_s^{-1} x_{k,i}\right) \quad j = 1, \cdots, q$$

$\boldsymbol{T} \in \boldsymbol{R}^{q \times p}$ with $\boldsymbol{p} * \boldsymbol{q}$ parameters

F_k is a diagonal matrix with $\boldsymbol{n}$ parameters

$$\underbrace{y_{k,j} = \sum_{i=1}^{p} T_{ji} x_{k,i} \quad + \quad x_{k+1,j} = h\left(\psi_s F_k \psi_s^{-1} y_{k,j}\right)}$$

Feature transformation — Graph convolution

The number of parameters reduces from $O(n * p * q)$ to $O(n + p * q)$

GWNN vs. ChebyNet

- **Benchmark datasets**

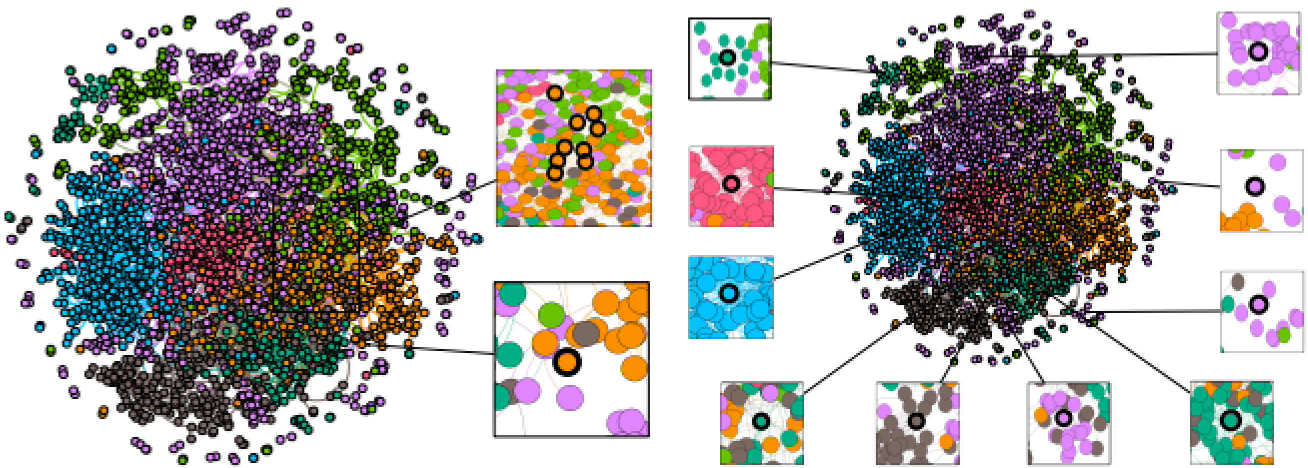
Dataset	Nodes	Edges	Classes	Features	Label Rate
Citeseer	3,327	4,732	6	3,703	0.036
Cora	2,708	5,429	7	1,433	0.052
Pubmed	19,717	44,338	3	500	0.003

- **Results at the task of node classification**

Method	Cora	Citeseer	Pubmed
MLP	55.1%	46.5%	71.4%
ManiReg	59.5%	60.1%	70.7%
SemiEmb	59.0%	59.6%	71.7%
LP	68.0%	45.3%	63.0%
DeepWalk	67.2%	43.2%	65.3%
ICA	75.1%	69.1%	73.9%
Planetoid	75.7%	64.7%	77.2%
Spectral CNN	73.3%	58.9%	73.9%
ChebyNet	81.2%	69.8%	74.4%
GWNN	**82.8%**	**71.7%**	**79.1%**

Graph wavelet neural network

- **Each Graph wavelet offers us a local view, i.e., from a center node, about the proximity for each pair of nodes**



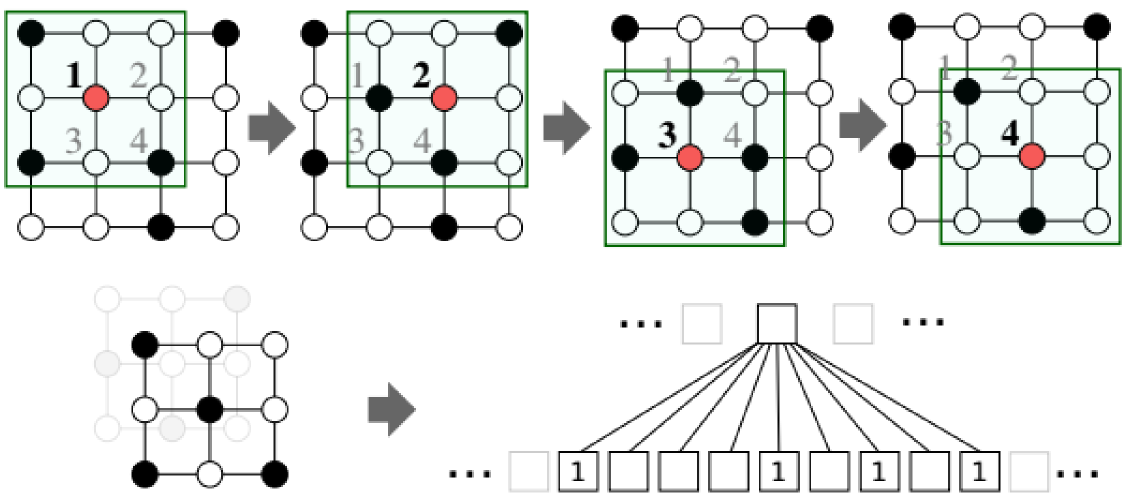
(a) (b)

Wavelet offers us a better basis for defining graph convolutional networks in spectral domain

Spatial methods for graph convolutional neural networks

Spatial Methods for Graph CNN

- **By analogy**
  - **What can we learn from the architecture of standard convolutional neural network?**



1. Determine Neighborhood

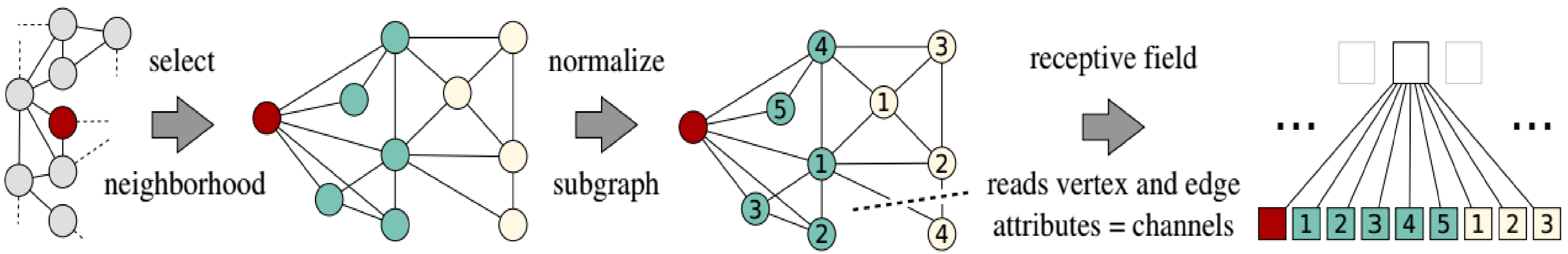
2. Impose an order in neighborhood

3. Parameter sharing

M. Niepert, M. Ahmed, K. Kutzkov. Learning Convolutional Neural Networks for Graphs. ICML, 2016.

Spatial Methods for Graph CNN

- **By analogy**
  - **For each node, select the fixed number of nodes as its neighboring nodes, according to certain proximity metric**
  - **Impose an order according to the proximity metric**
  - **Parameter sharing**



1. Determine Neighborhood

2. Impose an order in neighborhood

3. Parameter sharing

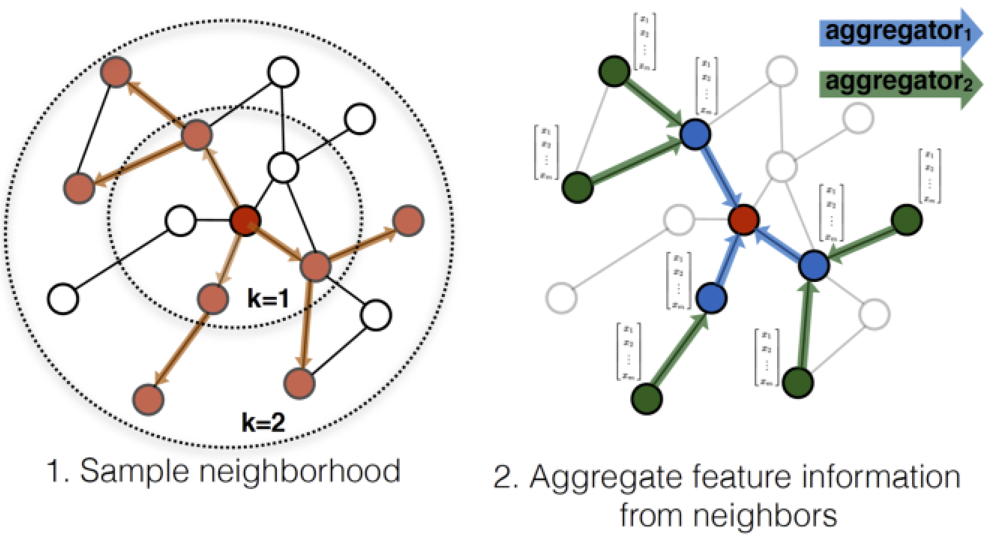
M. Niepert, M. Ahmed, K. Kutzkov. Learning Convolutional Neural Networks for Graphs. ICML, 2016.

Spatial Methods for Graph CNN

- **GraphSAGE**
  - **Sampling neighbors**
  - **Aggregating neighbors**

$$a_v^{(k)} = \text{AGGREGATE}^{(k)}\left(\left\{h_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right)$$

$$h_v^{(k)} = \text{COMBINE}^{(k)}\left(h_v^{(k-1)}, a_v^{(k)}\right)$$



GraphSAGE: Inductive Learning

General framework of graph neural networks: Aggregate the information of neighboring nodes to update the representation of center node

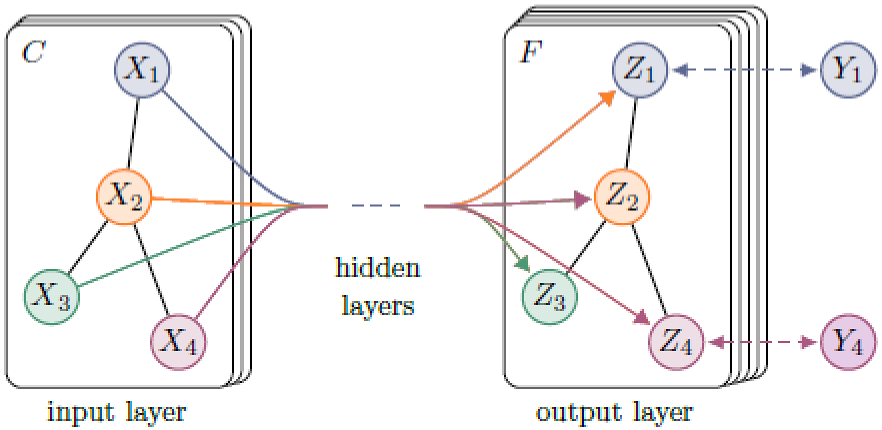
W. L. Hamilton, R. Ying, J. Leskovec. Inductive Representation Learning on Large Graphs. NeuralPS 2017

Spatial Methods for Graph CNN

- **GCN: Graph Convolution Network**
  - **Aggregating information from neighborhood via a normalized Laplacian matrix**
  - **Shared parameters are from feature transformation**
  - **A reduced version of ChebNet**

$$Z = f(X, A) = \mathrm{softmax}\left(\hat{A}\ \mathrm{ReLU}\left(\hat{A}X\boxed{W^{(0)}}\right)\boxed{W^{(1)}}\right)$$

Parameter for feature Transformation



T. N. Kipf, and M. Welling. Semi-supervised classification with graph convolutional networks. ICLR 2017

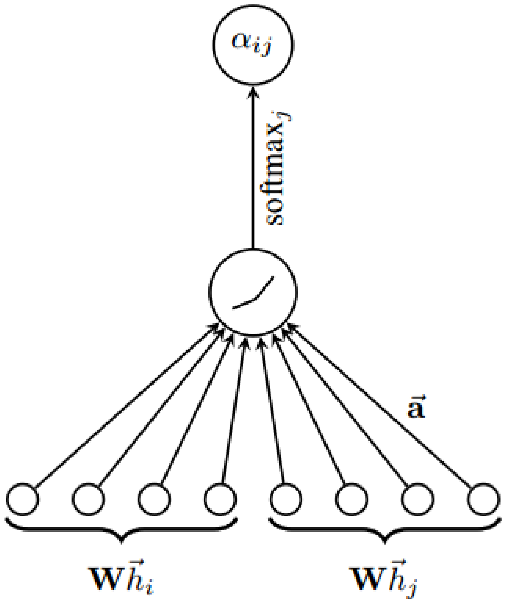
Spatial Methods for Graph CNN

- **GAT: Graph Attention Network**
  - **Learning the aggregation matrix, i.e., Laplacian matrix in GCN, via attention mechanism**
  - **Shared parameters contain two parts**
    - **Parameters for feature transformation**
    - **Parameters for attention**

Parameter for feature Transformation

$$\alpha_{ij} = \frac{\exp\left(\mathrm{LeakyReLU}\left(\vec{\mathbf{a}}^T[\boxed{\mathbf{W}}\vec{h}_i \| \mathbf{W}\vec{h}_j]\right)\right)}{\sum_{k\in\mathcal{N}_i}\exp\left(\mathrm{LeakyReLU}\left(\boxed{\vec{\mathbf{a}}^T}[\mathbf{W}\vec{h}_i \| \mathbf{W}\vec{h}_k]\right)\right)}$$

Parameter of Attention mechanism



Attention Mechanism in GAT

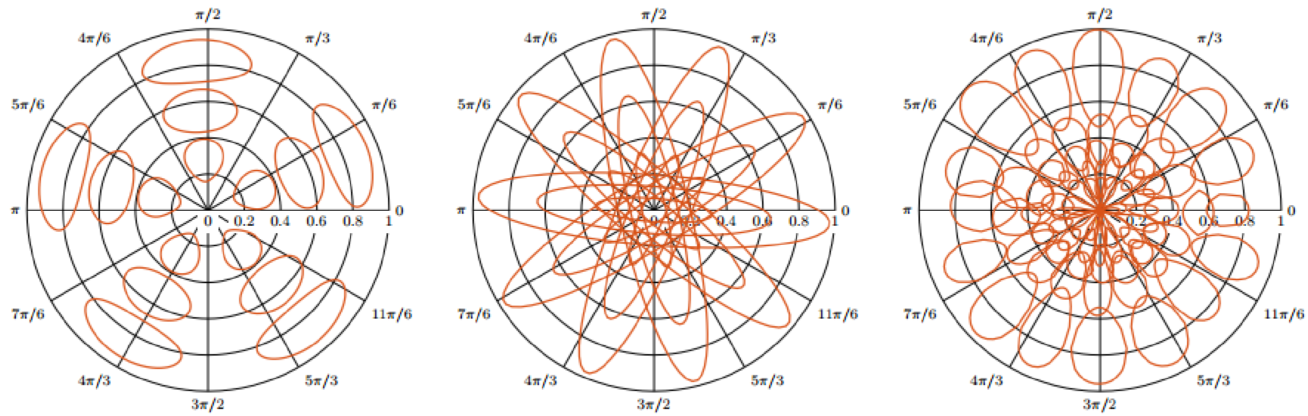
P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio. Graph Attention Networks. NeuralPS 2018.

Spatial Methods for Graph CNN

- **MoNet: A general framework for spatial methods**
  - **Define multiple kernel functions, parameterized or not, to measure the similarity between target node and other nodes**
  - **Convolution kernels are the weights of these kernel functions**

Convolution kernel

$$(f \star g)(x) = \sum_{j=1}^{J} \boxed{g_j} D_j(x) f$$



$\exp(-\frac{1}{2}(\mathbf{u}-\bar{\mathbf{u}}_j)^\top \begin{pmatrix} \bar{\sigma}_\rho^2 & \\ & \bar{\sigma}_\theta^2 \end{pmatrix}^{-1} (\mathbf{u}-\bar{\mathbf{u}}_j))$ $\exp(-\frac{1}{2}\mathbf{u}^\top \mathbf{R}_{\bar{\theta}_j} \begin{pmatrix} \bar{\alpha} & \\ & 1 \end{pmatrix} \mathbf{R}_{\bar{\theta}_j}^\top \mathbf{u})$ $\exp(-\frac{1}{2}(\mathbf{u}-\boldsymbol{\mu}_j)^\top \boldsymbol{\Sigma}_j^{-1} (\mathbf{u}-\boldsymbol{\mu}_j))$

F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model CNNs. CVPR 2017.

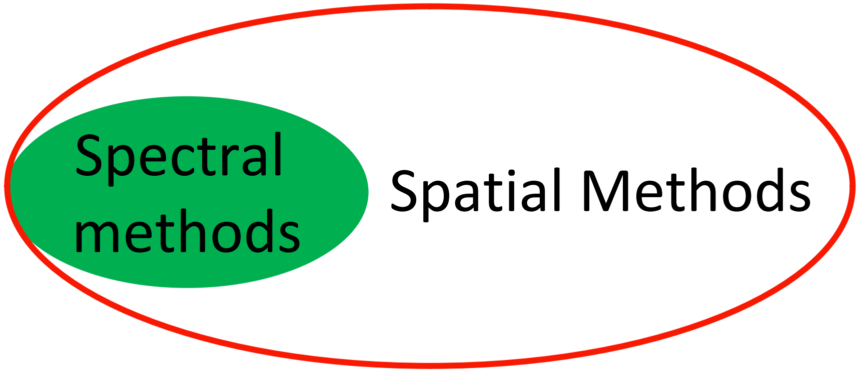
Our method:

Graph Convolutional Networks using Heat Kernel for Semi-supervised Learning

（IJCAI 2019, KRR-GSTR, August 15）

Spectral methods vs. Spatial methods

- **Connections**
  - **Spectral methods are special cases of spatial methods**



$$(f \star g)(x) = \sum_{j=1}^{J} g_j \boxed{D_j(x)} f$$

Kernel function: Characterizing the similarity or distance among nodes

- **Difference**
  - **Spectral methods define kernel functions via an explicit space transformation, i.e., projecting into spectral space**
  - **Spatial methods directly define kernel functions**

Spectral methods: Recap

- **Spectral CNN**

$$y = U g_\theta U^T x = (\theta_1 \boxed{u_1 u_1^T} + \theta_2 \boxed{u_2 u_2^T} + \cdots + \theta_n \boxed{u_n u_n^T})x$$

- **ChebNet**

$$y = (\theta_0 I + \theta_1 L + \theta_2 L^2 + \cdots + \theta_{K-1} L^{K-1})x$$

- **GCN**

$$y = \theta(I - L)x$$

Question:

Why GCN with less parameters performs better than ChebNet?

Graph Signal Processing: filter

- **Smoothness of a signal x over graph is measured by**

$$x^T L x = \sum_{(u,v)\in E} A_{uv} \left(\frac{x_u}{\sqrt{d_u}} - \frac{x_v}{\sqrt{d_v}} \right)^2$$

$\lambda_i = u_i^T L u_i$ can be viewed as the frequency of u_i

- **Basic filters**
  - $u_i u_i^T (1 \leq i \leq n)$ **are a set of basic filters**
  - **For a graph signal x, the basic filter $u_i u_i^T$ only allows the component with frequency λ_i passes**

$$x = \alpha_1 u_1 + \alpha_2 u_2 + \cdots + \alpha_n u_n,$$

$$u_i u_i^T x = \alpha_i u_i$$

Combined filters: High-pass vs. Low-pass

- **Combined filters**
  - A linear combination of basic filters

$$\theta_1 u_1 u_1^T + \theta_2 u_2 u_2^T + \cdots + \theta_n u_n u_n^T$$

  - L^k is a combined filter with the coefficients $\{\lambda_i^k\}_{i=1}^n$
  - L^k assign high coefficients to basic filters with high-frequency, i.e., L^k is a high-pass filter

- GCN only consider $k = 0$ and $k = 1$, avoiding the boosting effect to basic filters with high-frequency
  - Behaving as a low-pass combined filter

Our method: GraphHeat

- **Low-pass combined filters**
  - $\{e^{-skL}\}$, where s is scaling parameter, and k is order
  - e^{-sL} is heat kernel over graph, which defines the similarity among nodes via heat diffusion over graph

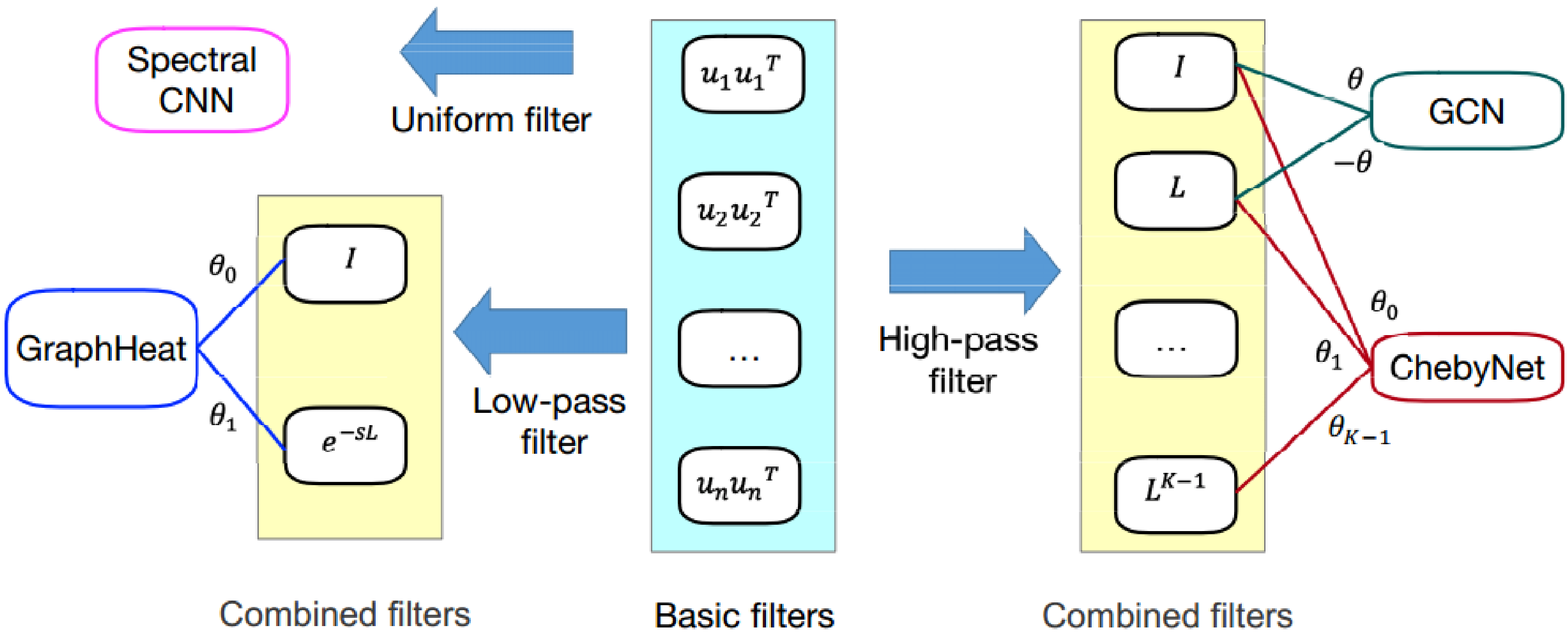
$$e^{-sL} = Ue^{-s\Lambda}U^T, \Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_n)$$

  - The basic filter $u_i u_i^T (1 \leq i \leq n)$ has the coefficient $e^{-s\lambda_i}$, suppressing signals with high-frequency

B. Xu, H. Shen, Q. Cao, K. Cen, X. Cheng. Graph Convolutional Networks using Heat Kernel for Semi-supervised Learning, IJCAI 2019.

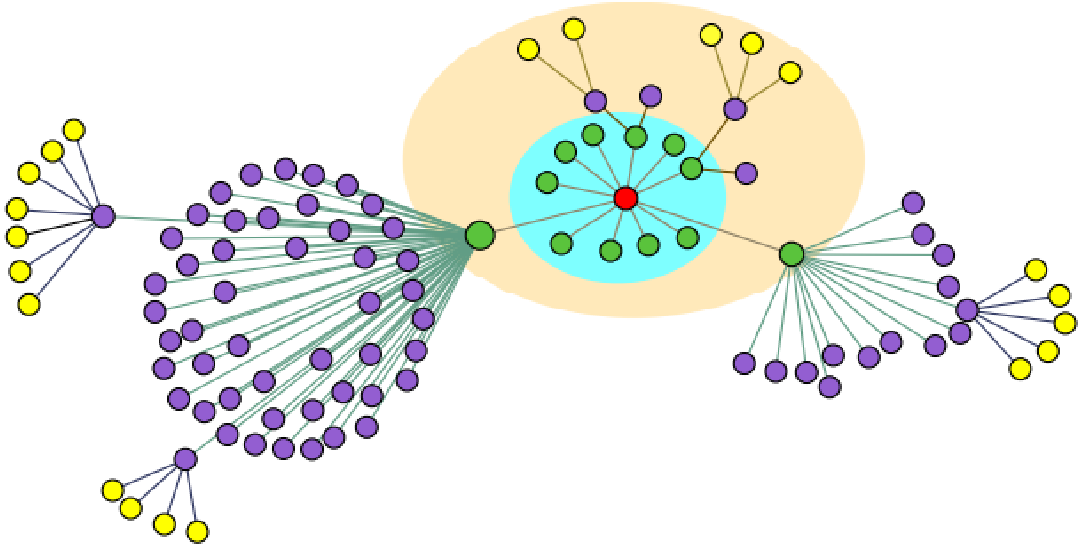
GraphHeat vs. baseline methods

- **Compared with baseline methods**



GraphHeat vs. baseline methods

- **Neighborhood**
  - **GCN and ChebNet determine neighborhood according to the hops away from center node, i.e., in an order-style**
    - **Nodes in different colors**
  - **GraphHeat determines neighborhood according to the similarity function by heat diffusion over graph**
    - **Nodes in different circles**



Experimental results

- **Results at the task of node classification**

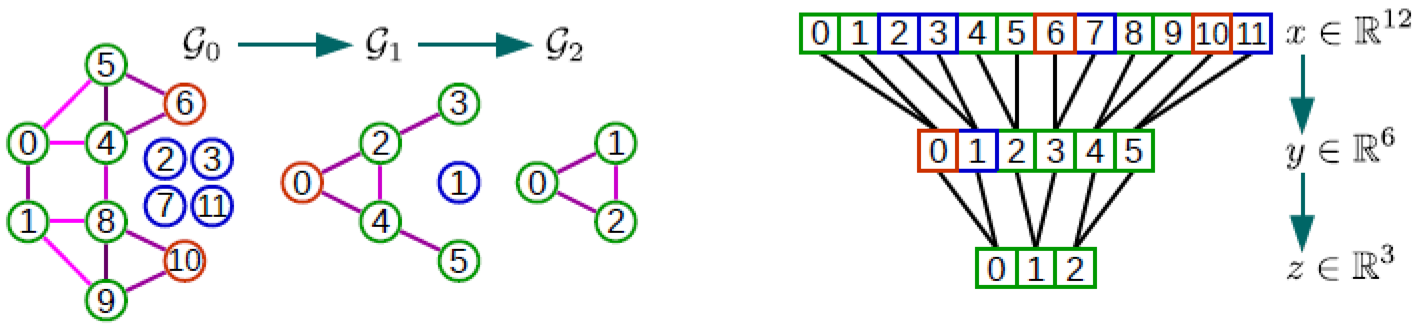
Method	Cora	Citeseer	Pubmed
MLP	55.1%	46.5%	71.4%
ManiReg	59.5%	60.1%	70.7%
SemiEmb	59.0%	59.6%	71.7%
LP	68.0%	45.3%	63.0%
DeepWalk	67.2%	43.2%	65.3%
ICA	75.1%	69.1%	73.9%
Planetoid	75.7%	64.7%	77.2%
ChebyNet	81.2%	69.8%	74.4%
GCN	81.5%	70.3%	79.0%
MoNet	81.7±0.5%	—	78.8±0.3%
GAT	83.0±0.7%	72.5±0.7%	79.0±0.3%
GraphHeat	**83.7%**	**72.5%**	**80.5%**

GraphHeat achieves state-of-the-art performance on the task of node classification on the three benchmark datasets

Graph Pooling

Graph Pooling via graph coarsening

- **Graph coarsening**
  - **Merging nodes into clusters and take each cluster as a super node**

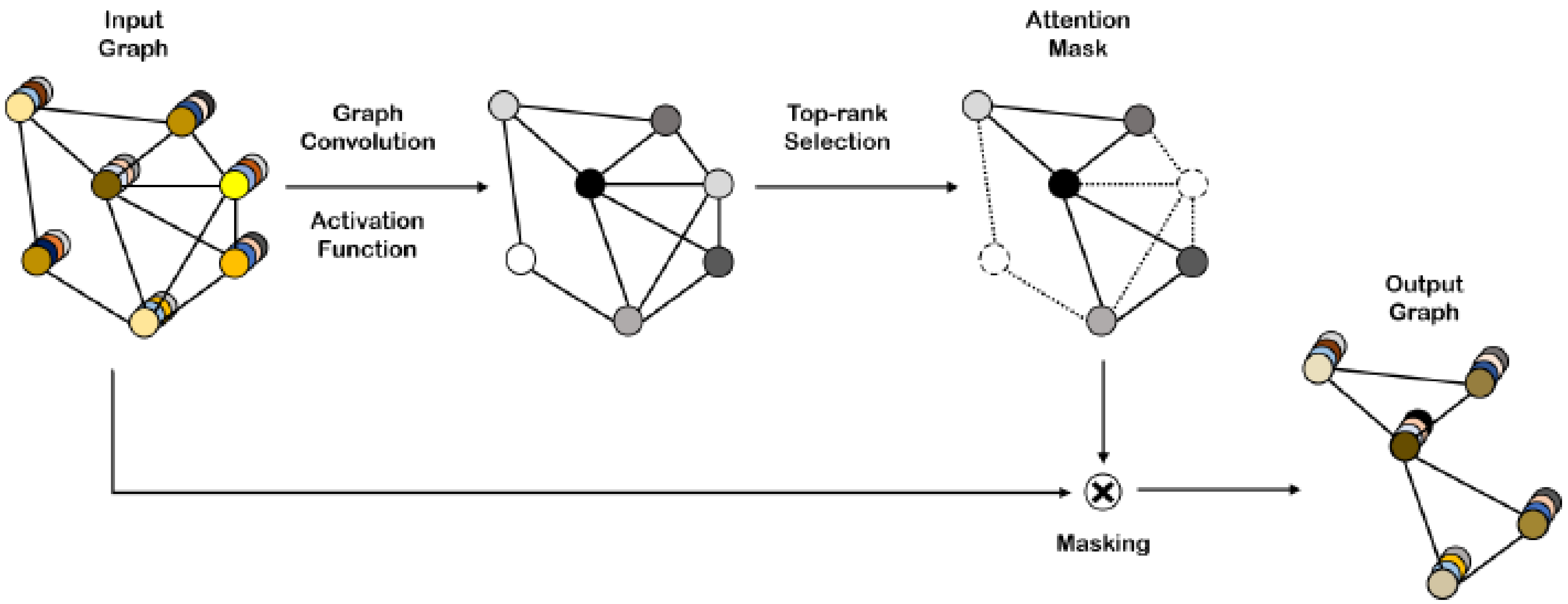


  - **Node merging could be done a priori or during the training process of graph convolutional neural networks, e.g, DiffPooling**

Ying, R., You, J., Morris, C., Ren, X., Hamilton, W. L., and Leskovec, J. Hierarchical graph representation learning with differentiable pooling, NeuralPS 2018

Graph pooling via node selection

- **Node selection**
  - **Learn a metric to quantity the importance of nodes and select several nodes according to the learned metric**



J. Lee, I. Lee, J. Kang. Self-attention Graph Pooling, ICML 2019.

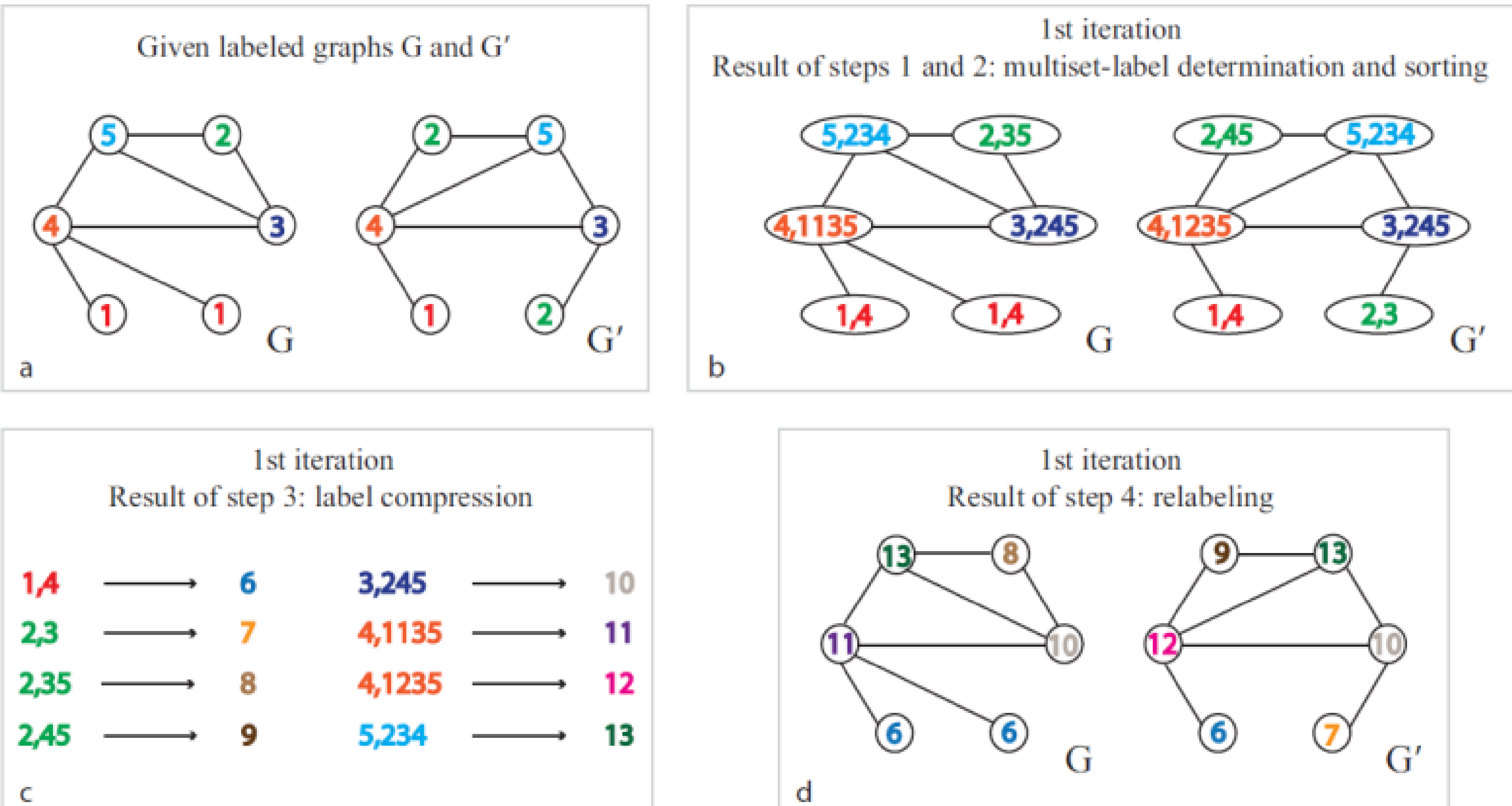
Discussions

Question 1: Does structure matters?

- **CNN learns stationary local patterns. How about graph CNN?**
  - **Both spectral methods and spatial methods fail to offer explicit clues or possibility to extract structural patterns**
  - **Instead, it seems that graph CNNs aim to learn the way in which features of neighboring nodes diffuse to the center node**
    - **Context representation**
  - **Explicitly correlate graph CNN with structural patterns, e.g., motif-based graph CNN, or graph CNN on heterogeneous networks**

Question 2: Context representation?

- **Weisfeiler-Lehman isomorphism Test: WL Test**



Is graph CNN a soft version of WL test, working on networks with real-value node attributes instead of discrete labels?

K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks? ICLR 2019.

Question 3: Future applications?

- **Three major scenarios**
  - **Node-level**
    - Node classification: predict the label of nodes according to several labeled nodes and graph structure
    - Link prediction: predict the existence or occurrence of links among pairs of nodes
  - **Graph-level**
    - Graph classification: predict the label of graphs via learning a graph representation using graph CNN
  - **Signal-level**
    - Signal classification, similar to image classification which is signal-level scenario on a grid-like network

Acknowledgement

Bingbing Xu

Qi Cao

Keting Cen

Yunqi Qiu

Xueqi Cheng

Thank you for your attentions!