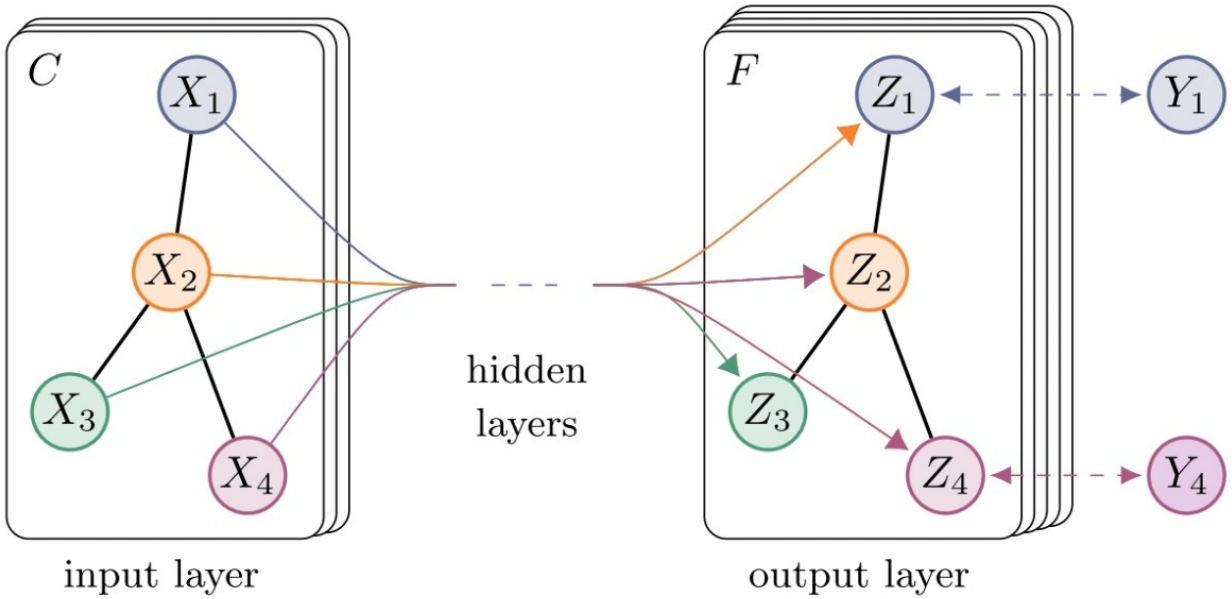
Graph Convolutional Network(GCN)



$$m_{\mathcal{N}(u)}^k = Aggragate\left(\{h_u^k | v \in \mathcal{N}(u)\}\right) = \frac{\sum_{v \in \mathcal{N}(u)} h_u^k}{|\mathcal{N}(u)|}$$

$$m_{\mathcal{N}(u)}^k = Aggragate\left(\{h_u^k | v \in \mathcal{N}(u)\}\right)$$

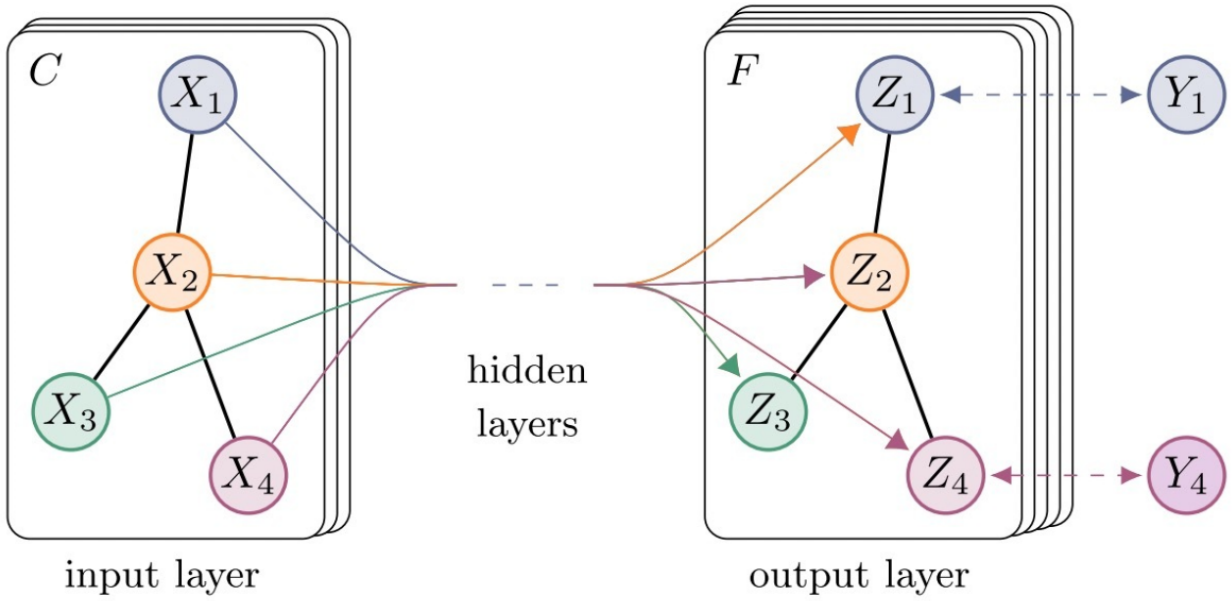
$$= \sum_{v \in \mathcal{N}(u)} \frac{h_u^k}{\sqrt{|\mathcal{N}(u)| \cdot |\mathcal{N}(v)|}}$$

$$h_u^{k+1} = Update\left(h_u^k, m_{\mathcal{N}(u)}^k\right)$$

$$= \sigma(W_u^k \frac{h_u^k}{|\mathcal{N}(u)|} + W_m^k m_{\mathcal{N}(u)}^k)$$

https://tkipf.github.io/graph-convolutional-networks/

Graph Convolutional Network(GCN)



$$m_{\mathcal{N}(u)}^{k} = Aggragate\left(\{h_u^k | v \in \mathcal{N}(u)\}\right)$$

$$= \sum_{v \in \mathcal{N}(u)} \frac{h_u^k}{\sqrt{|\mathcal{N}(u)| \cdot |\mathcal{N}(v)|}}$$

$$h_u^{k+1} = Update\left(h_u^k, m_{\mathcal{N}(u)}^k\right)$$

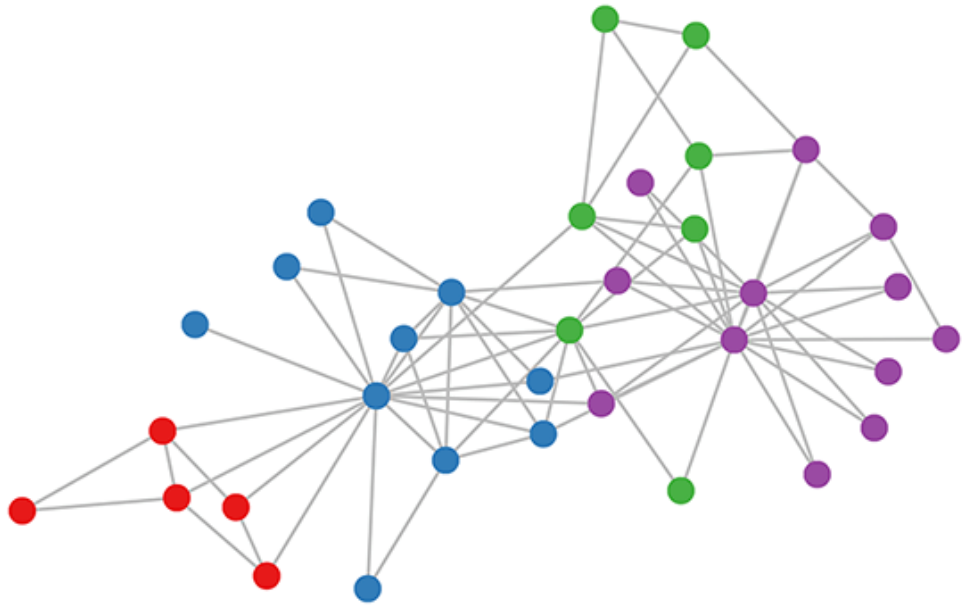
$$= \sigma\left(W_u^k \frac{h_u^k}{|\mathcal{N}(u)|} + W_m^k m_{\mathcal{N}(u)}^k\right)$$

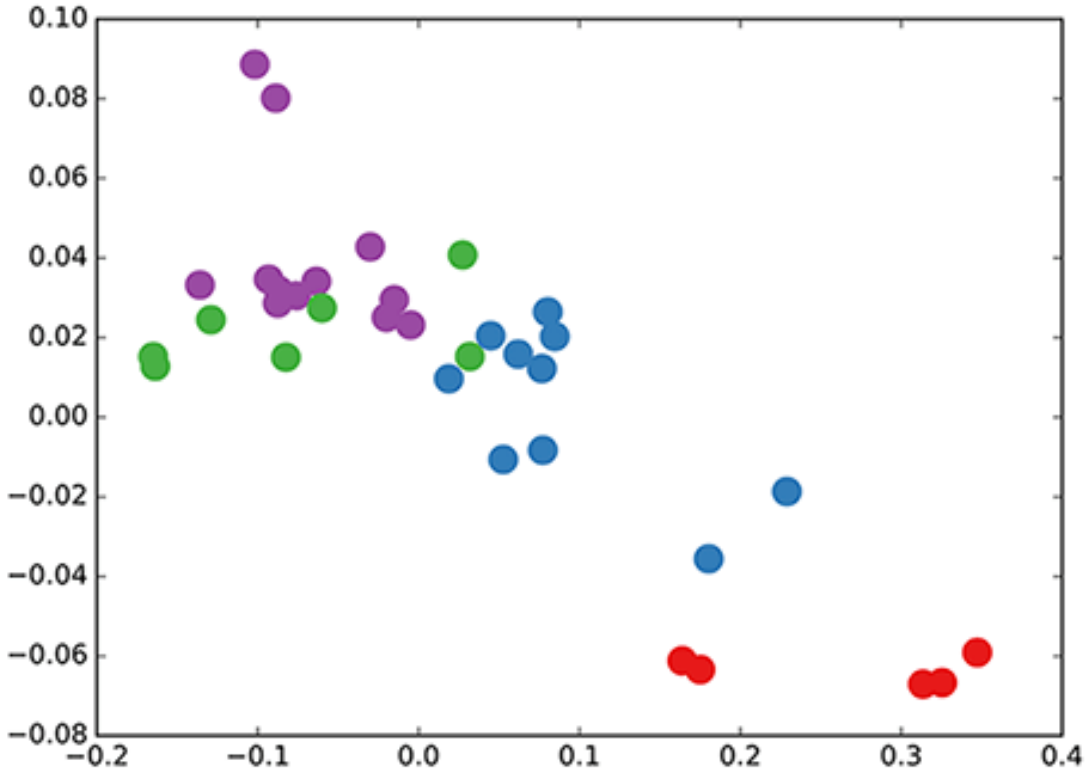
$$H^{k+1} = \sigma\left(\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} H^k W^k\right)$$

$$\hat{A} = A + I \qquad \hat{D} = diag\left(Degree\left(\hat{A}\right)\right)$$

https://tkipf.github.io/graph-convolutional-networks/

No Training – just graph convolution



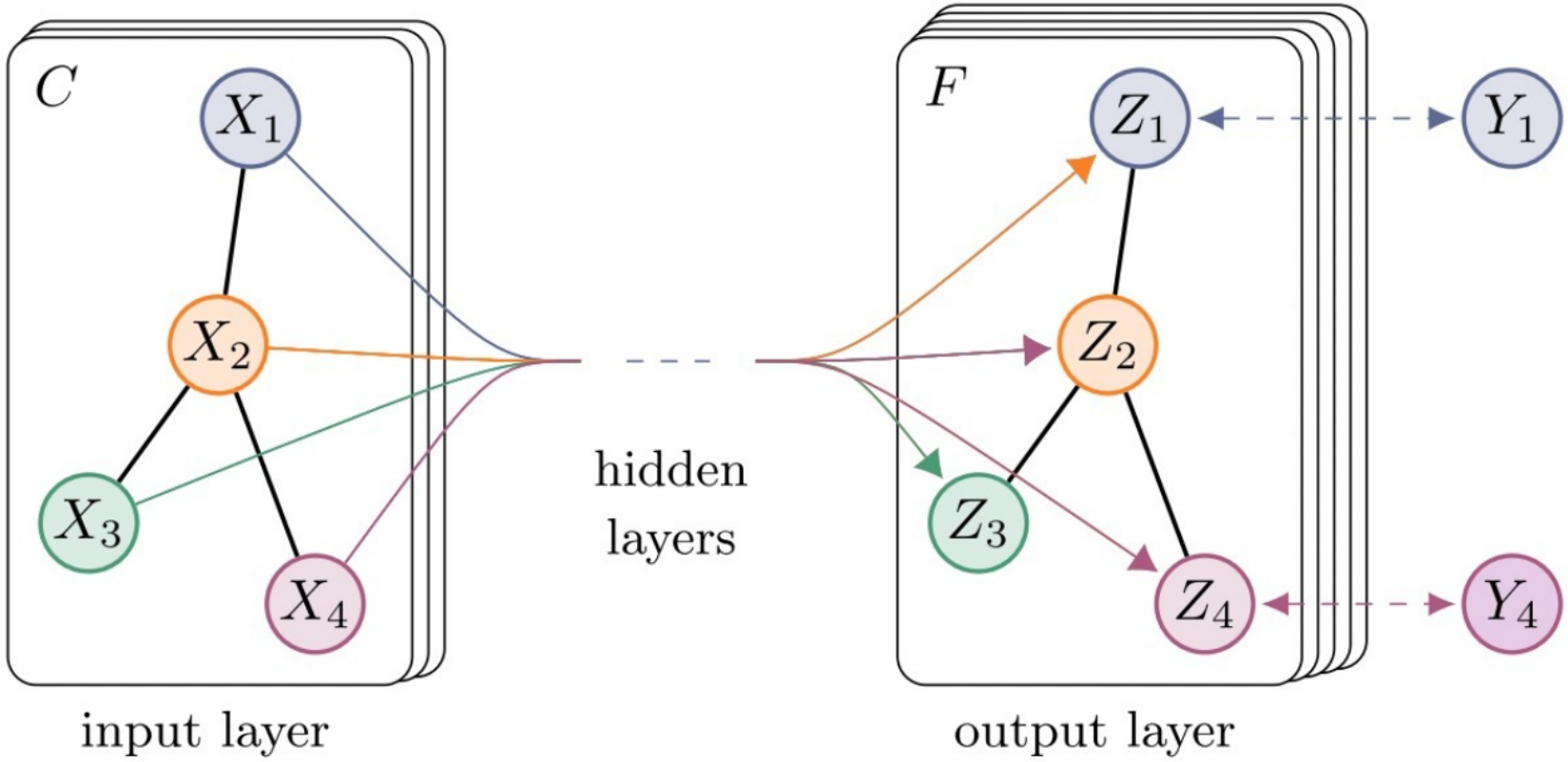


$$Z = \tanh\left(\hat{A}\,\tanh\left(\hat{A}\,\tanh\left(\hat{A}XW^{(0)}\right)W^{(1)}\right)W^{(2)}\right)$$

$$Z = Hash\left(\sum_{v \in N(u)} h_v\right)$$

https://tkipf.github.io/graph-convolutional-networks/

Semi Supervised

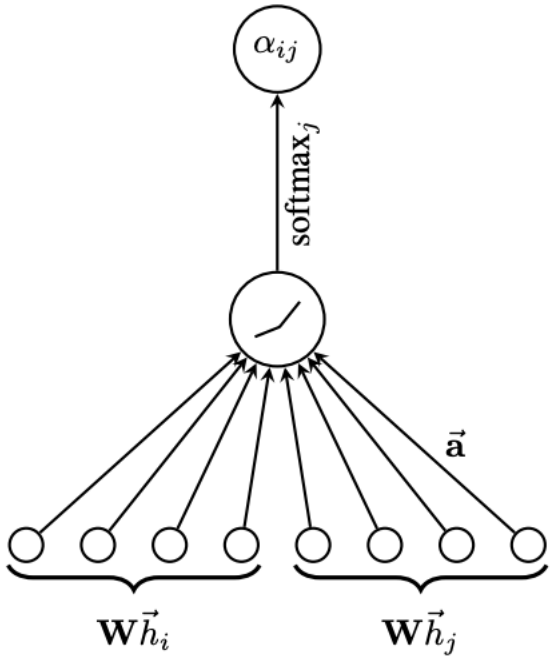


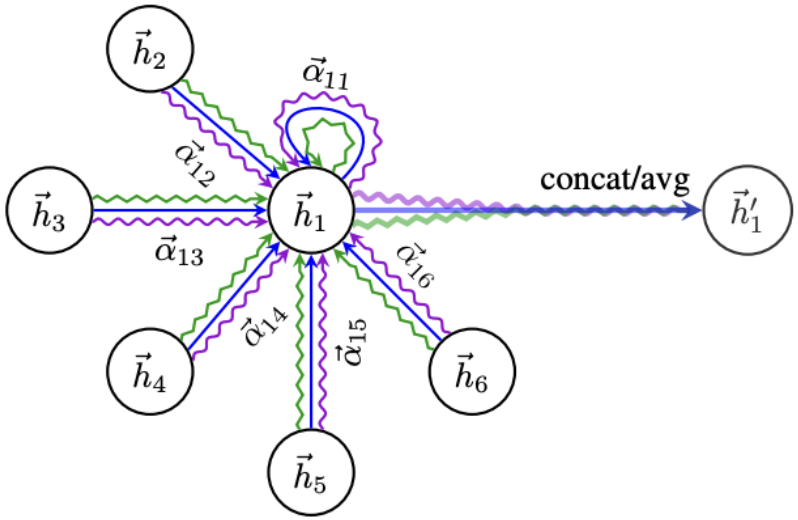
Results

Table 2: Summary of results in terms of classification accuracy (in percent).

Method	**Citeseer**	**Cora**	**Pubmed**	**NELL**
ManiReg [3]	60.1	59.5	70.7	21.8
SemiEmb [28]	59.6	59.0	71.1	26.7
LP [32]	45.3	68.0	63.0	26.5
DeepWalk [22]	43.2	67.2	65.3	58.1
ICA [18]	69.1	75.1	73.9	23.1
Planetoid* [29]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0** (38s)	**66.0** (48s)
GCN (rand. splits)	67.9 ± 0.5	80.1 ± 0.5	78.9 ± 0.7	58.4 ± 1.7

Graph Attention Network





$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \alpha_{u,v} \mathbf{h}_v$$

$$\alpha_{u,v} = \frac{\exp\left(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_u \oplus \mathbf{W}\mathbf{h}_v]\right)}{\sum_{v' \in \mathcal{N}(u)} \exp\left(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_u \oplus \mathbf{W}\mathbf{h}_{v'}]\right)}, \qquad \alpha_{u,v} = \frac{\exp\left(\mathbf{h}_u^\top \mathbf{W}\mathbf{h}_v\right)}{\sum_{v' \in \mathcal{N}(u)} \exp\left(\mathbf{h}_u^\top \mathbf{W}\mathbf{h}_{v'}\right)},$$

$$\alpha_{u,v} = \frac{\exp\left(\mathrm{MLP}(\mathbf{h}_u, \mathbf{h}_v)\right)}{\sum_{v' \in \mathcal{N}(u)} \exp\left(\mathrm{MLP}(\mathbf{h}_u, \mathbf{h}_{v'})\right)},$$

https://arxiv.org/abs/1710.10903

Graph Attention Network

Transductive

Method	Cora	Citeseer	Pubmed
MLP	55.1%	46.5%	71.4%
ManiReg (Belkin et al., 2006)	59.5%	60.1%	70.7%
SemiEmb (Weston et al., 2012)	59.0%	59.6%	71.7%
LP (Zhu et al., 2003)	68.0%	45.3%	63.0%
DeepWalk (Perozzi et al., 2014)	67.2%	43.2%	65.3%
ICA (Lu & Getoor, 2003)	75.1%	69.1%	73.9%
Planetoid (Yang et al., 2016)	75.7%	64.7%	77.2%
Chebyshev (Defferrard et al., 2016)	81.2%	69.8%	74.4%
GCN (Kipf & Welling, 2017)	81.5%	70.3%	**79.0%**
MoNet (Monti et al., 2016)	81.7 ± 0.5%	—	78.8 ± 0.3%
GCN-64*	81.4 ± 0.5%	70.9 ± 0.5%	**79.0** ± 0.3%
GAT (ours)	**83.0** ± 0.7%	**72.5** ± 0.7%	**79.0** ± 0.3%

Inductive

Method	PPI
Random	0.396
MLP	0.422
GraphSAGE-GCN (Hamilton et al., 2017)	0.500
GraphSAGE-mean (Hamilton et al., 2017)	0.598
GraphSAGE-LSTM (Hamilton et al., 2017)	0.612
GraphSAGE-pool (Hamilton et al., 2017)	0.600
GraphSAGE*	0.768
Const-GAT (ours)	0.934 ± 0.006
GAT (ours)	**0.973** ± 0.002

https://arxiv.org/abs/1710.10903

Comparison

GCN

$$m_{\mathcal{N}(u)}^{k} = Aggragate\left(\{h_u^k | v \in \mathcal{N}(u)\}\right) = \frac{\sum_{v \in \mathcal{N}(u} h_u^k}{|\mathcal{N}(u)|}$$

$$m_{\mathcal{N}(u)}^{k} = Aggragate\left(\{h_u^k | v \in \mathcal{N}(u)\}\right)$$

$$= \sum_{v \in \mathcal{N}(u)} \frac{h_u^k}{\sqrt{|\mathcal{N}(u)| \cdot |\mathcal{N}(v)|}}$$

Deep Set

$$\mathbf{m}_{\mathcal{N}(u)} = \mathrm{MLP}_\theta\left(\sum_{v \in N(u)} \mathrm{MLP}_\phi(\mathbf{h}_v)\right)$$

Graph Attention：

$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \alpha_{u,v} \mathbf{h}_v$$

https://arxiv.org/pdf/1703.06114.pdf

Extension: Update

$$\text{UPDATE}_{\text{concat}}(\mathbf{h}_u, \mathbf{m}_{\mathcal{N}(u)}) = [\text{UPDATE}_{\text{base}}(\mathbf{h}_u, \mathbf{m}_{\mathcal{N}(u)}) \oplus \mathbf{h}_u],$$

$$\text{UPDATE}_{\text{interpolate}}(\mathbf{h}_u, \mathbf{m}_{\mathcal{N}(u)}) = \boldsymbol{\alpha}_1 \circ \text{UPDATE}_{\text{base}}(\mathbf{h}_u, \mathbf{m}_{\mathcal{N}(u)}) + \boldsymbol{\alpha}_2 \odot \mathbf{h}_u,$$

$$\mathbf{h}_u^{(k)} = \text{UPDATE}\left(\mathbf{h}_u^{(k-1)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right) = \text{GRU}\left(\mathbf{h}_u^{(k-1)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right)$$

Extended Message Passing

新增连边的表征，可以更好的利用边上的信息。

$$\mathbf{h}_{(u,v)}^{(k)} = \text{UPDATE}_{\text{edge}}\left(\mathbf{h}_{(u,v)}^{(k-1)}, \mathbf{h}_u^{(k-1)}, \mathbf{h}_v^{(k-1)}, \mathbf{h}_{\mathcal{G}}^{(k-1)}\right)$$

$$\mathbf{m}_{\mathcal{N}(u)} = \text{AGGREGATE}_{\text{node}}\left(\{\mathbf{h}_{(u,v)}^{(k)} \forall v \in \mathcal{N}(u)\}\right)$$

$$\mathbf{h}_u^{(k)} = \text{UPDATE}_{\text{node}}\left(\mathbf{h}_u^{(k-1)}, \mathbf{m}_{\mathcal{N}(u)}, \mathbf{h}_{\mathcal{G}}^{(k-1)}\right)$$

$$\mathbf{h}_{\mathcal{G}}^{(k)} = \text{UPDATE}_{\text{graph}}\left(\mathbf{h}_{\mathcal{G}}^{(k-1)}, \{\mathbf{h}_u^{(k)}, \forall u \in \mathcal{V}\}, \{\mathbf{h}_{(u,v)}^{(k)} \forall (u,v) \in \mathcal{E}\}\right)$$

Extended Message Passing

全图表征的更新可以利用前述任意图池化技术

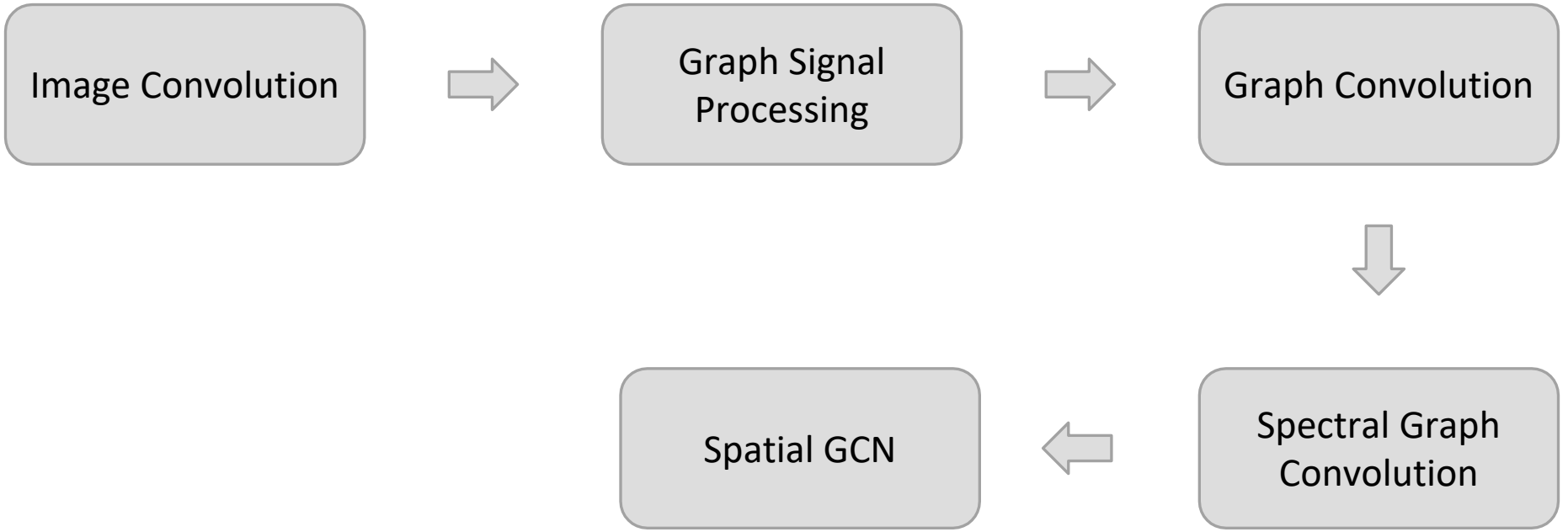
$$\mathbf{h}_{(u,v)}^{(k)} = \text{UPDATE}_{\text{edge}}\left(\mathbf{h}_{(u,v)}^{(k-1)}, \mathbf{h}_u^{(k-1)}, \mathbf{h}_v^{(k-1)}, \mathbf{h}_{\mathcal{G}}^{(k-1)}\right)$$

$$\mathbf{m}_{\mathcal{N}(u)} = \text{AGGREGATE}_{\text{node}}\left(\{\mathbf{h}_{(u,v)}^{(k)} \forall v \in \mathcal{N}(u)\}\right)$$

$$\mathbf{h}_u^{(k)} = \text{UPDATE}_{\text{node}}\left(\mathbf{h}_u^{(k-1)}, \mathbf{m}_{\mathcal{N}(u)}, \mathbf{h}_{\mathcal{G}}^{(k-1)}\right)$$

$$\mathbf{h}_{\mathcal{G}}^{(k)} = \text{UPDATE}_{\text{graph}}\left(\mathbf{h}_{\mathcal{G}}^{(k-1)}, \{\mathbf{h}_u^{(k)}, \forall u \in \mathcal{V}\}, \{\mathbf{h}_{(u,v)}^{(k)} \forall (u,v) \in \mathcal{E}\}\right)$$

History of GCN



Back to Kipf's GCN

参考和进一步阅读

1.入门博客：A Gentle Introduction to Graph Neural Networks

2.入门课本：Graph Representation Learning

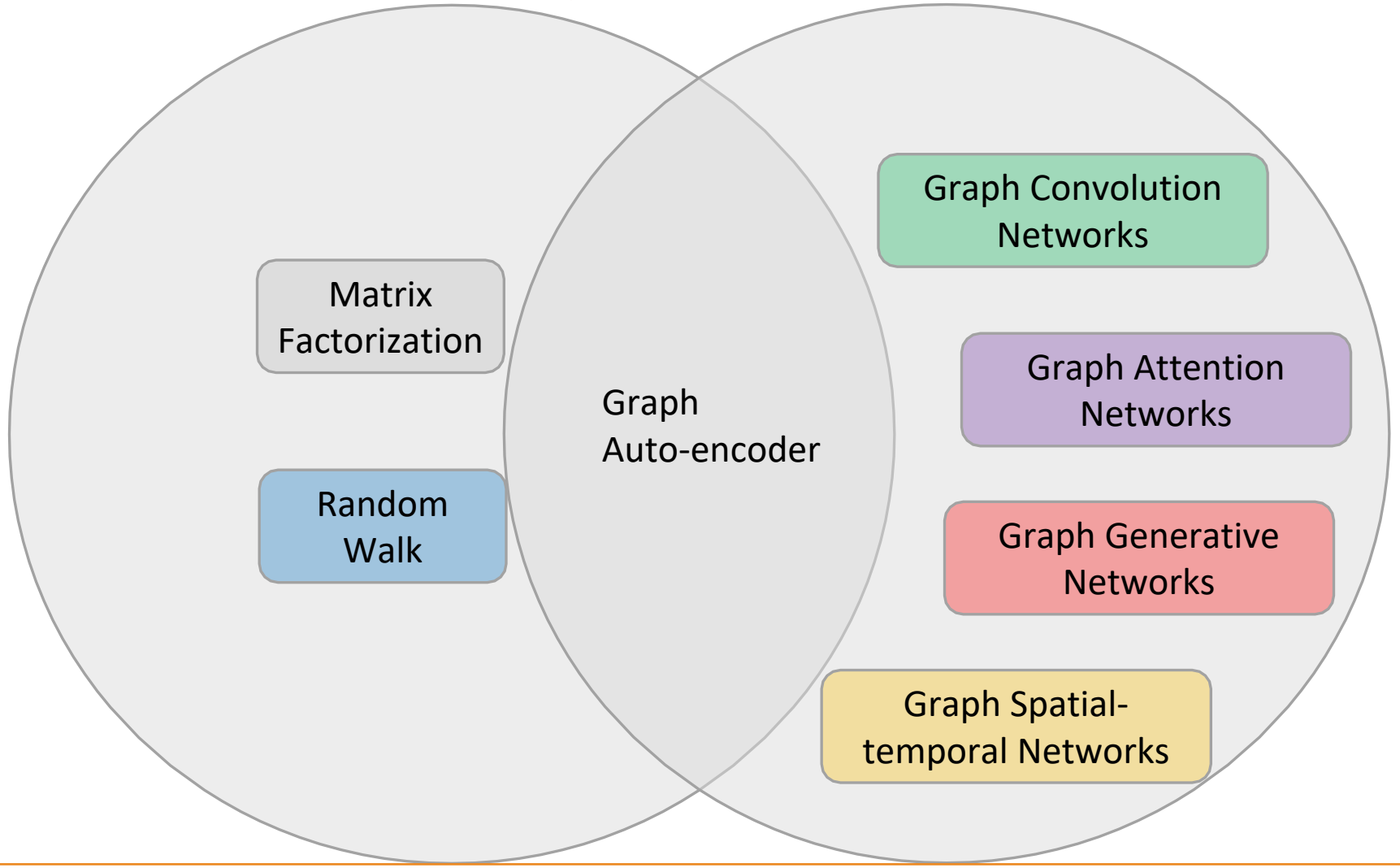
3.入门公开课：CS224W:Machine Learning with Graphs

4.最新博客：Michael Bronstein

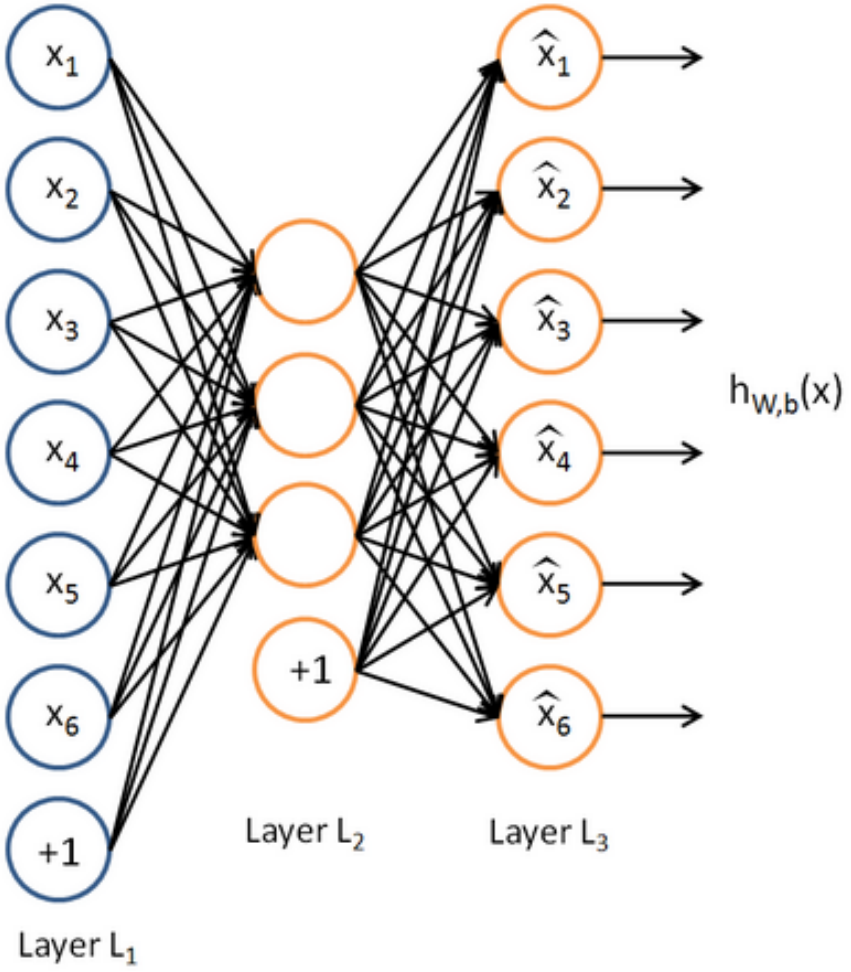
Outline

- Background
- Network Embedding
- Graph Neural Networks
  - Graph Convolution
  - Graph Attention Network
  - Other extension
- Variational Graph Autoencoder
- Graph Generation

Network Embedding v.s. Graph Neural Networks

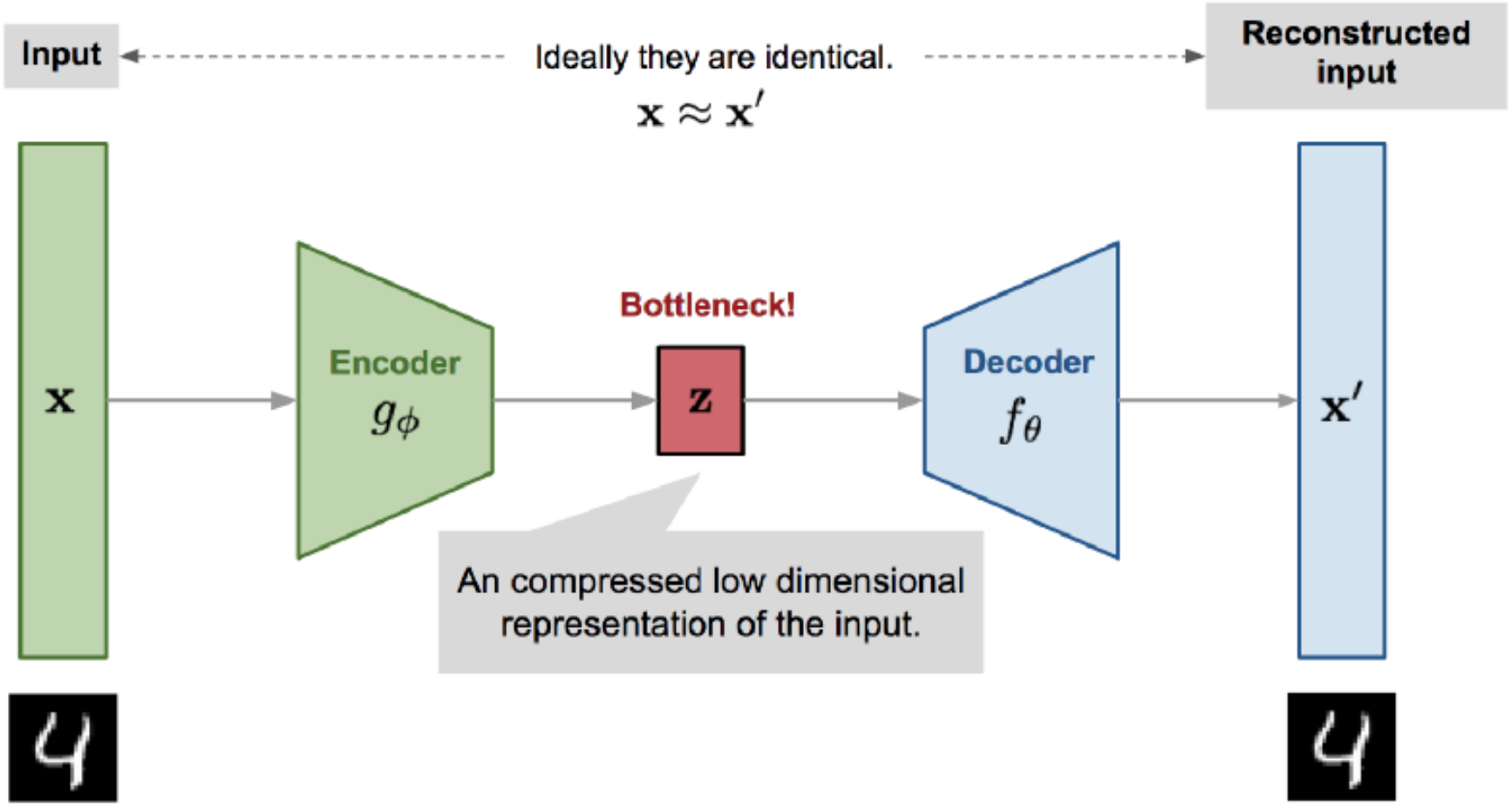


Autoencoder

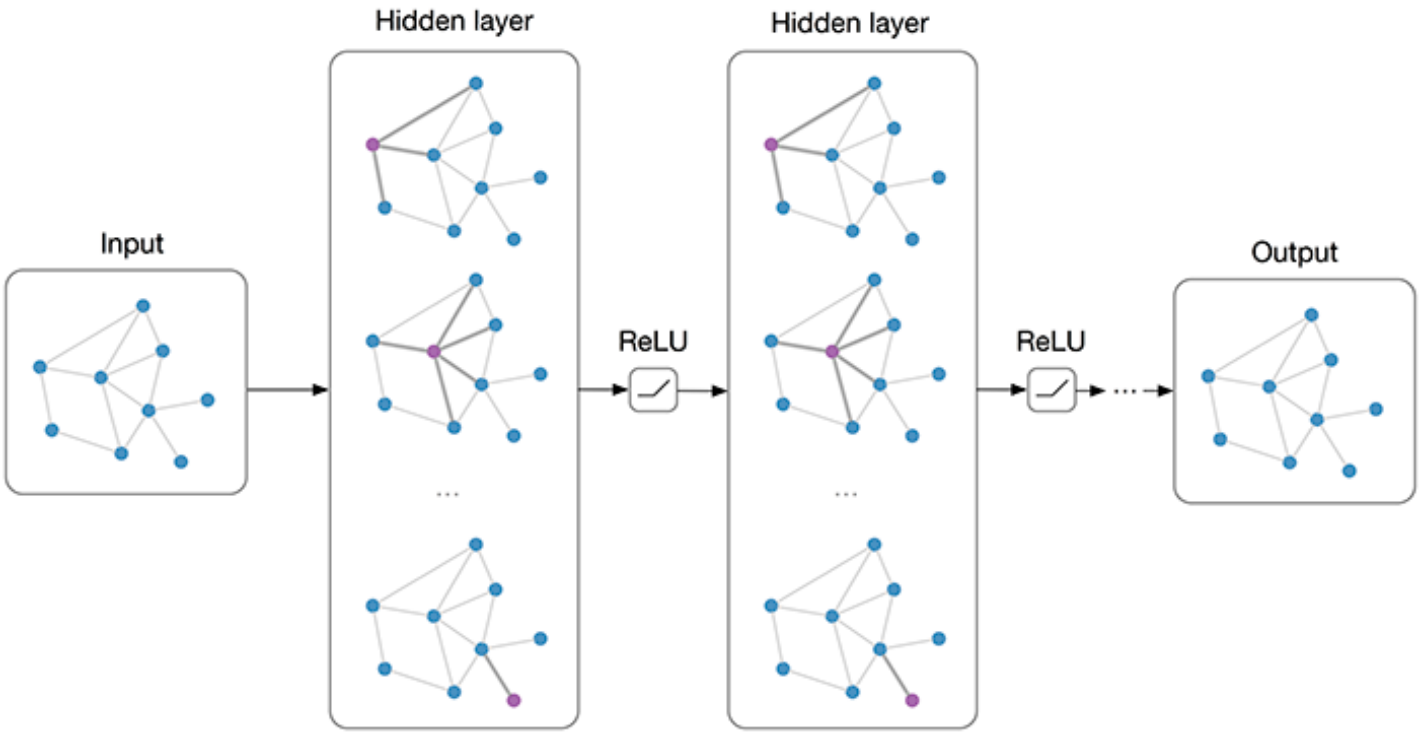


- Unsupervised learning
- Feed-back
  - Reconstruct input: $h_\theta(x) \approx x$
  - BP Learning
- Constraints

Variational Autoencoder



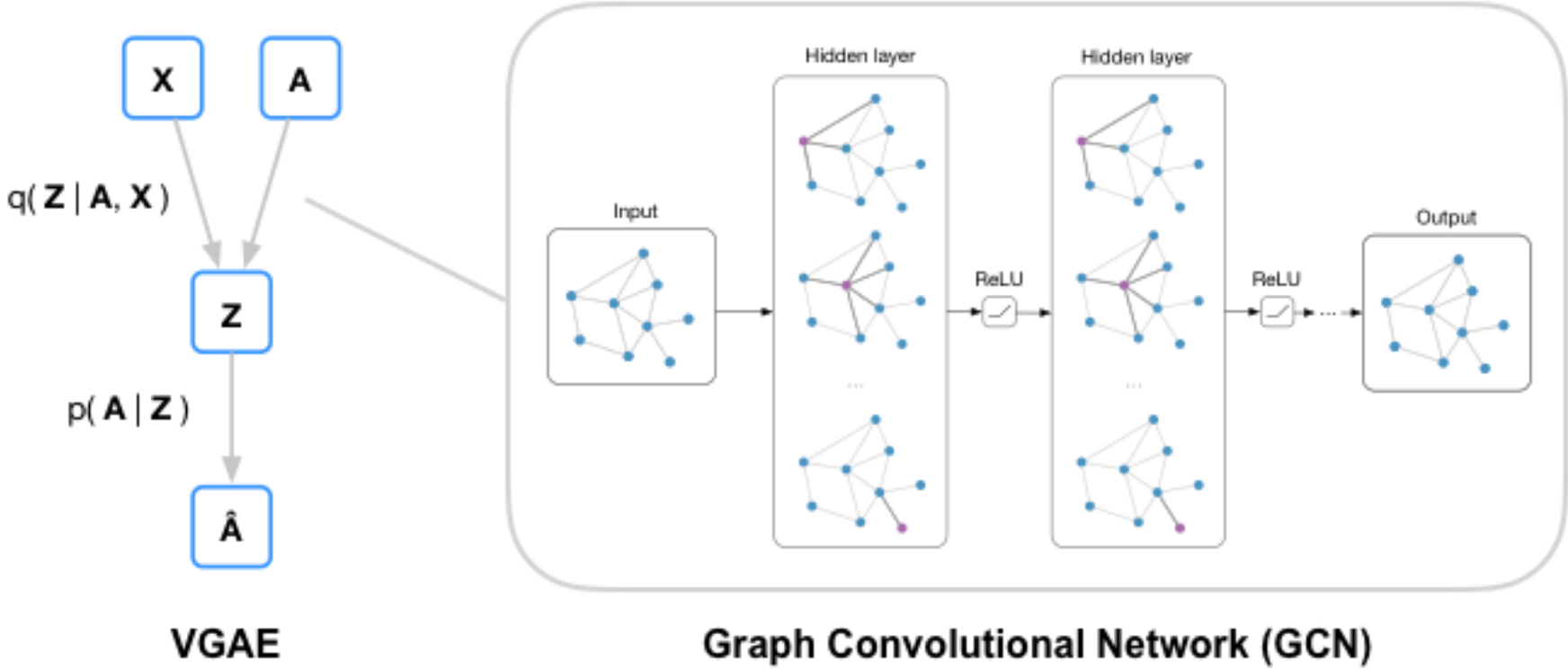
Graph Autoencoder



$$\hat{\mathbf{A}} = \sigma\left(\mathbf{Z}\mathbf{Z}^{\top}\right), \quad \text{with} \quad \mathbf{Z} = \mathrm{GCN}(\mathbf{X}, \mathbf{A})$$

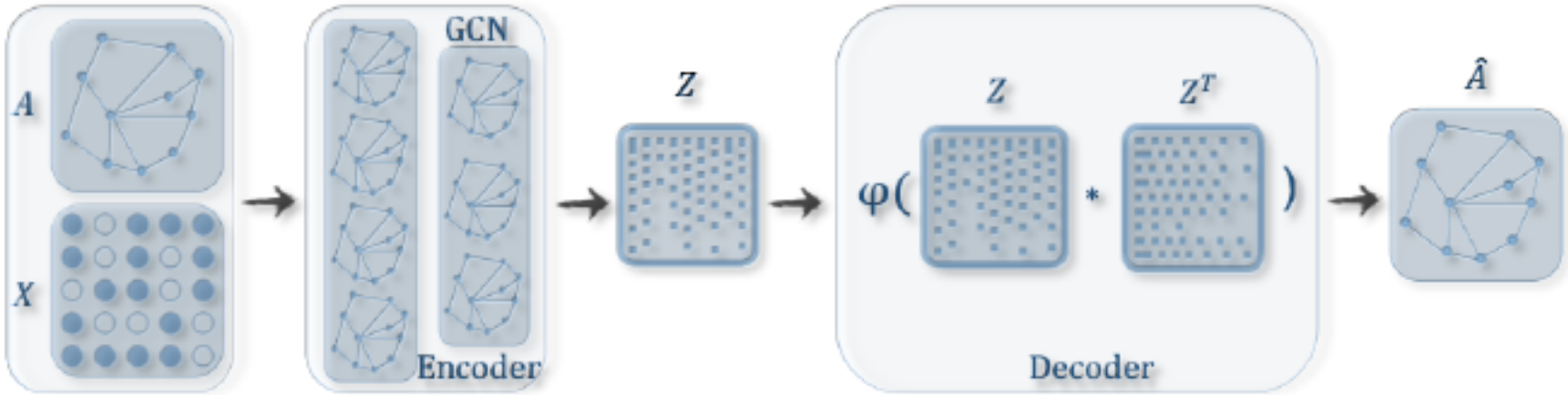
https://arxiv.org/pdf/1611.07308.pdf

Variational Graph Autoencoder



https://github.com/tkipf/gae

Variational Graph Autoencoder



Encoder：

$$q(\mathbf{Z}\,|\,\mathbf{X},\mathbf{A}) = \prod_{i=1}^{N} q(\mathbf{z}_i\,|\,\mathbf{X},\mathbf{A})\,, \quad \text{with} \quad q(\mathbf{z}_i\,|\,\mathbf{X},\mathbf{A}) = \mathcal{N}(\mathbf{z}_i\,|\,\boldsymbol{\mu}_i, \mathrm{diag}(\boldsymbol{\sigma}_i^2))$$

Decoder:
(generative model)：

$$p\,(\mathbf{A}\,|\,\mathbf{Z}) = \prod_{i=1}^{N}\prod_{j=1}^{N} p\,(A_{ij}\,|\,\mathbf{z}_i,\mathbf{z}_j)\;, \quad \text{with} \quad p\,(A_{ij}=1\,|\,\mathbf{z}_i,\mathbf{z}_j) = \sigma(\mathbf{z}_i^{\top}\mathbf{z}_j)$$

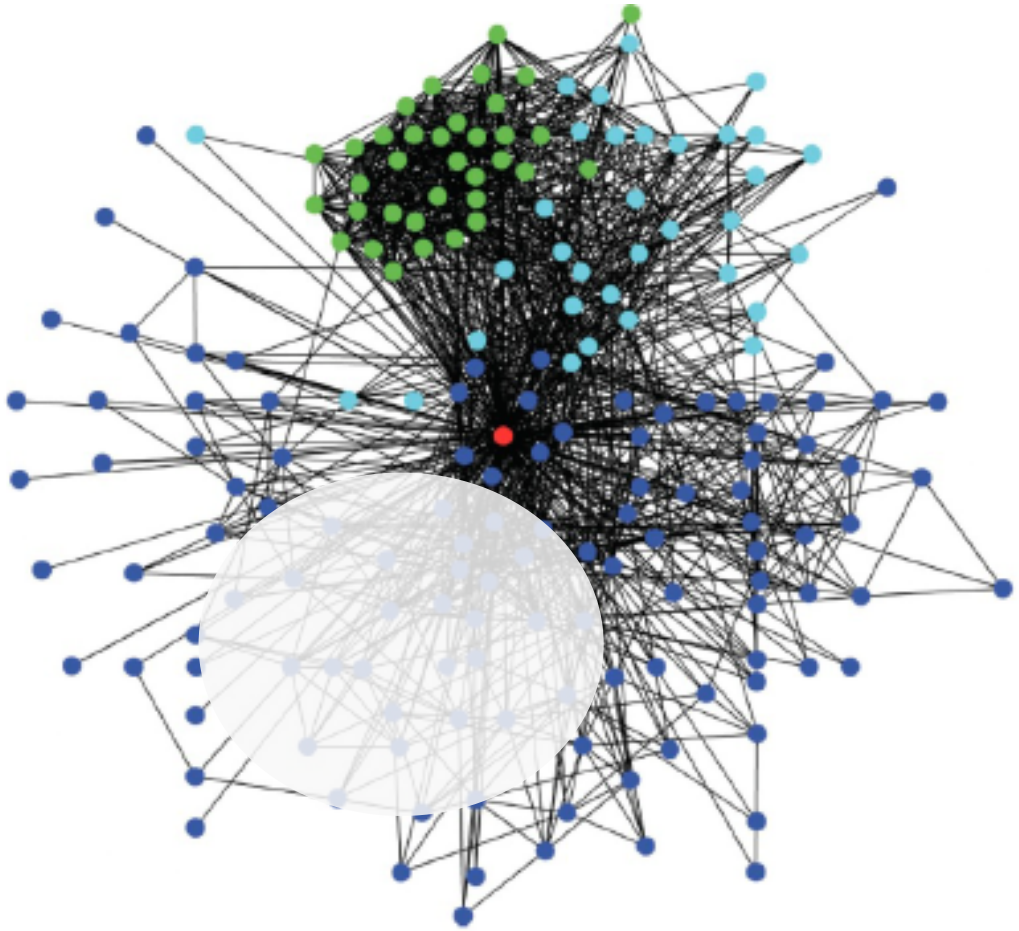
Learning:

$$\mathcal{L} = \mathbb{E}_{q(\mathbf{Z}|\mathbf{X},\mathbf{A})}\left[\log p\,(\mathbf{A}\,|\,\mathbf{Z})\right] - \mathrm{KL}\left[q(\mathbf{Z}\,|\,\mathbf{X},\mathbf{A})\,||\,p(\mathbf{Z})\right]$$

https://zhuanlan.zhihu.com/p/78340397

https://github.com/tkipf/gae

Link Prediction



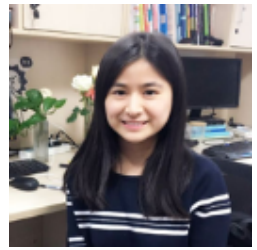
Results – Variational Graph Autoencoder

Table 1: Link prediction task in citation networks. See [1] for dataset details.

Method	Cora		Citeseer		Pubmed	
	AUC	AP	AUC	AP	AUC	AP
SC [5]	84.6 ± 0.01	88.5 ± 0.00	80.5 ± 0.01	85.0 ± 0.01	84.2 ± 0.02	87.8 ± 0.01
DW [6]	83.1 ± 0.01	85.0 ± 0.00	80.5 ± 0.02	83.6 ± 0.01	84.4 ± 0.00	84.1 ± 0.00
GAE*	84.3 ± 0.02	88.1 ± 0.01	78.7 ± 0.02	84.1 ± 0.02	82.2 ± 0.01	87.4 ± 0.00
VGAE*	84.0 ± 0.02	87.7 ± 0.01	78.9 ± 0.03	84.1 ± 0.02	82.7 ± 0.01	87.5 ± 0.01
GAE	91.0 ± 0.02	92.0 ± 0.03	89.5 ± 0.04	89.9 ± 0.05	$\mathbf{96.4} \pm 0.00$	$\mathbf{96.5} \pm 0.00$
VGAE	$\mathbf{91.4} \pm 0.01$	$\mathbf{92.6} \pm 0.01$	$\mathbf{90.8} \pm 0.02$	$\mathbf{92.0} \pm 0.02$	94.4 ± 0.02	94.7 ± 0.02

https://arxiv.org/pdf/1611.07308.pdf

Representation via Link Prediction



谷伟伟

JOURNAL OF SOCIAL COMPUTING
ISSN 2688-5255 04/06 pp43–51
Volume 2, Number 1, March 2021
DOI: 10.23919/JSC.2021.0001

Learning Universal Network Representation via Link Prediction by Graph Convolutional Neural Network

Weiwei Gu, Fei Gao, Ruiqi Li*, and Jiang Zhang*

Abstract: Network representation learning algorithms, which aim at automatically encoding graphs into low-dimensional vector representations with a variety of node similarity definitions, have a wide range of downstream applications. Most existing methods either have low accuracies in downstream tasks or a very limited application field, such as article classification in citation networks. In this paper, we propose a novel network representation method, named Link Prediction based Network Representation (LPNR), which generalizes the latest graph neural network and optimizes a carefully designed objective function that preserves linkage structures. LPNR can not only learn meaningful node representations that achieve competitive accuracy in node centrality measurement and community detection but also achieve high accuracy in the link prediction task. Experiments prove the effectiveness of LPNR on three real-world networks. With the mini-batch and fixed sampling strategy, LPNR can learn the embedding of large graphs in a few hours.

Key words: network representation; link prediction; deep learning

scientific reports

Check for updates

OPEN

Discovering latent node Information by graph attention network

Weiwei Gu[1], Fei Gao[2], Xiaodan Lou[2] & Jiang Zhang[2]✉

In this paper, we propose graph attention based network representation (GANR) which utilizes the graph attention architecture and takes graph structure as the supervised learning information. Compared with node classification based representations, GANR can be used to learn representation for any given graph. GANR is not only capable of learning high quality node representations that achieve a competitive performance on link prediction, network visualization and node classification but it can also extract meaningful attention weights that can be applied in node centrality measuring task. GANR can identify the leading venture capital investors, discover highly cited papers and find the most influential nodes in Susceptible Infected Recovered Model. We conclude that link structures in graphs are not limited on predicting linkage itself, it is capable of revealing latent node information in an unsupervised way once a appropriate learning algorithm, like GANR, is provided.

We are surrounded by various relations, in which, rich information is behind. For example, nodes that bridge different communities play a vital role in information sharing[1]; who you coauthor with in scientific research is highly related to the success of your research career[2,3]; some unique niche of species may become the bottleneck of the energy flows in the entire food web chain[4]; the linked web pages rather than the contents that play more vital roles in ranking web pages[5]. Thus unraveling the latent information and values of nodes is of great significance in both theory and application. It is one of the most important tasks in network analysis.

A bunch of node centrality measures have been proposed to uncover nodes' importance. For example, degree, clustering coefficient, betweenness, average distance, all those indicators can be computed directly[6]. There are also some task oriented methods such as PageRank for webpages[5], node impact factor for trade flow networks[7], energy bottleneck index[4] for food webs, and disruption index[8] for citation networks. All these ingenious indicators are hand-crafted, prior domain knowledge and human heuristics are required. Besides, they are designed only for node centrality measuring task and can hardly be transferred to solve other problems such as node classification and graph visualization.

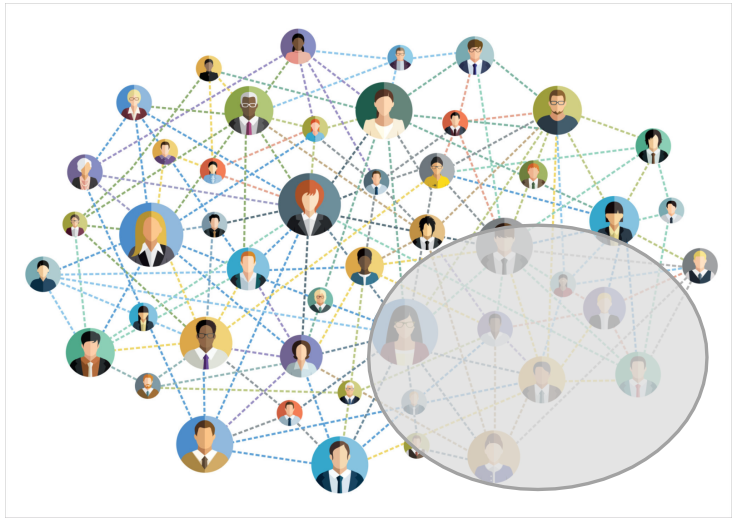
Network representation learning tackles the above mentioned problems by embedding nodes into a low-

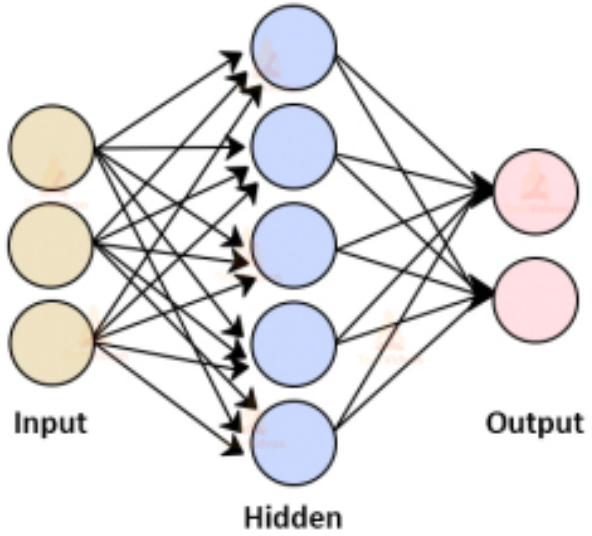
https://ieeexplore.ieee.org/document/9355034

https://www.nature.com/articles/s41598-021-85826-x

Main Idea

Graph Attention Network for Link Prediction

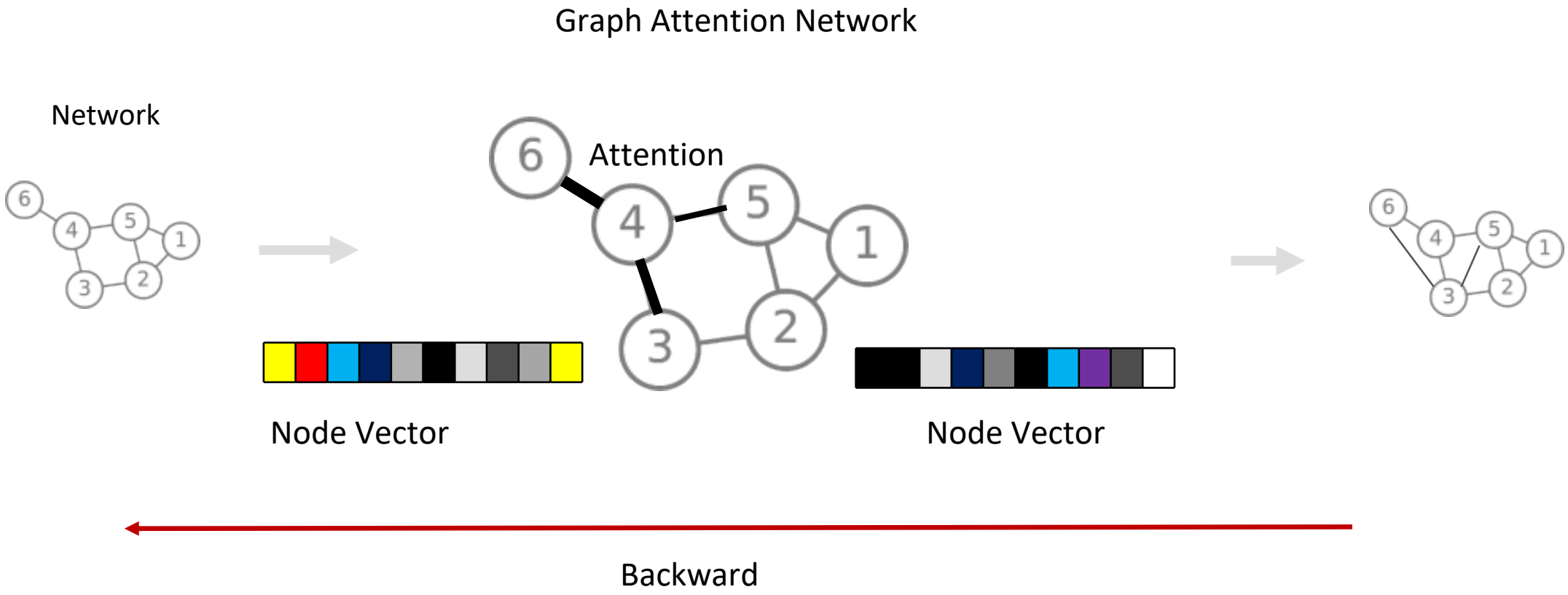




- Centrality
- Community
- Node Classification

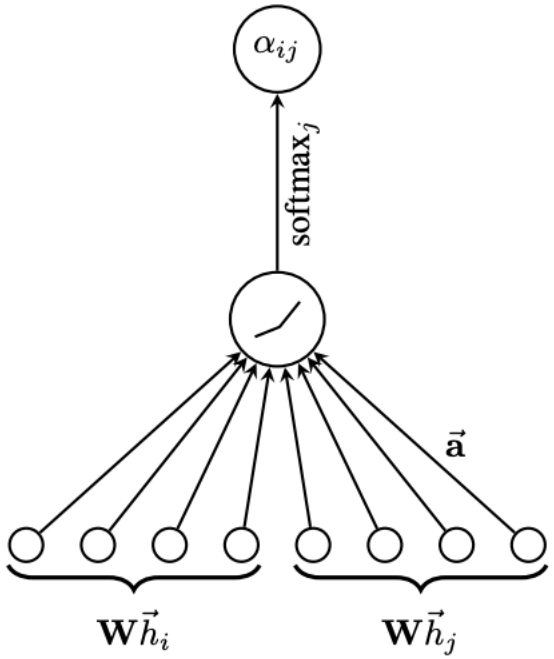
https://arxiv.org/abs/1710.10903

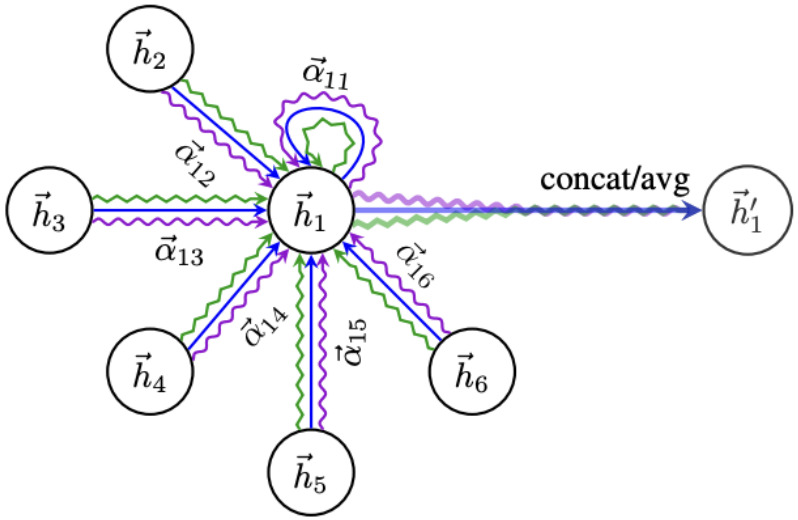
Pipe Line



Gu, W., Gao, F., Lou, X. et al. Discovering latent node Information by graph attention network. Scientific Reports 11, 6967 (2021).

Graph Attention Network





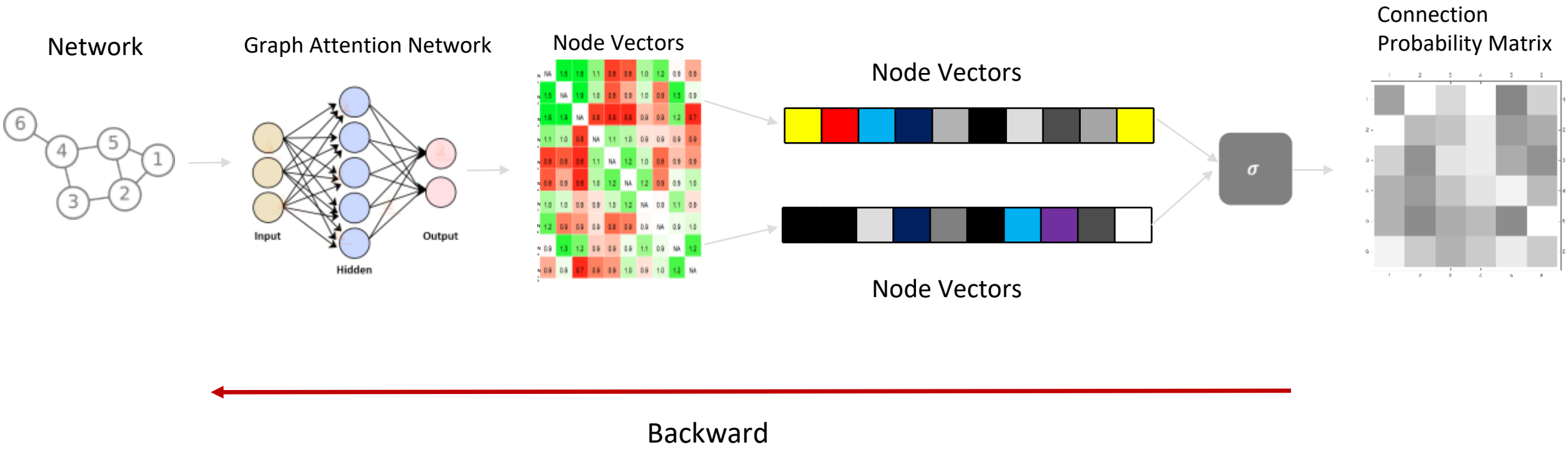
$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \alpha_{u,v} \mathbf{h}_v$$

$$\alpha_{u,v} = \frac{\exp\left(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_u \oplus \mathbf{W}\mathbf{h}_v]\right)}{\sum_{v' \in \mathcal{N}(u)} \exp\left(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_u \oplus \mathbf{W}\mathbf{h}_{v'}]\right)}, \qquad \alpha_{u,v} = \frac{\exp\left(\mathbf{h}_u^\top \mathbf{W}\mathbf{h}_v\right)}{\sum_{v' \in \mathcal{N}(u)} \exp\left(\mathbf{h}_u^\top \mathbf{W}\mathbf{h}_{v'}\right)},$$

$$\alpha_{u,v} = \frac{\exp\left(\mathrm{MLP}(\mathbf{h}_u, \mathbf{h}_v)\right)}{\sum_{v' \in \mathcal{N}(u)} \exp\left(\mathrm{MLP}(\mathbf{h}_u, \mathbf{h}_{v'})\right)},$$

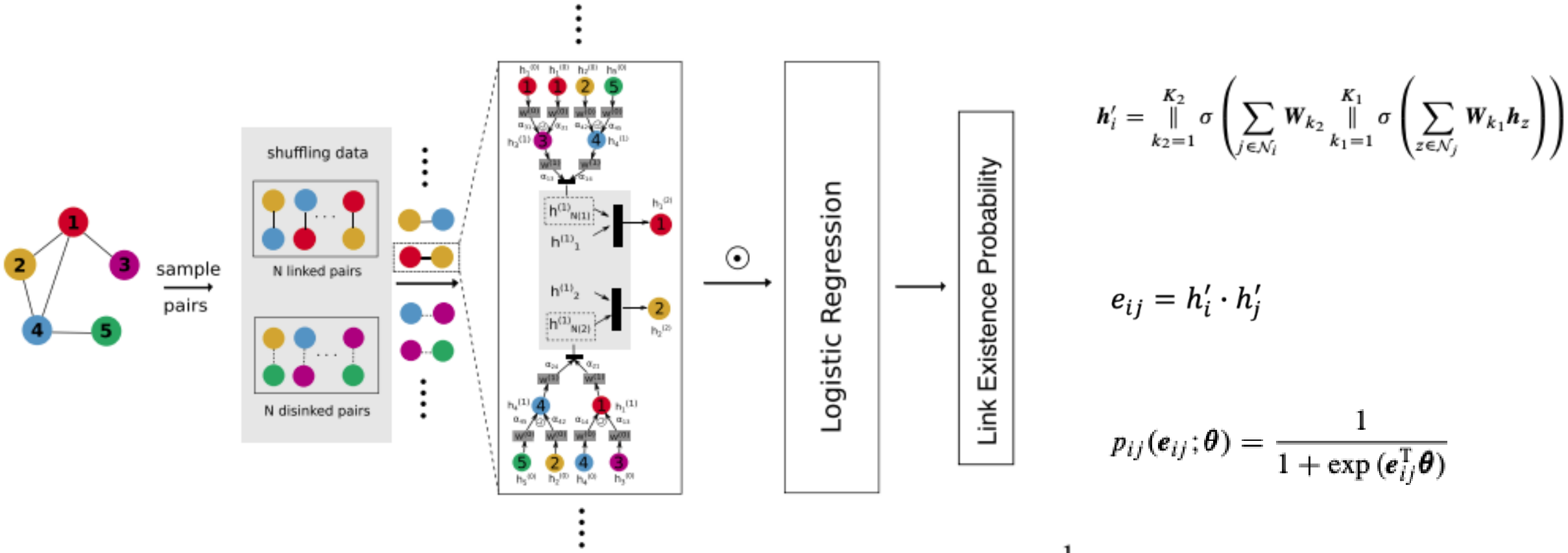
https://arxiv.org/abs/1710.10903

Pipe Line



Gu, W., Gao, F., Lou, X. et al. Discovering latent node Information by graph attention network. Scientific Reports 11, 6967 (2021).

Graph Attention Network for Large Networks



$$h_i' = \Big\Vert_{k_2=1}^{K_2} \sigma\left(\sum_{j\in\mathcal{N}_i} W_{k_2} \Big\Vert_{k_1=1}^{K_1} \sigma\left(\sum_{z\in\mathcal{N}_j} W_{k_1} h_z\right)\right)$$

$$e_{ij} = h_i' \cdot h_j'$$

$$p_{ij}(e_{ij};\theta) = \frac{1}{1+\exp\left(e_{ij}^{\mathrm{T}}\theta\right)}$$

$$\mathcal{L} = -\frac{1}{|\varepsilon|}\sum_{(i,j)\in\varepsilon} y_{i,j}\log p_{i,j} + (1-y_{i,j})\log(1-p_{i,j})$$

Gu, W., Gao, F., Lou, X. et al. Discovering latent node Information by graph attention network. Scientific Reports 11, 6967 (2021).

Results – Link Prediction

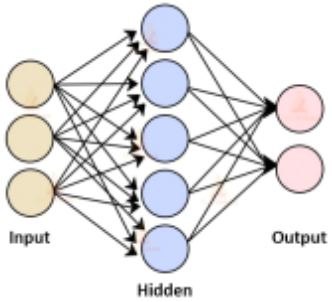
Accuracy/AUC	Cora	Citeseer	VC network	APS network
GANR (proposed)	**0.87/0.93**	**0.85**/0.91	0.80/0.90	**0.84/0.94**
RA	0.41/0.75	0.32/0.73	0.33/0.76	0.35/0.78
LINE	0.69/0.76	0.67/0.73	0.78/0.84	0.68/0.74
node2vec	0.82/0.92	0.85/0.89	0.77/0.87	0.83/0.89
GraphSAGE-mean	0.83/0.89	0.84/0.90	**0.81**/0.90	0.82/0.88
VGAE	0.75/0.91	0.75/0.90	0.76/0.88	0.76/**0.94**
ARVGA	0.73/**0.93**	0.73/**0.94**	0.72/**0.91**	0.72/0.92

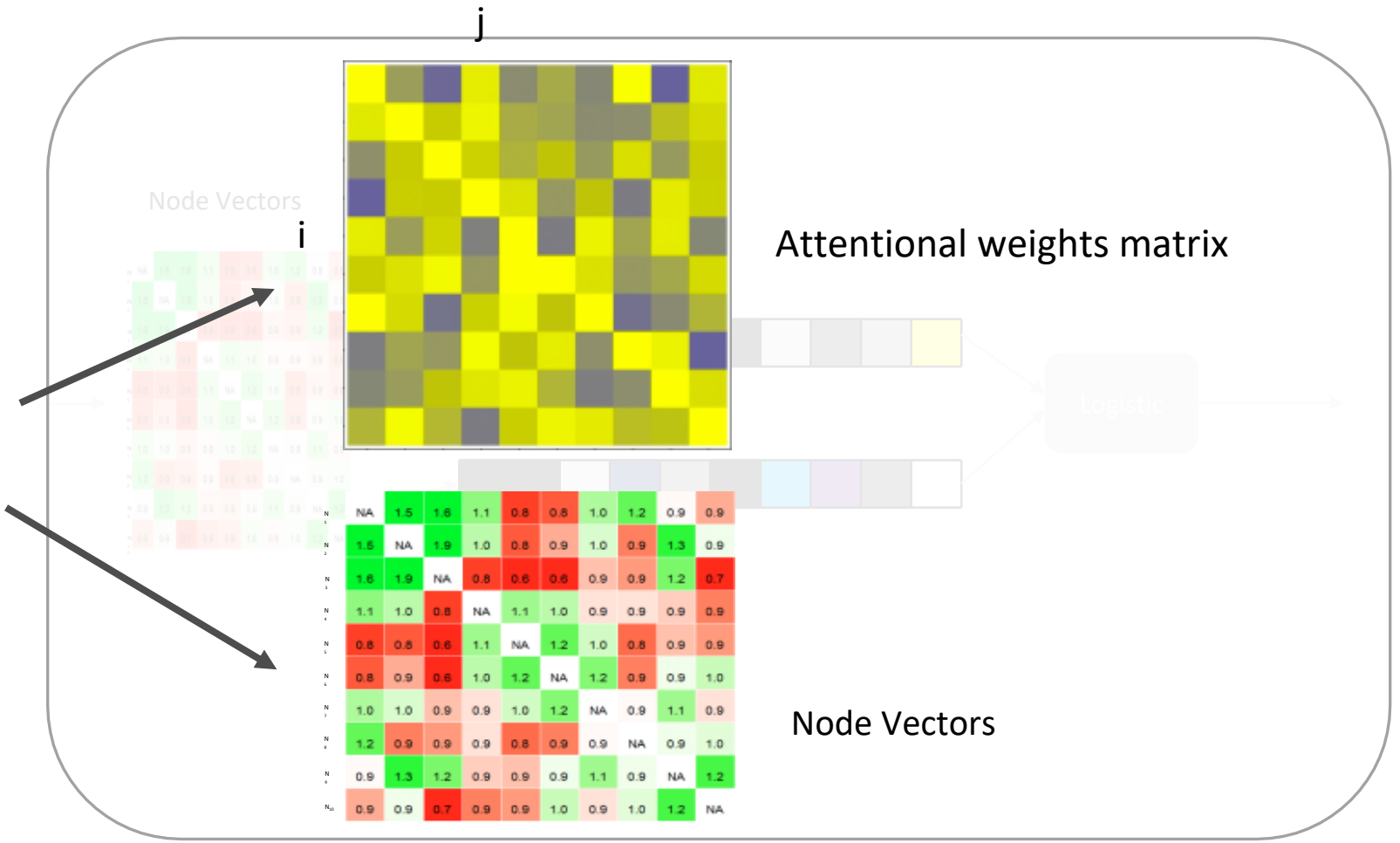
Gu, W., Gao, F., Lou, X. et al. Discovering latent node Information by graph attention network. Scientific Reports 11, 6967 (2021).

Pipe Line

Network

Graph Attention Network

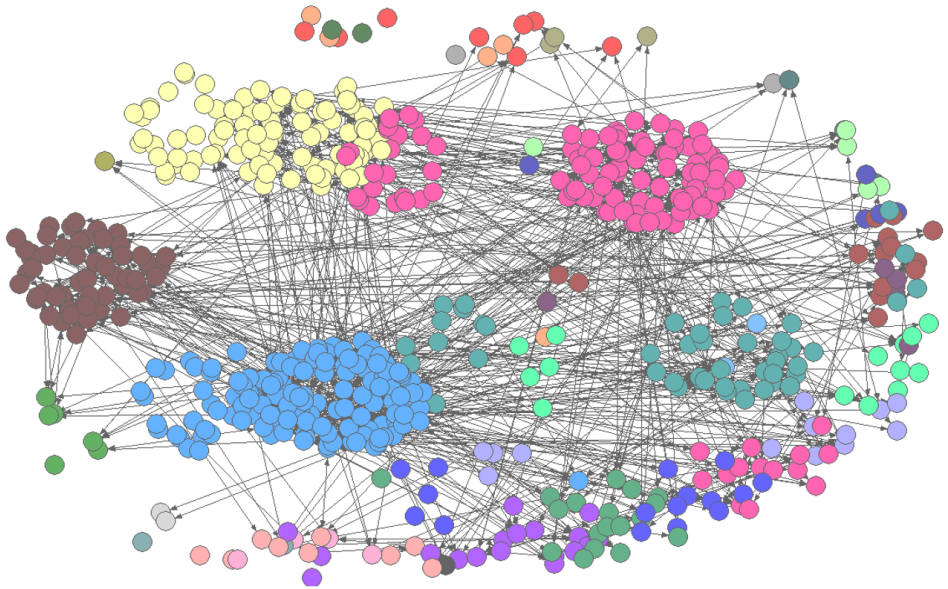




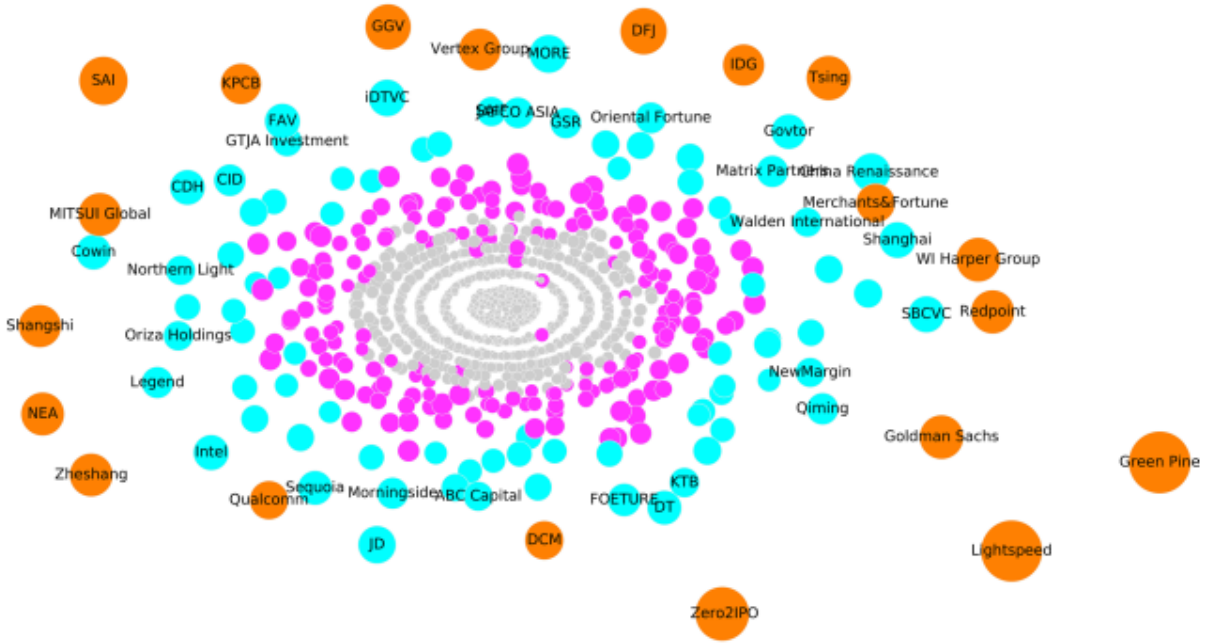
Gu, W., Gao, F., Lou, X. et al. Discovering latent node Information by graph attention network. Scientific Reports 11, 6967 (2021).

Results – Visualization

$$\tilde{c}_i = \sum_{k=1}^{K_2} \sum_{j \in \mathcal{N}_i} \alpha_{ji}^k$$



VC network: VC co-invest



Visualization for attentional matrix by MDS algorithm

Gu, W., Gao, F., Lou, X. et al. Discovering latent node Information by graph attention network. Scientific Reports 11, 6967 (2021).

Results – Centrality

Rank	VC name	Label is_elite	Rank	VC name	Label is_elite
1	MORE/Shenzhen Capital Group	Yes	9	JAFCO ASIA	Yes
2	IDG Capital	Yes	10	FOETURE Capital	Yes
3	Sequoia	Yes	11	GGV Capital	Yes
4	Legend Captital	Yes	12	Walden International	Yes
5	Goldman Sachs	Yes	13	SBCVC	Yes
6	Intel Capital	Yes	14	DFJ Venture Capital	Yes
7	Northern Light Venture Capital	Yes	15	Qiming	Yes
8	DT Capital	Yes	16	Cowin	Yes

$$\bar{c}_i = \sum_{k=1}^{K_2} \sum_{j \in \mathcal{N}_i} \alpha_{ji}^k$$

Gu, W., Gao, F., Lou, X. et al. Discovering latent node Information by graph attention network. Scientific Reports 11, 6967 (2021).

Results – Centrality

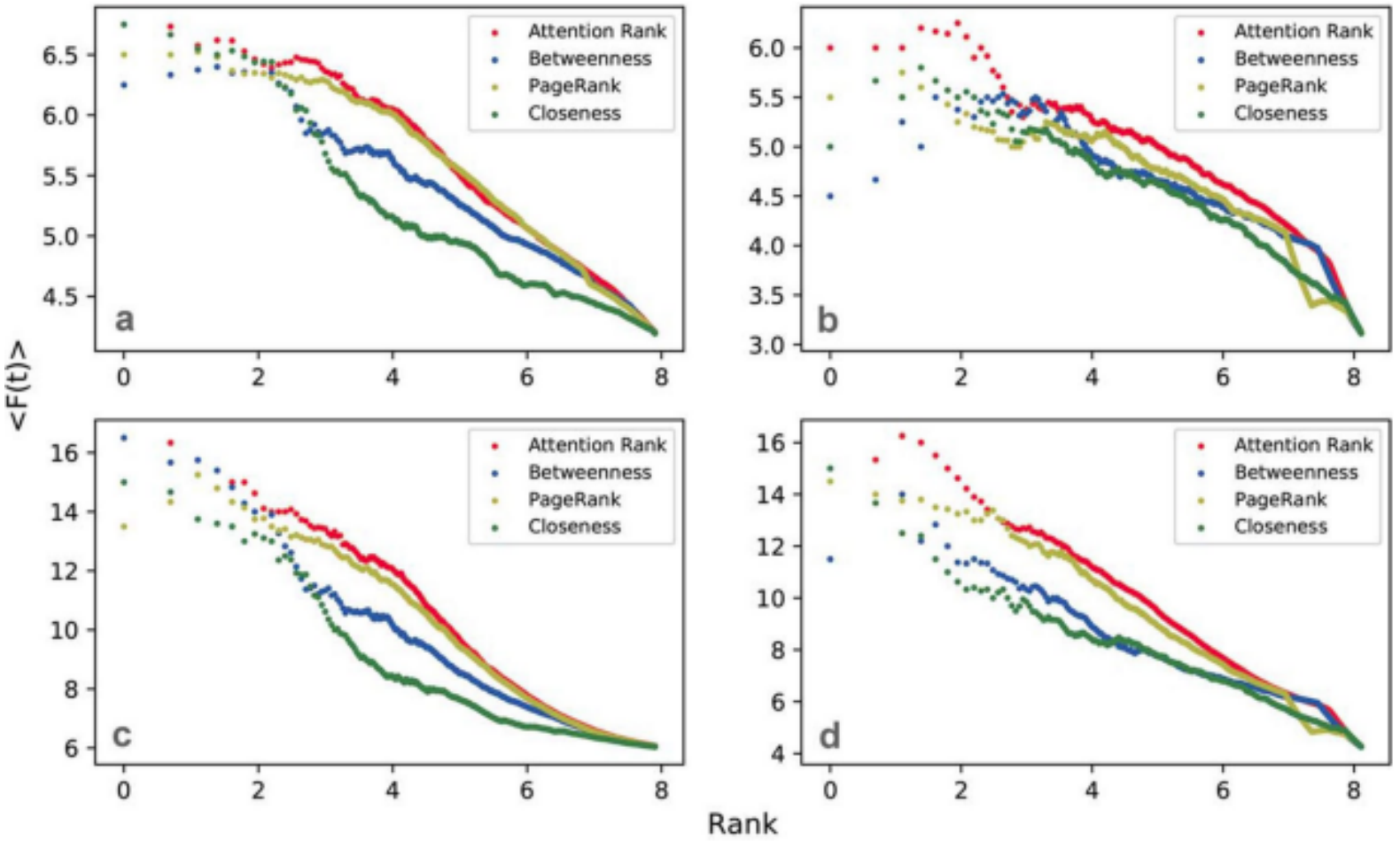


Figure 1. The comparison between the spreading power of different ranking algorithms over F(t). In panels (a) and (b), we plot the average number of F(t) for t = 3 of the top-L users as ranked by the five centrality measures of Cora (a) and Citeseer (b) graphs, while in panels (b) and (d), we plot the average number of F(t) for t = 9 of Cora (c) and Citeseer (d) graphs.

Gu, W., Gao, F., Lou, X. et al. Discovering latent node Information by graph attention network. Scientific Reports 11, 6967 (2021).

Results – Centrality

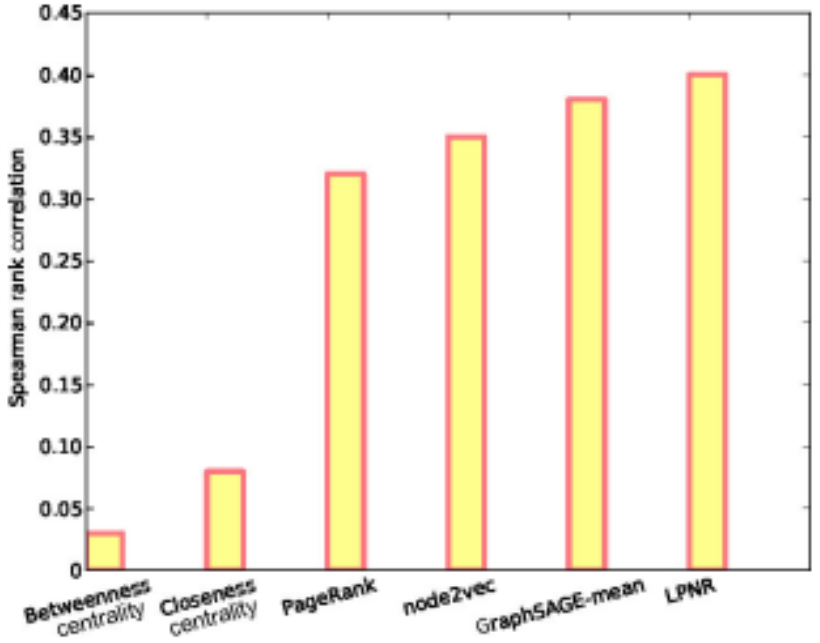


Fig. 1 Comparison of ranking performance under different ranking methods.

Dataset	Evaluation	PageRank	Closeness	Betweenness	Attention centrality
VC	Accuracy	0.65	0.60	0.58	0.72
APS	Rank coor.	0.32	0.08	0.03	0.42

Table 3. Ranking performance comparison between different unsupervised ranking algorithms.

Gu, W., Gao, F., Lou, X. et al. Discovering latent node Information by graph attention network. Scientific Reports 11, 6967 (2021).

Node Clustering

Cora dataset

GraphSAGE

Ours

Visualization for Node Representation Vectors

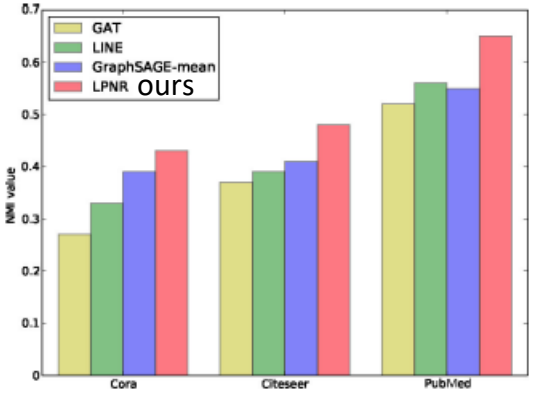
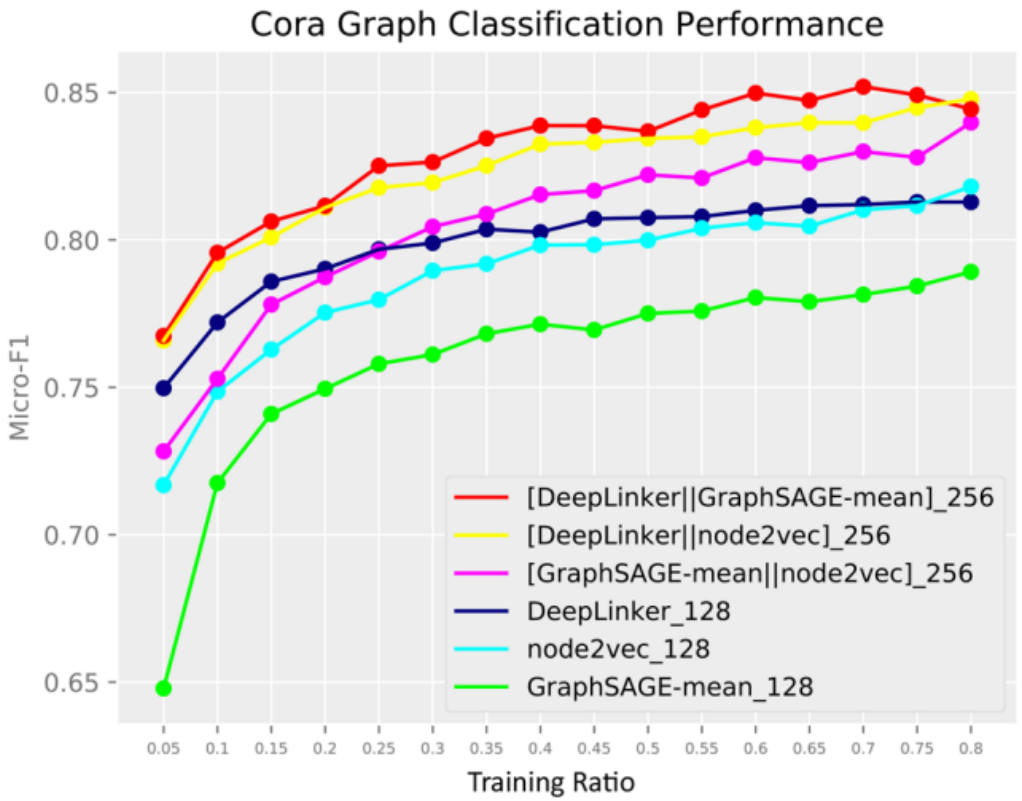
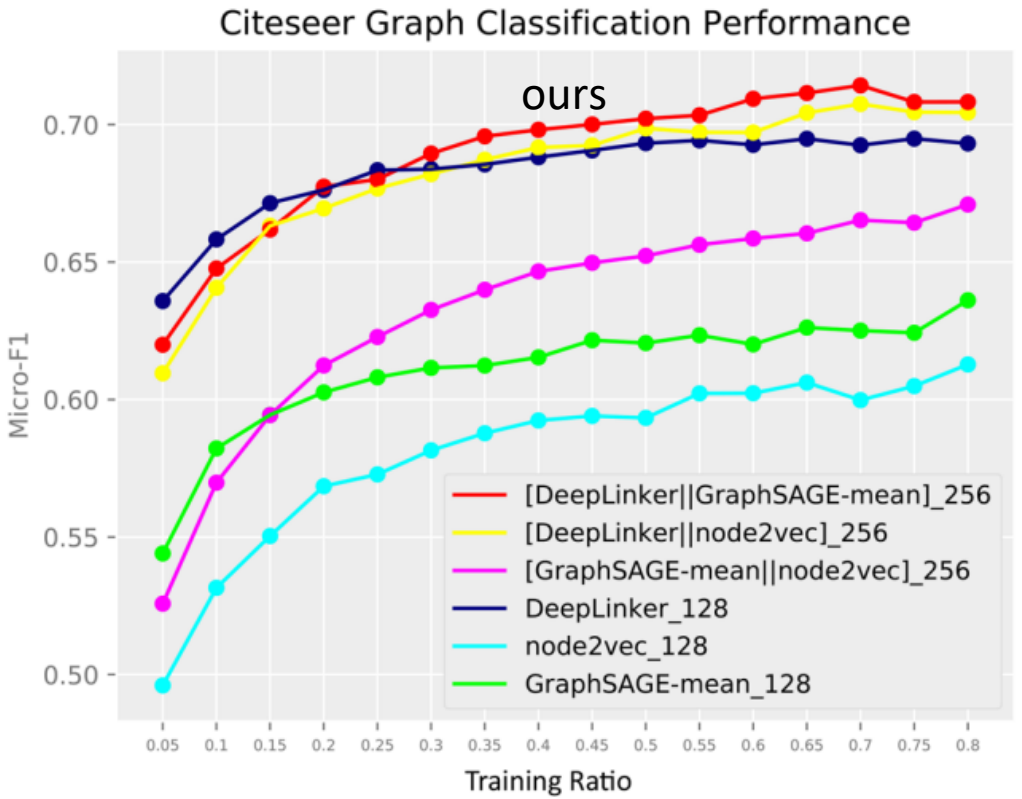


Fig. 2 Node representation evaluation under the node clustering task.

Gu, W., Gao, F., Lou, X. et al. Discovering latent node Information by graph attention network. Scientific Reports 11, 6967 (2021).

Node Clustering

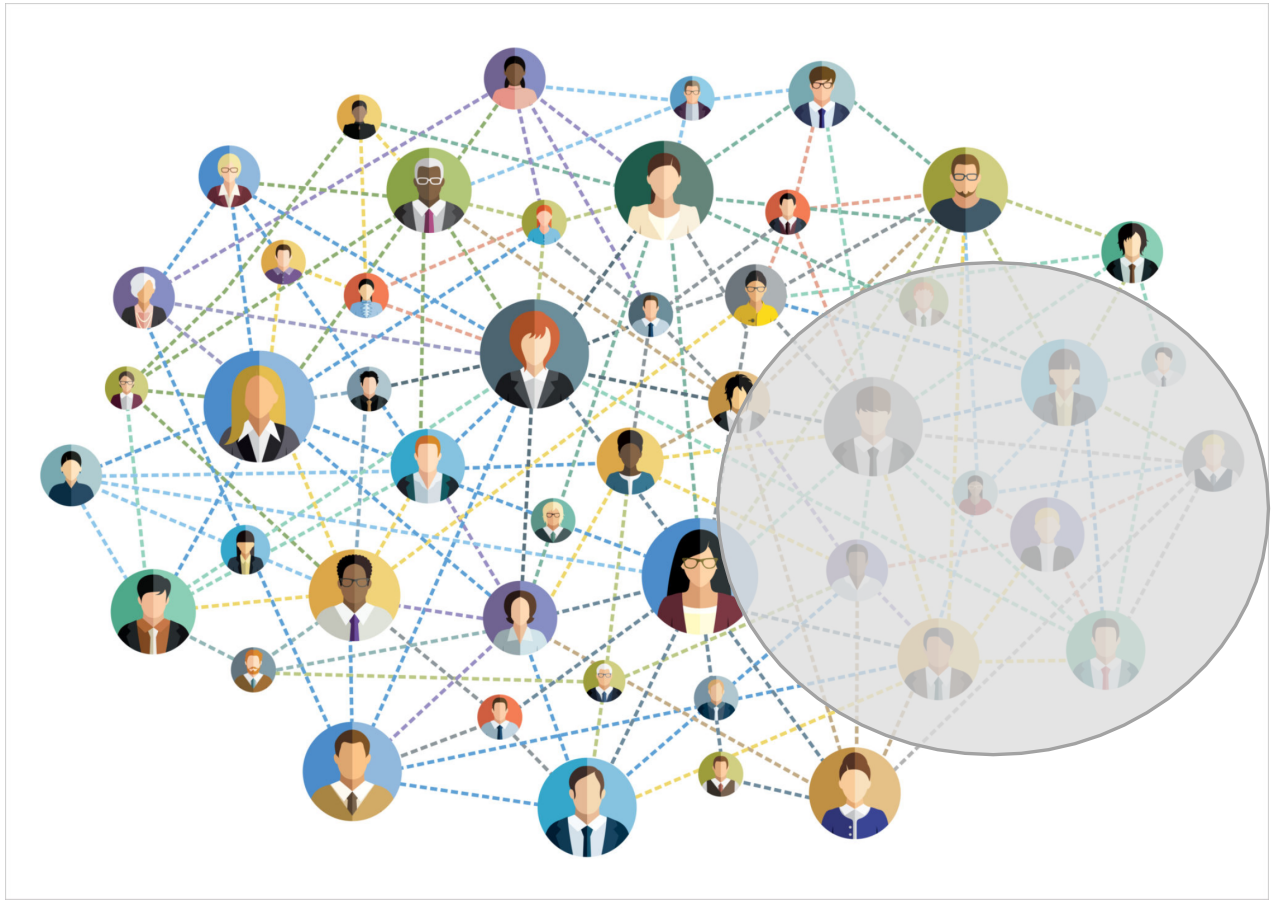




Deep Linker = ours

Gu, W., Gao, F., Lou, X. et al. Discovering latent node Information by graph attention network. Scientific Reports 11, 6967 (2021).

Network Completion



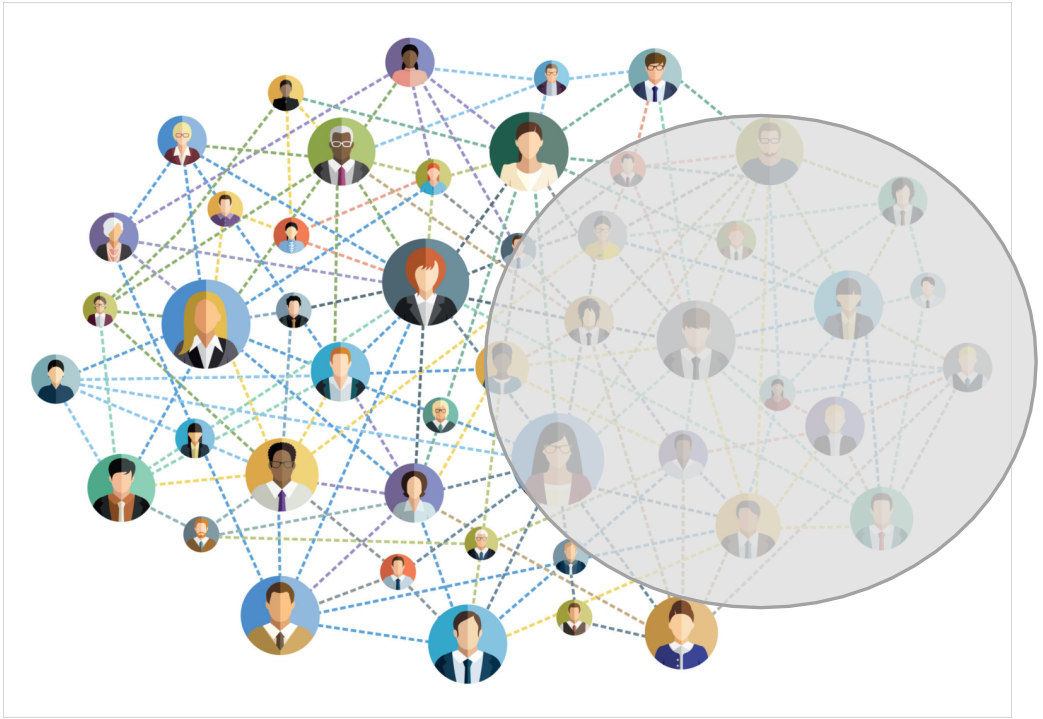
- Missing data
- Noisy data
- Incomplete information

- Social network
- Gene network
- Brain network
- Citation network

Two Kinds of Tasks

Random Missing

Block Missing



Matrix Completion v.s. Network Completion

0	1	0	?	?
1	0	1	0	?
0	1	0	1	1
?	0	1	?	?
?	?	1	?	?

(a) Low-rank matrix completion

0	1	0	?	?
1	0	1	?	?
0	1	0	?	?
?	?	?	?	?
?	?	?	?	?

(b) Network completion

https://ieeexplore.ieee.org/document/9334012

Kronecker Graphs

Kronecker graphs: An Approach to Modeling Networks

Jure Leskovec JURE@CS.STANFORD.EDU
Computer Science Department, Stanford University

Deepayan Chakrabarti DEEPAY@YAHOO-INC.COM
Yahoo! Research

Jon Kleinberg KLEINBER@CS.CORNELL.EDU
Computer Science Department, Cornell University

Christos Faloutsos CHRISTOS@CS.CMU.EDU
Computer Science Department, Carnegie Mellon University

Zoubin Ghahramani ZOUBIN@ENG.CAM.AC.UK
Department of Engineering, University of Cambridge
Machine Learning Department, Carnegie Mellon University

Abstract

How can we generate realistic networks? In addition, how can we do so with a mathematically tractable model that allows for rigorous analysis of network properties? Real networks exhibit a long list of surprising properties: Heavy tails for the in- and out-degree distribution, heavy tails for the eigenvalues and eigenvectors, small diameters, and densification and shrinking diameters over time. Current network models and generators either fail to match several of the above properties, are complicated to analyze mathematically, or both. Here we propose a generative model for networks that is both mathematically tractable and can generate networks that have all the above mentioned structural properties. Our main idea here is to use a non-standard matrix operation, the *Kronecker product*, to generate graphs which we refer to as "Kronecker graphs".

The Network Completion Problem: Inferring Missing Nodes and Edges in Networks

Myunghwan Kim* Jure Leskovec†

Abstract

While the social and information networks have become ubiquitous, the challenge of collecting complete network data still persists. Many times the collected network data is incomplete with nodes and edges missing. Commonly, only a part of the network can be observed and we would like to infer the unobserved part of the network. We address this issue by studying the *Network Completion Problem*: Given a network with missing nodes and edges, can we complete the missing part?

We cast the problem in the Expectation Maximization (EM) framework where we use the observed part of the network to fit a model of network structure, and then we estimate the missing part of the network using the model, re-estimate the parameters and so on. We combine the EM algorithm with the Kronecker graphs model and design a scalable Metropolized Gibbs sampling approach that allows for the estimation of the model parameters as well as the inference about missing nodes and edges of the network.

Experiments on synthetic and several real-world networks show that our approach can effectively recover the network even when about half of the nodes in the network are missing. Our algorithm outperforms not only classical link-prediction approaches but also the state of the art Stochastic block modeling approach. Furthermore, our algorithm easily scales to networks with tens of thousands of nodes.

complete with nodes and edges missing [16]. For example, networks arising from the popular social networking services are not completely mapped due to network boundary effects, *i.e.*, there are people who do not use the social network service so we cannot observe their connections. Similarly, when a social scientist collects the data by surveying people, he or she may not have access to certain parts of the network or people may not respond to the survey. For example, complete social networks of hidden or hard-to-reach populations, like drug injection users or sex workers and their clients, are practically impossible to collect. Thus, network data is often incomplete, which means that some nodes and edges are missing from the dataset. Analyzing such incomplete network data can significantly alter our estimates of network-level statistics and inferences about structural properties of the underlying network [16].

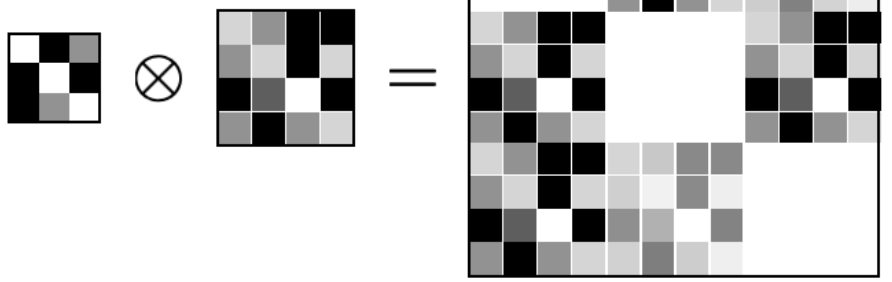
We systematically investigate the *Network Completion Problem* where we aim to infer the unobserved part of the network. While the problem of missing link inference [7, 11] has been thoroughly studied before, we consider a problem where both edges and nodes are missing. Thus, the question that we address here is: given an incomplete network with missing nodes and edges, can we fill in the missing parts? For example, can we estimate the structure of the full Facebook social network by only observing/collecting a part of it? Figure 1(a) illustrates the task. There is a complete network represented by

https://arxiv.org/pdf/0812.4905.pdf

Kronecker Graphs

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \dots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \dots & a_{mn}\mathbf{B} \end{bmatrix}$$

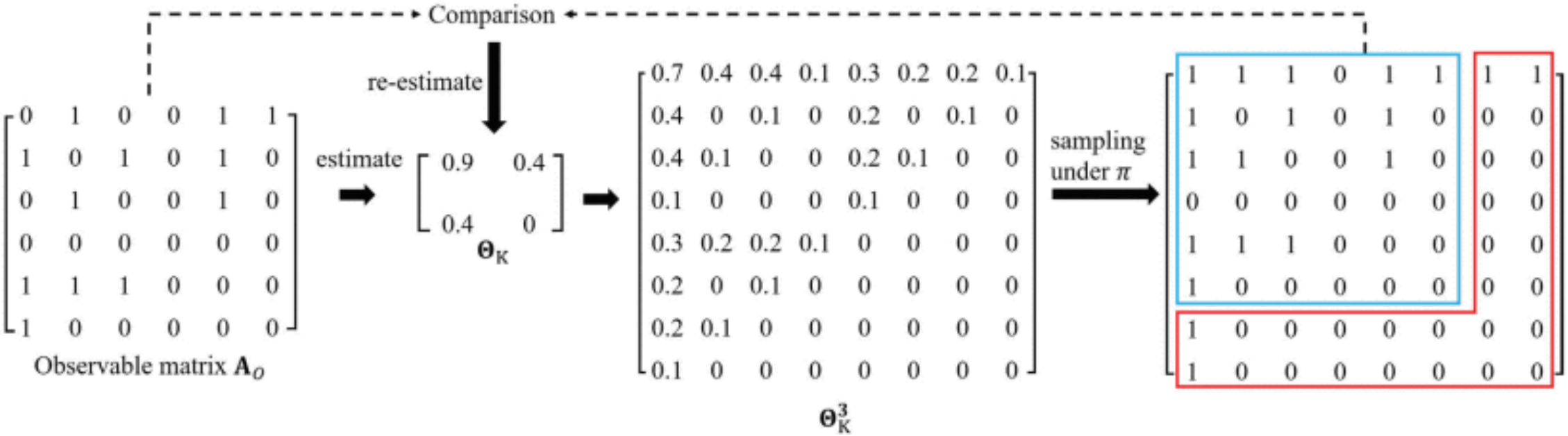
$$\mathbf{A} \otimes \mathbf{B} := \begin{pmatrix} [\mathbf{A}]_{1,1}\mathbf{B} & \cdots & [\mathbf{A}]_{1,n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ [\mathbf{A}]_{m,1}\mathbf{B} & \cdots & [\mathbf{A}]_{m,n}\mathbf{B} \end{pmatrix} \in \mathbb{R}^{ma \times nb}$$



$\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{B} \in \mathbb{R}^{a \times b}$: Kronecker factors

P. M. Weichsel, "The kronecker product of graphs", *J. Am. Math. Soc*, vol. 13, no. 1, pp. 47-52, 1962.

Kronecker Graphs



M. Kim and J. Leskovec, "The network completion problem: Inferring missing nodes and edges in networks", *Proc. 2011 SIAM Int. Conf. Data Mining (SDM '11)*, pp. 47-58, Apr. 2011.