Inductive Representation Learning on Large Graphs

William L. Hamilton* wleif@stanford.edu
Rex Ying* rexying@stanford.edu
Jure Leskovec jure@cs.stanford.edu

Department of Computer Science
Stanford University
Stanford, CA, 94305

NIPS 2017

Citations : 1139

speaker : 刘闯
supervisor :
mail : chuangliu@whu.edu.cn

Introduction of Author

Jure Leskovec

已关注

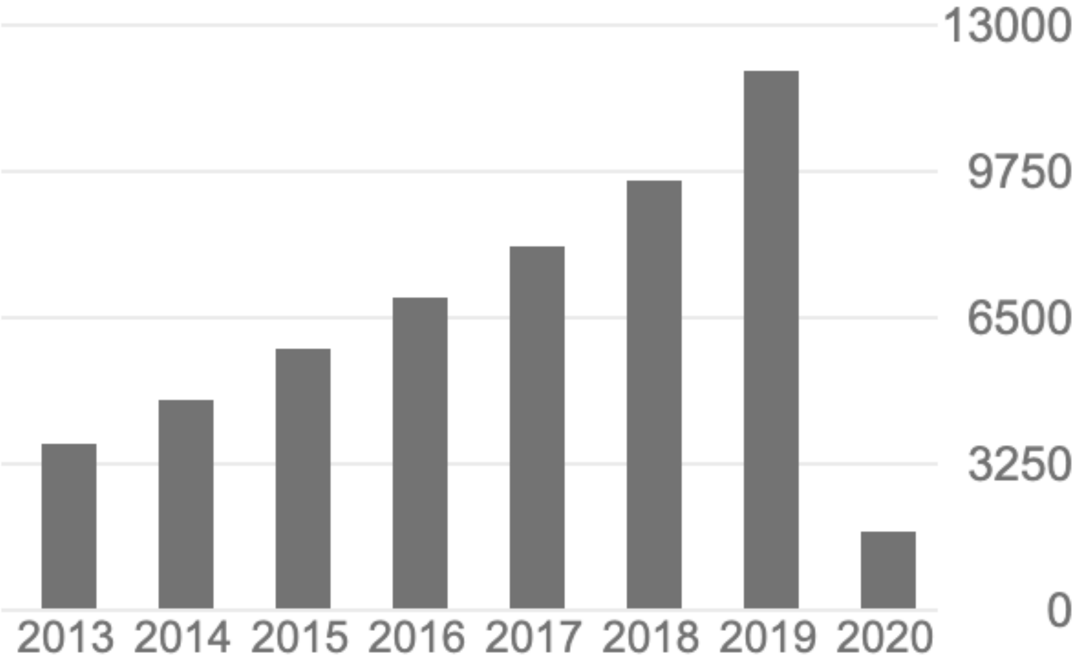
Professor of Computer Science, Stanford University
在 cs.stanford.edu 的电子邮件经过验证 - 首页

Data mining   Machine Learning   Graph Neural Networks   Knowledge Graphs   Complex Networks

创建我的个人资料

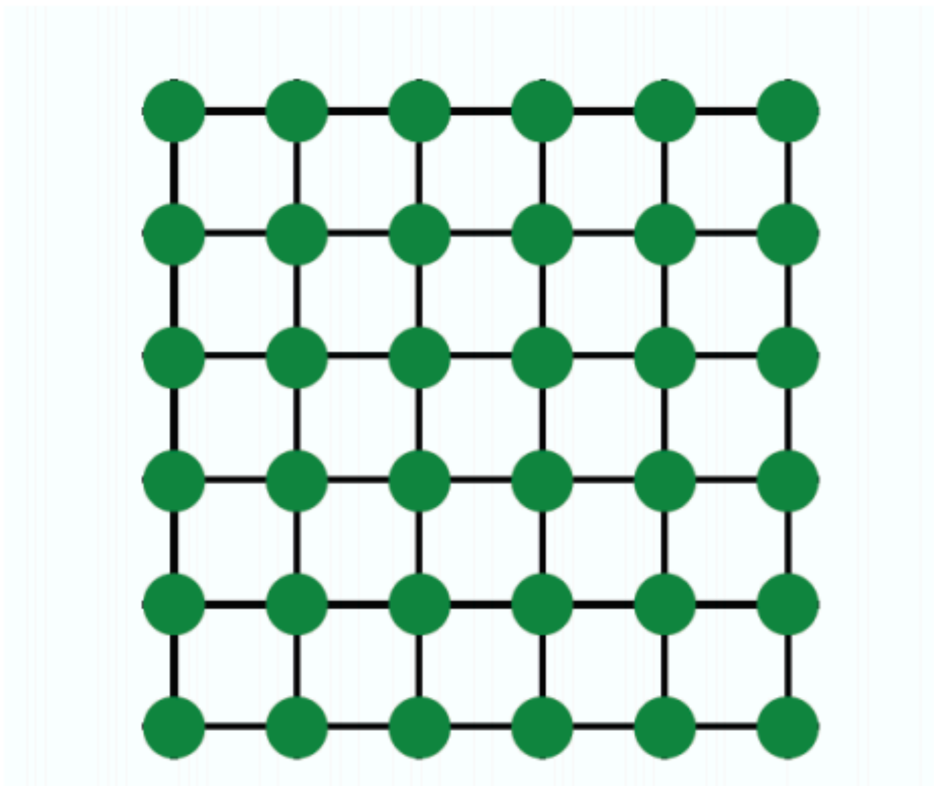
引用次数   查看全部

	总计	2015 年至今
引用	61303	44315
h 指数	103	92
i10 指数	213	200

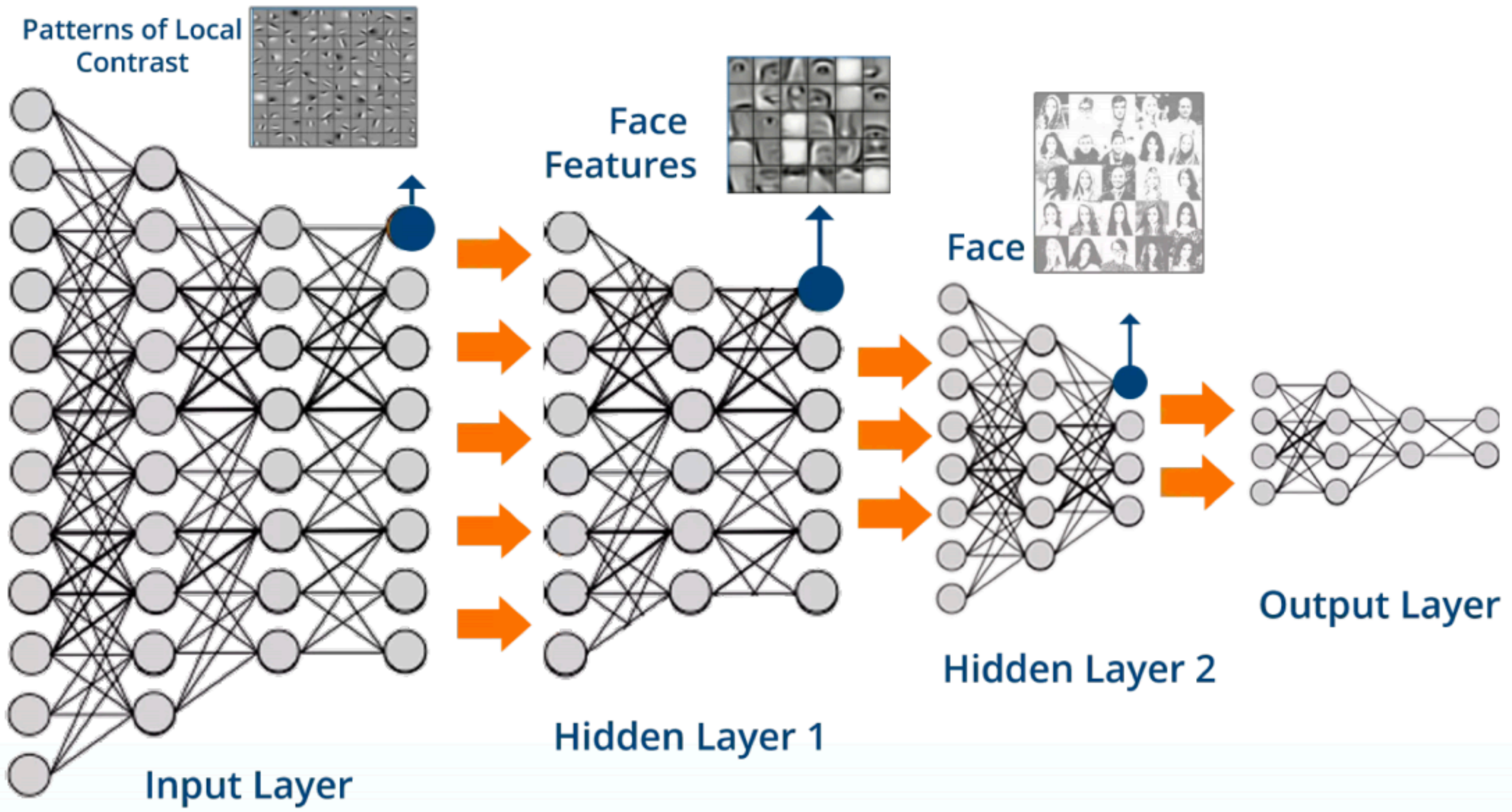


标题	引用次数	年份
node2vec: Scalable feature learning for networks A Grover, J Leskovec Proceedings of the 22nd ACM SIGKDD international conference on Knowledge …	2660	2016
Graphs over time: densification laws, shrinking diameters and possible explanations J Leskovec, J Kleinberg, C Faloutsos Proceedings of the eleventh ACM SIGKDD international conference on Knowledge …	2379	2005
The dynamics of viral marketing J Leskovec, LA Adamic, BA Huberman ACM Transactions on the Web (TWEB) 1 (1), 5-es	2374	2007
Friendship and mobility: user movement in location-based social networks E Cho, SA Myers, J Leskovec	2328	2011

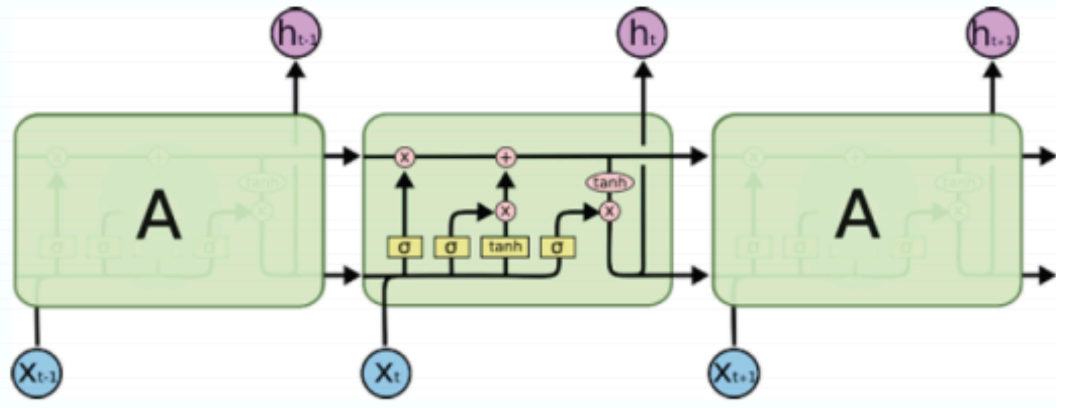
Modern Deep Learning



Images



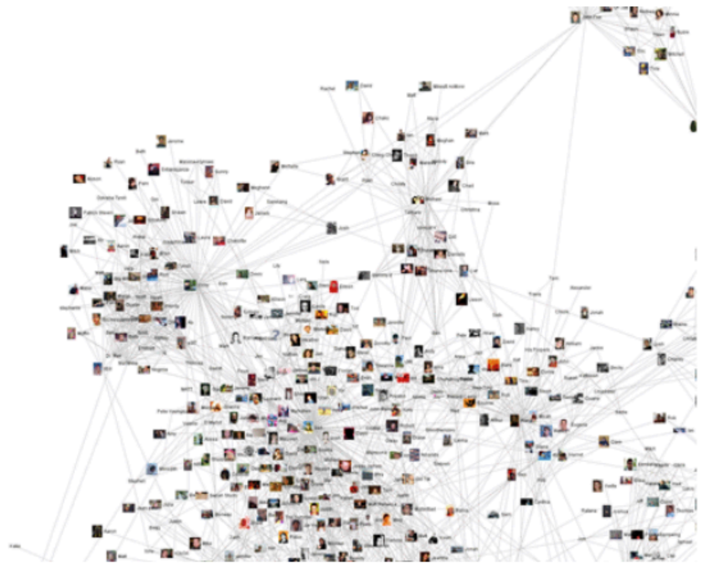
Text/Speech



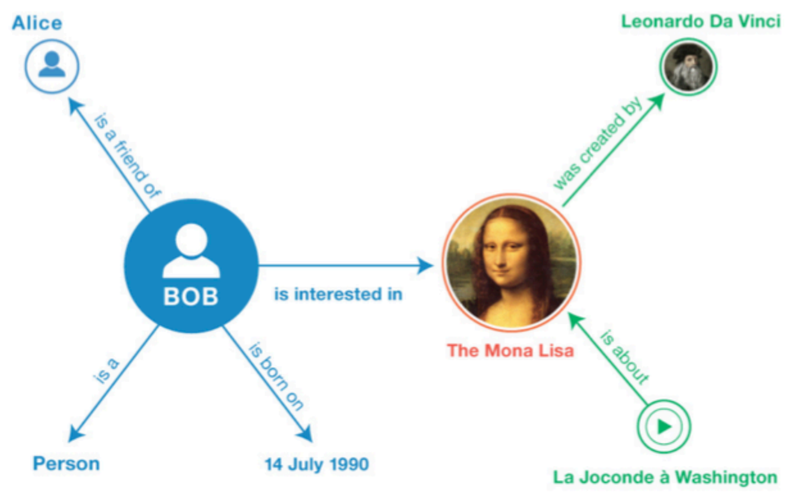
Modern deep learning toolbox is designed for simple sequences & grids

Networks

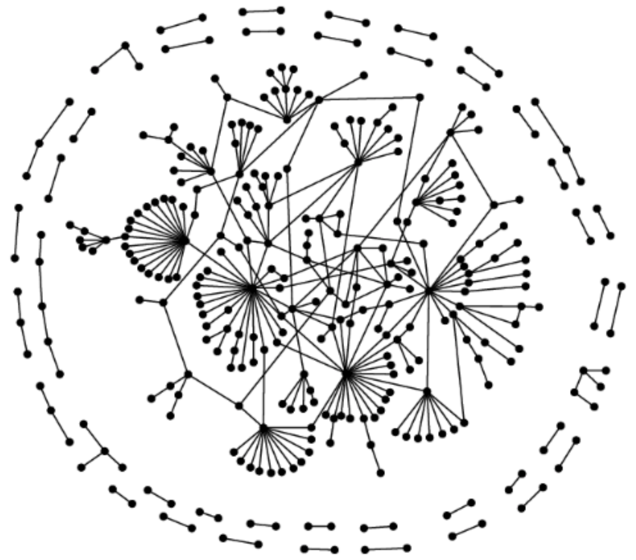
Not everything can be represented as a sequence or a grid



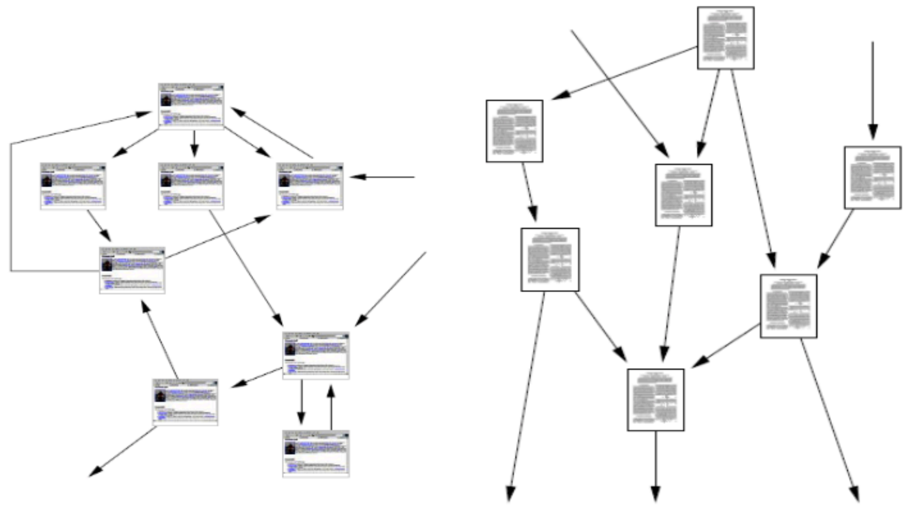
Social networks



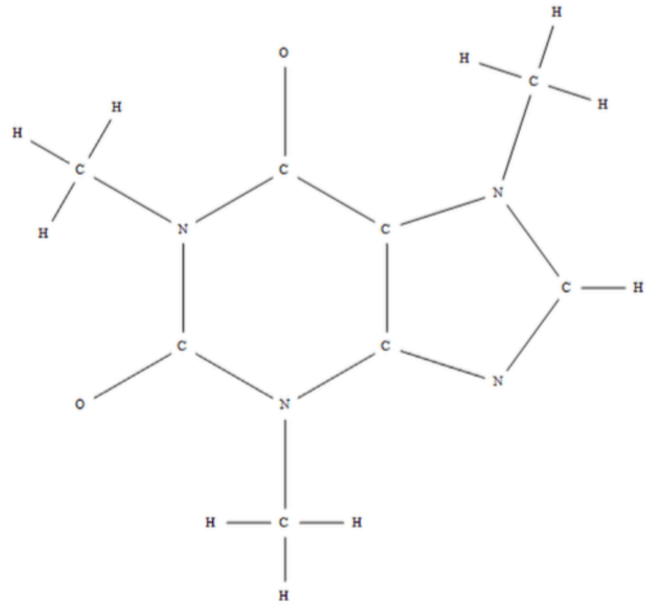
Knowledge graphs



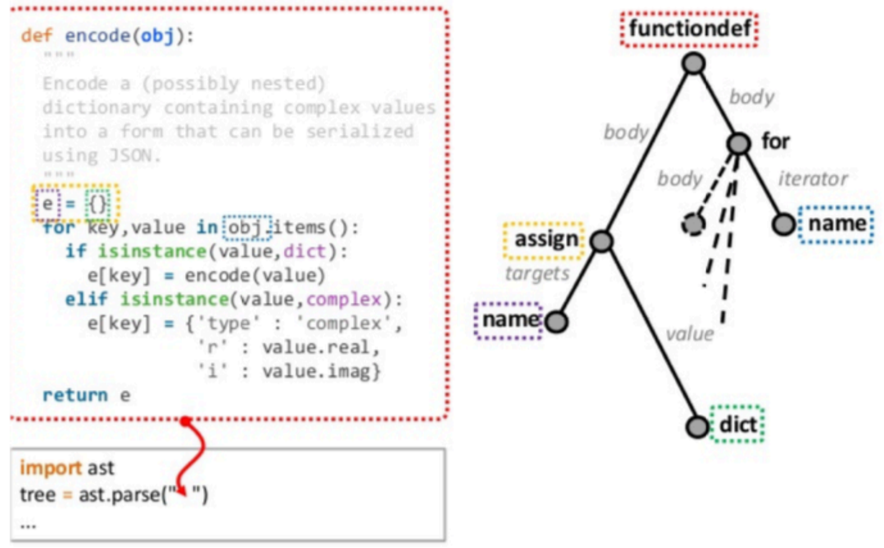
Biological networks



Complex Systems



Molecules

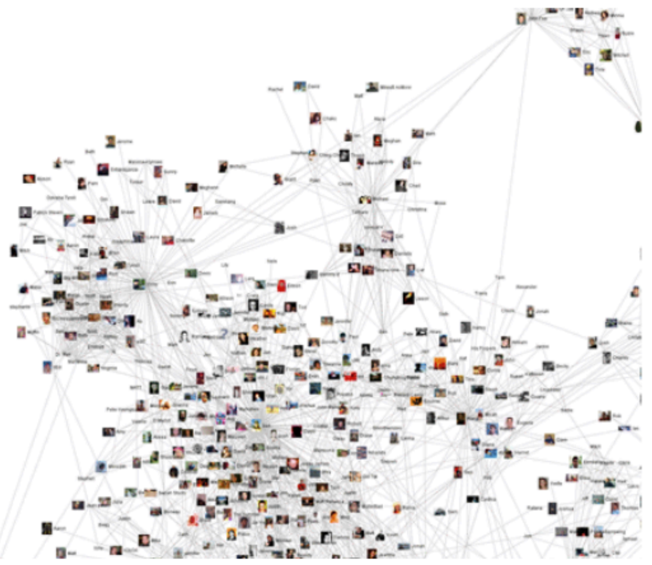


Code

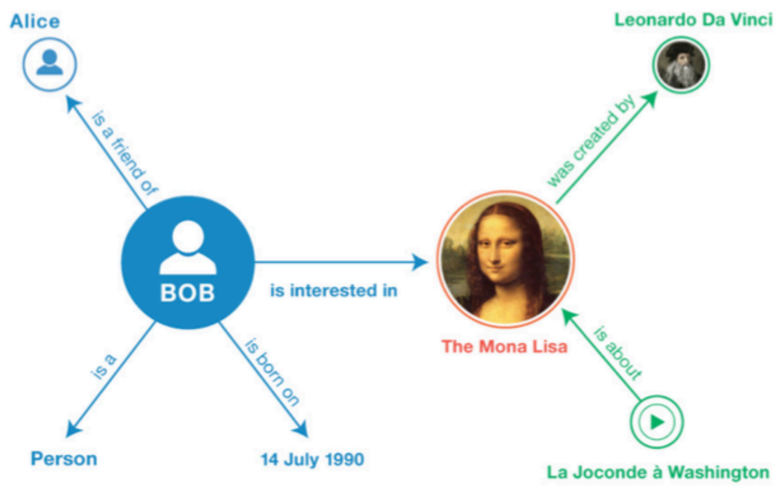
Task

- Link prediction
- Node classification
- Community detection

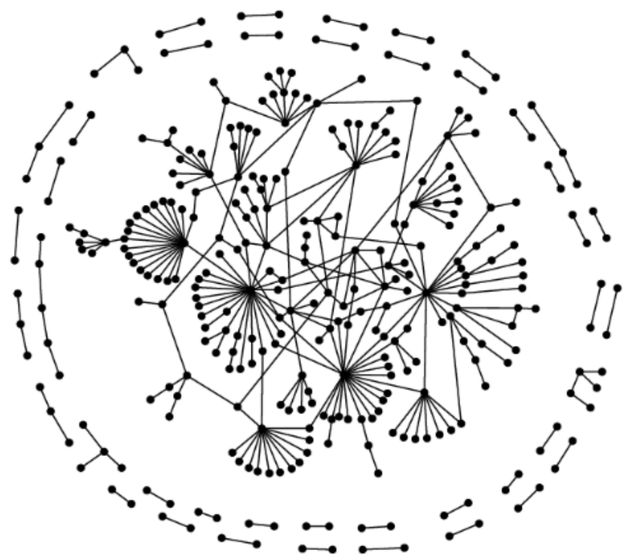
Networks : Graph



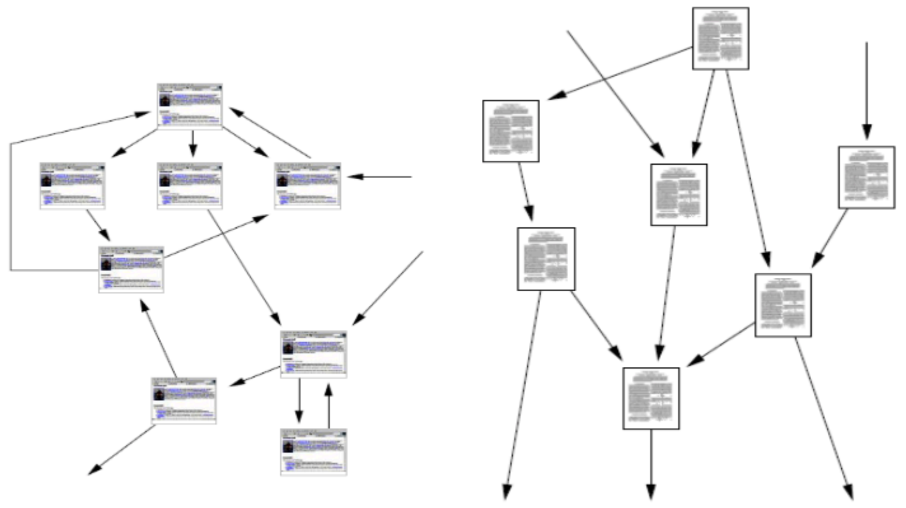
Social networks



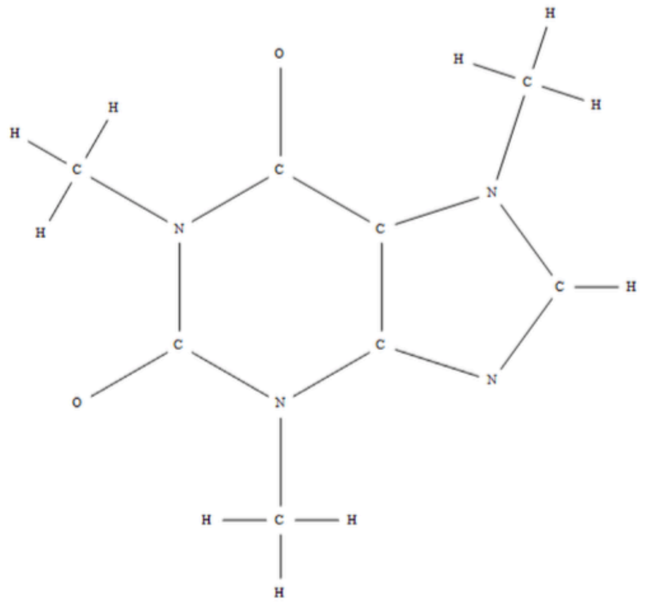
Knowledge graphs



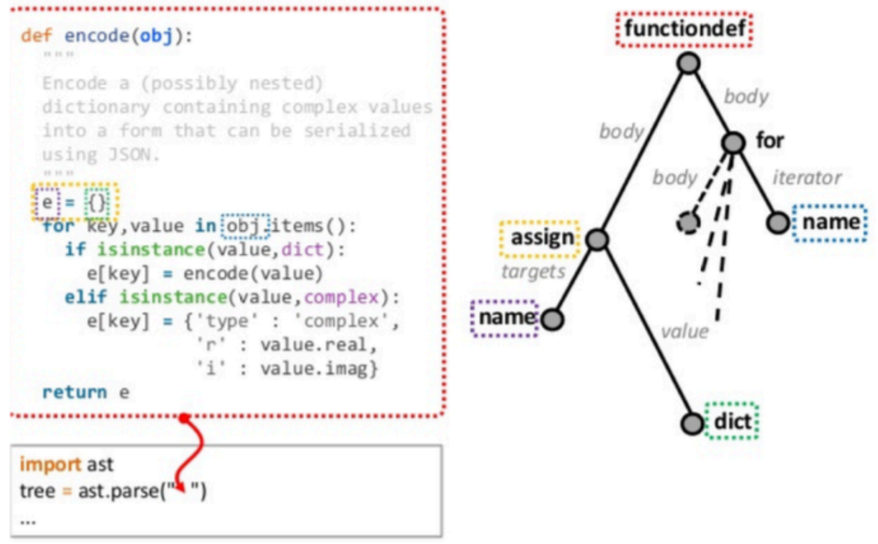
Biological networks



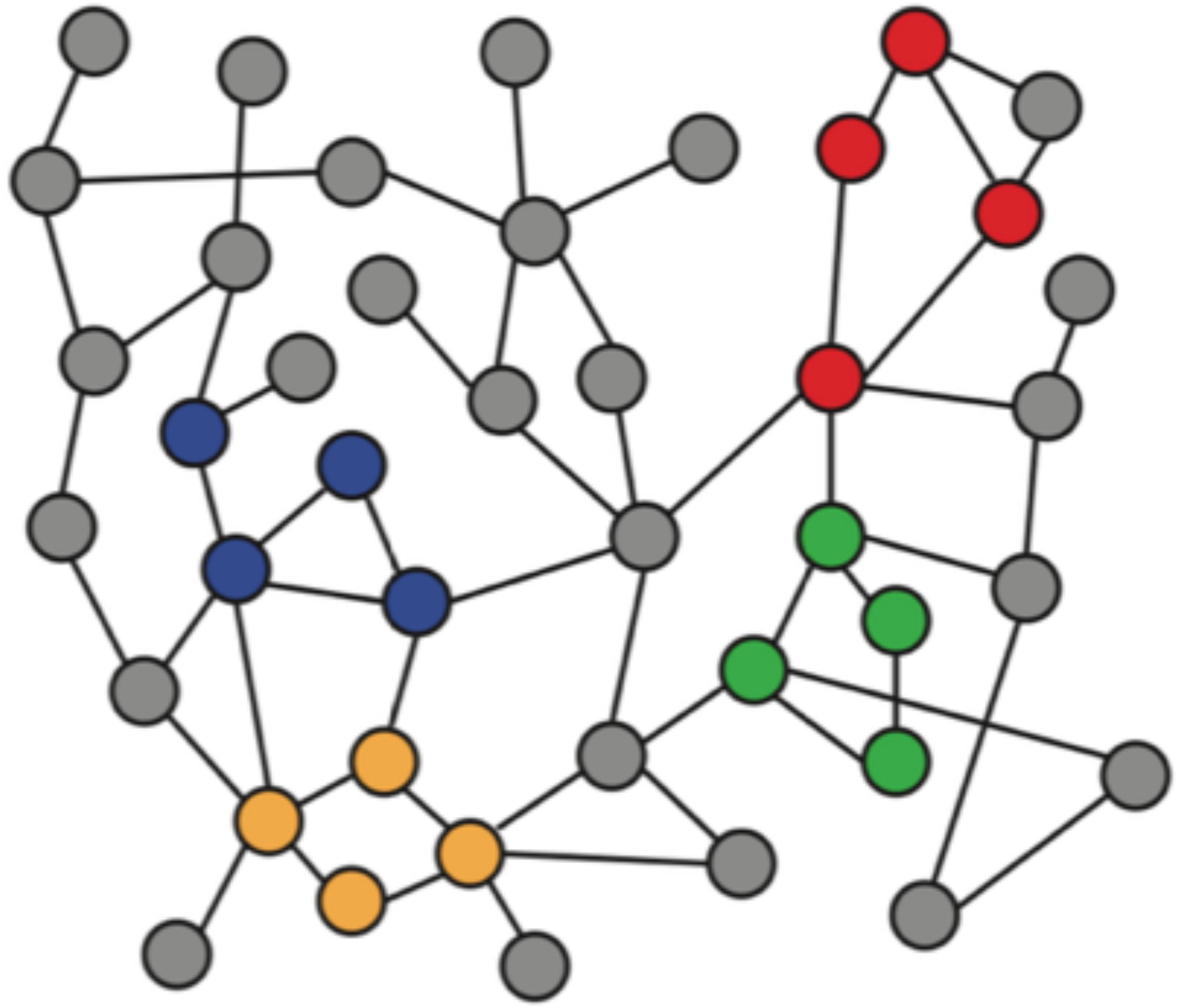
Complex Systems



Molecules



Code

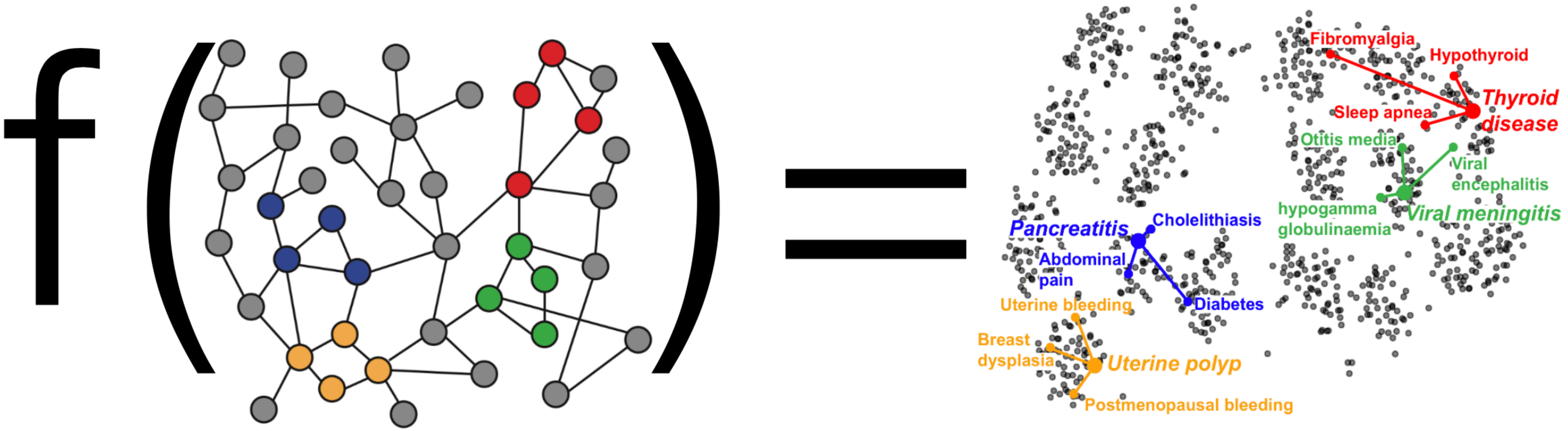


Abstract

Networks → Graph

Graph(node) Embedding

Map nodes to *d*-dimensional embeddings

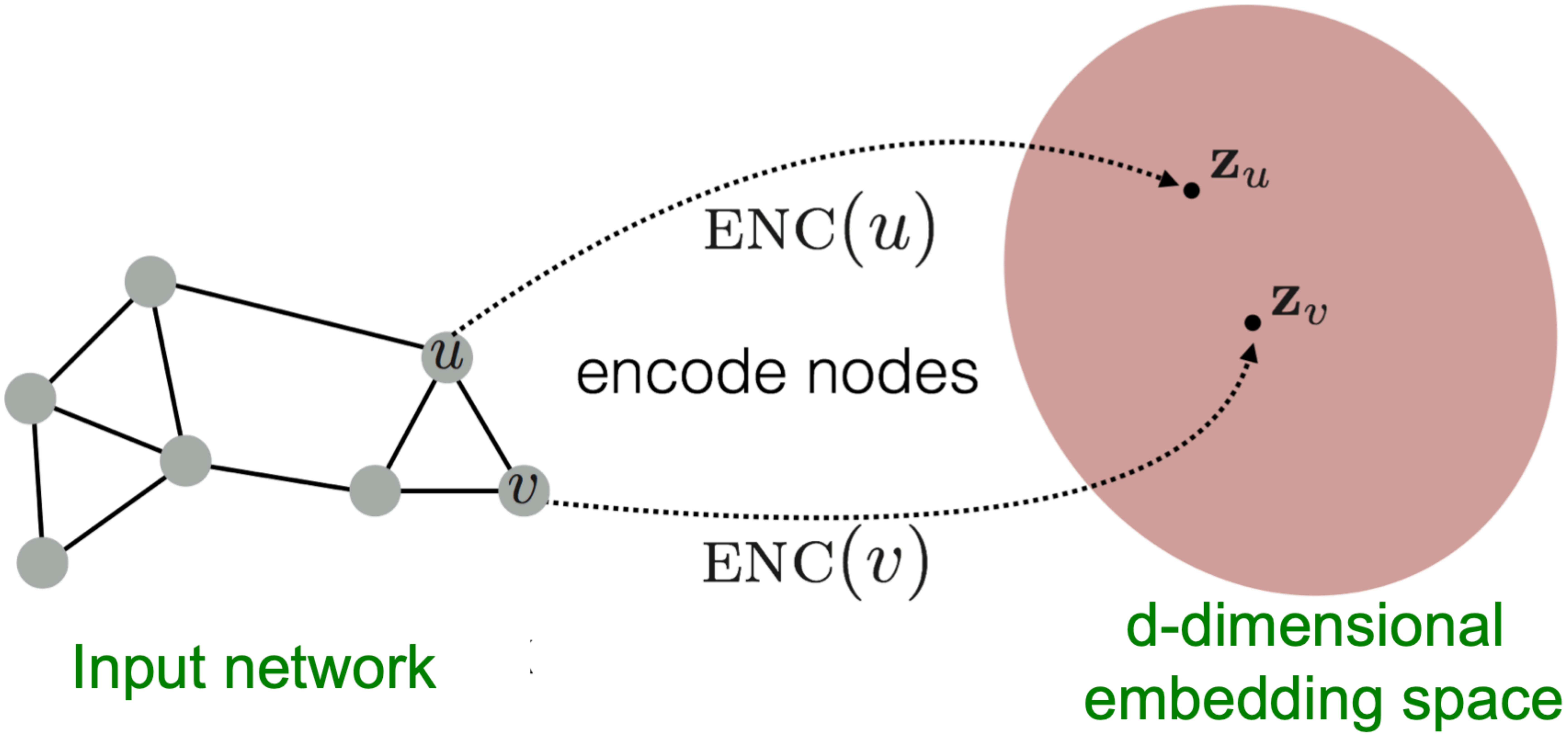


Input graph

2D node embeddings

How to learn mapping function f

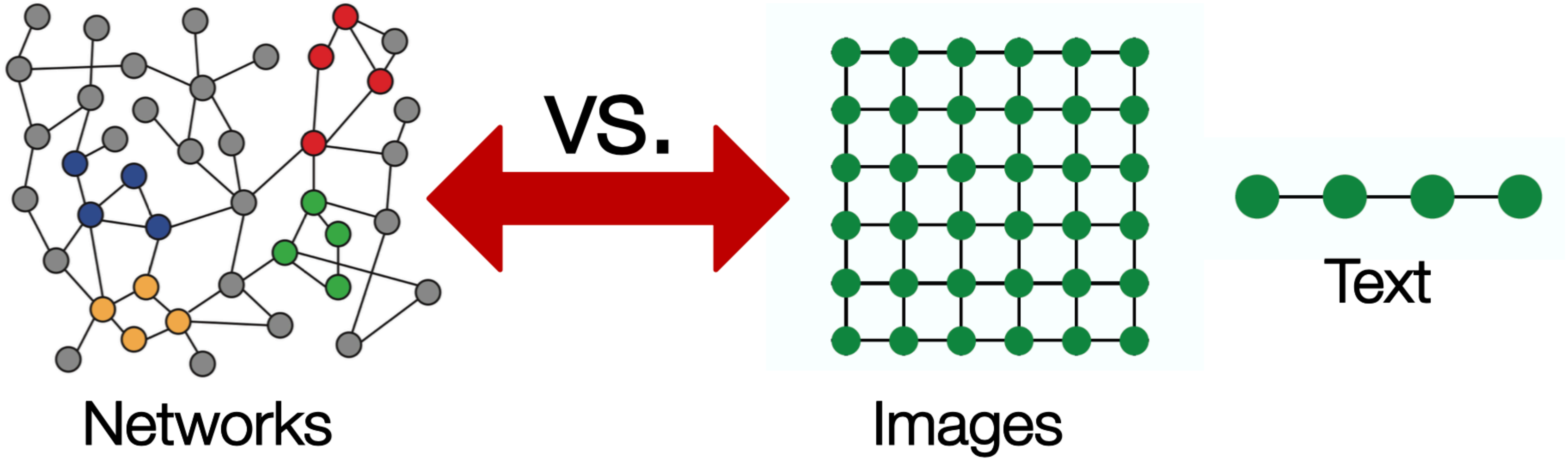
Embedding Rule



$$\textbf{similarity}\ (u, v) \approx \mathbf{z}_v^\top \mathbf{z}_u$$

In the network

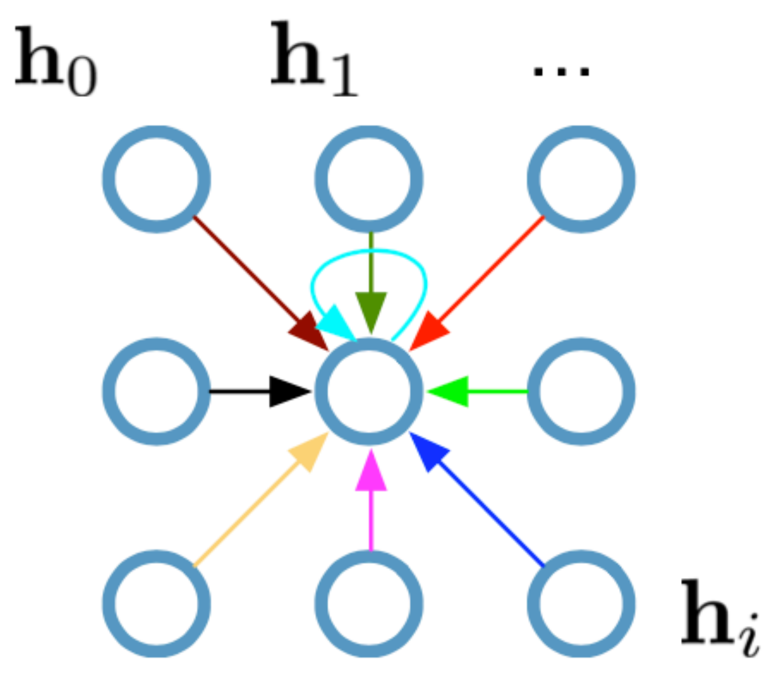
Hard : Networks are complex !



- 1. Arbitrary size and complex topological structure (no spatial locality)
- 2. No fixed node ordering or reference point
- 3. Often dynamic and have multimodal features

Idea : Convolutional Networks

Generalize convolutions from lattices to graph

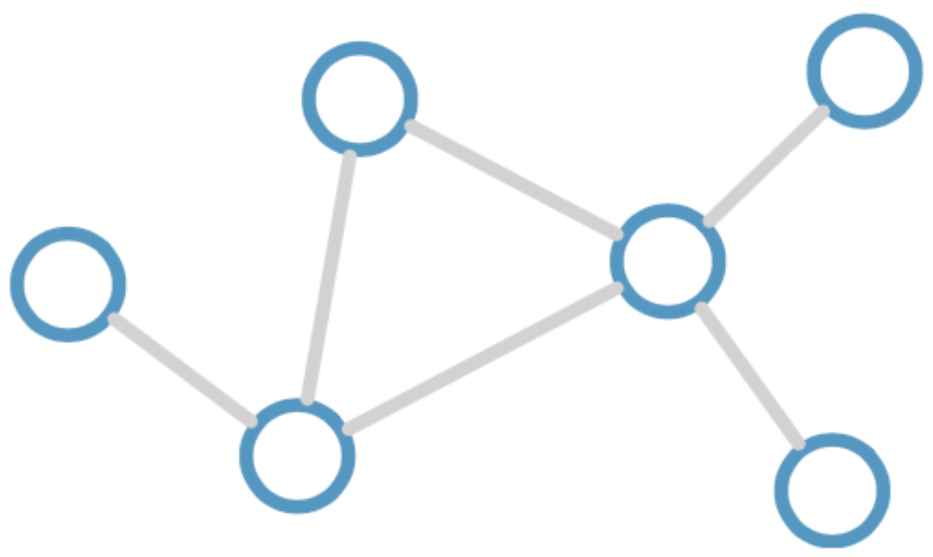


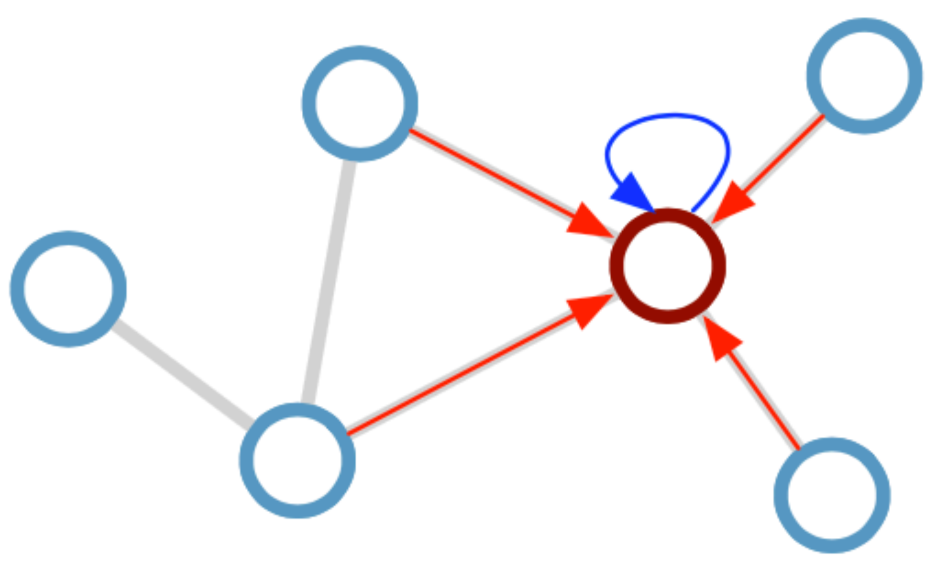
Update for a single pixel:

- Transform messages individually $\mathbf{W}_i \mathbf{h}_i$
- Add everything up $\sum_i \mathbf{W}_i \mathbf{h}_i$

Full update:

$$\mathbf{h}_4^{(l+1)} = \sigma\left(\mathbf{W}_0^{(l)}\mathbf{h}_0^{(l)} + \mathbf{W}_1^{(l)}\mathbf{h}_1^{(l)} + \cdots + \mathbf{W}_8^{(l)}\mathbf{h}_8^{(l)}\right)$$



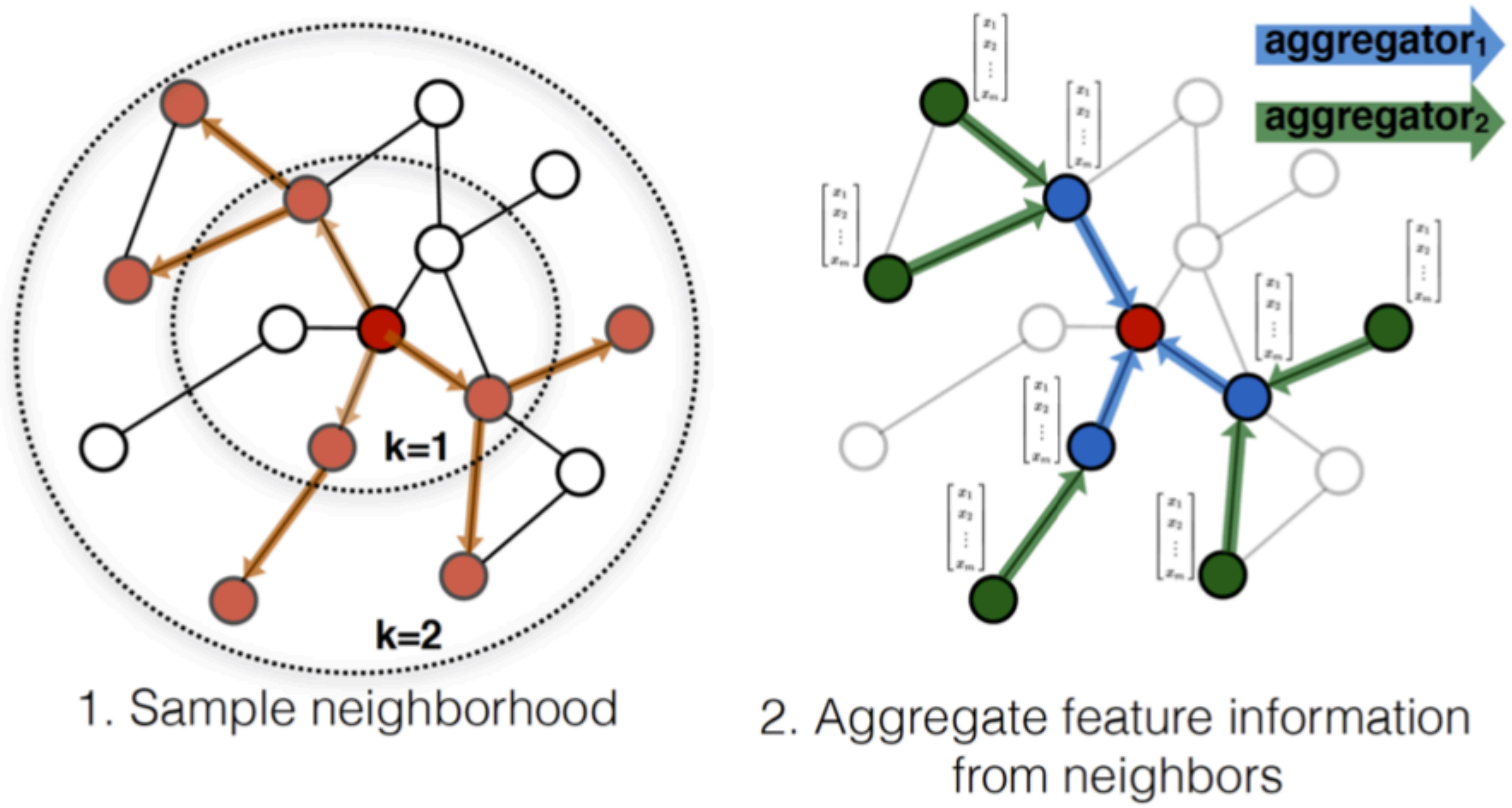


Update rule:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} \mathbf{h}_j^{(l)} \mathbf{W}_1^{(l)}\right)$$

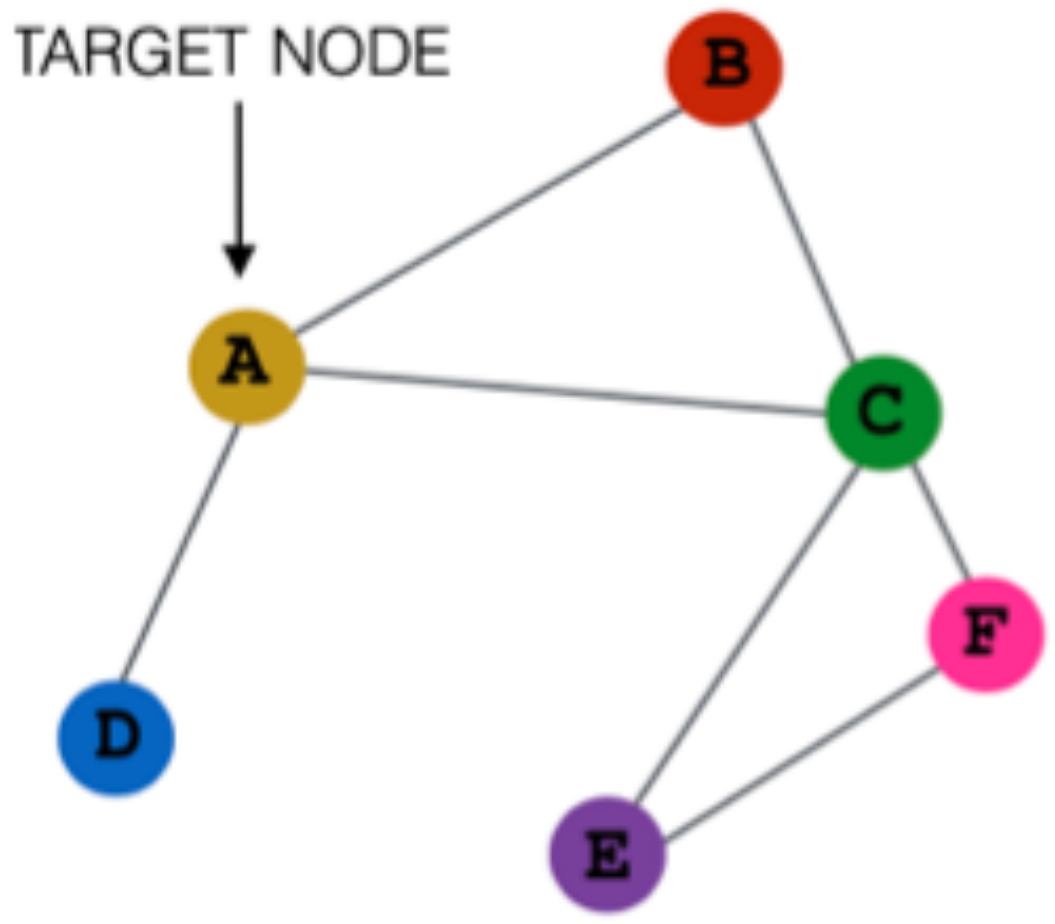
Approach : GraphSAGE

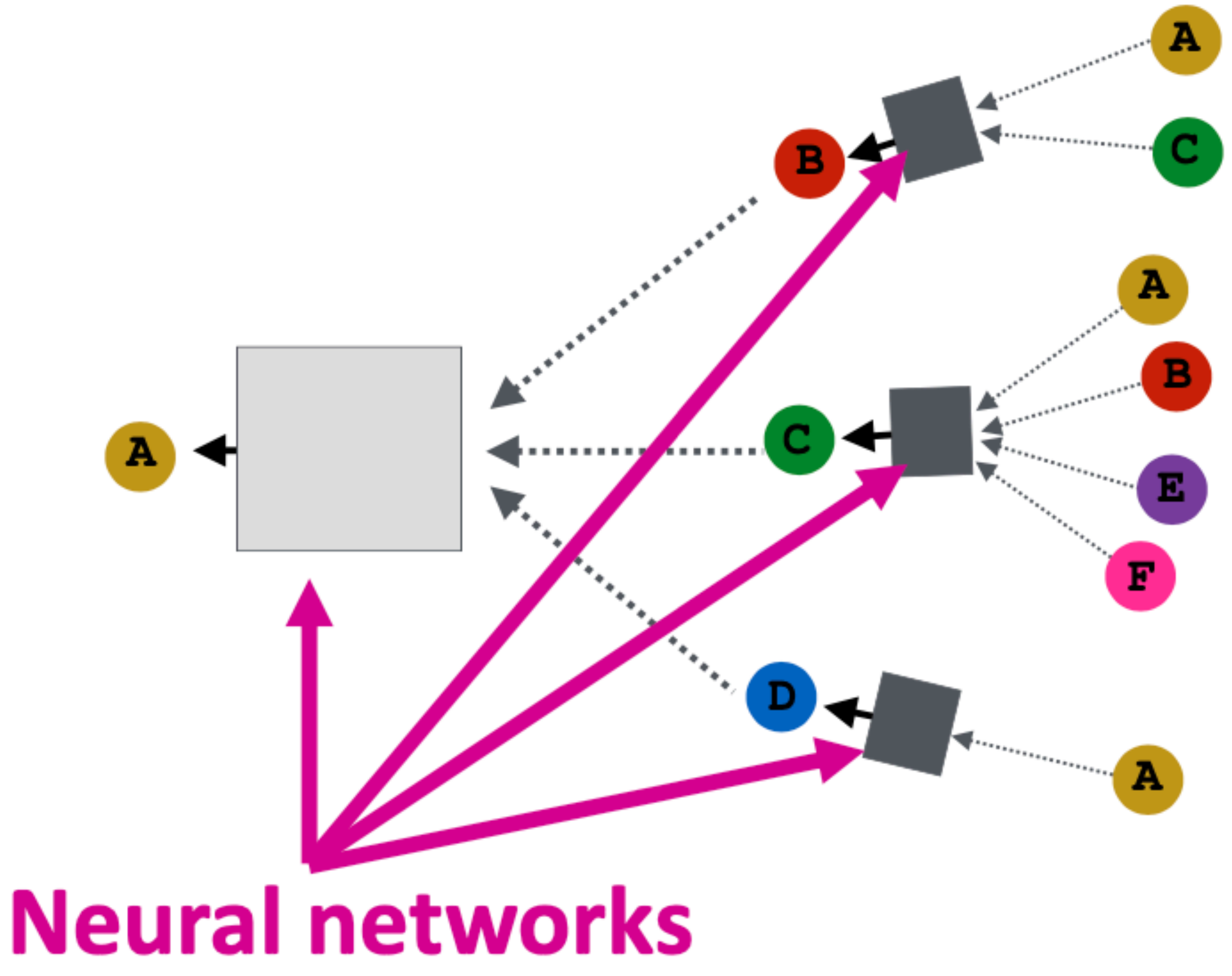
GraphSAGE = **Graph** + **SA**mple + aggre**G**at**E**

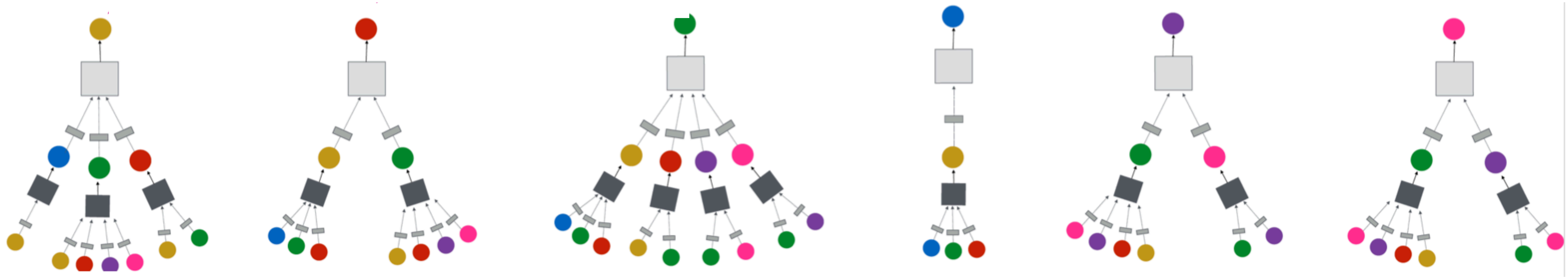


Generate node embeddings based on local network neighborhoods

GraphSAGE: Aggregate







Layer K = 2

GraphSAGE : Aggregator Architectures

Invariant to permutations of its input(no nature **ordering**)

- **Mean**

$$\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{MEAN}\left(\{\mathbf{h}_v^{k-1}\} \cup \{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\}\right)\right)$$

- **LSTM**

Adapt LSTMs to operate on an unordered set by simply applying the LSTMs to a random permutation of the node's neighbor

- **Pooling**

$$\text{AGREGATE}_k^{\text{pool}} = \max\left(\left\{\sigma\left(\mathbf{W}_{\text{pool}}\mathbf{h}_{u_i}^k + \mathbf{b}\right), \forall u_i \in \mathcal{N}(v)\right\}\right)$$

GraphSAGE : Parameters

$$\mathbf{h}_v^k = \sigma\left(\left[\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|}, \mathbf{B}_k \mathbf{h}_v^{k-1}\right]\right), \forall k \in \{1, \ldots, K\} \tag{1}$$

Weight matrices and parameters of the aggregator functions

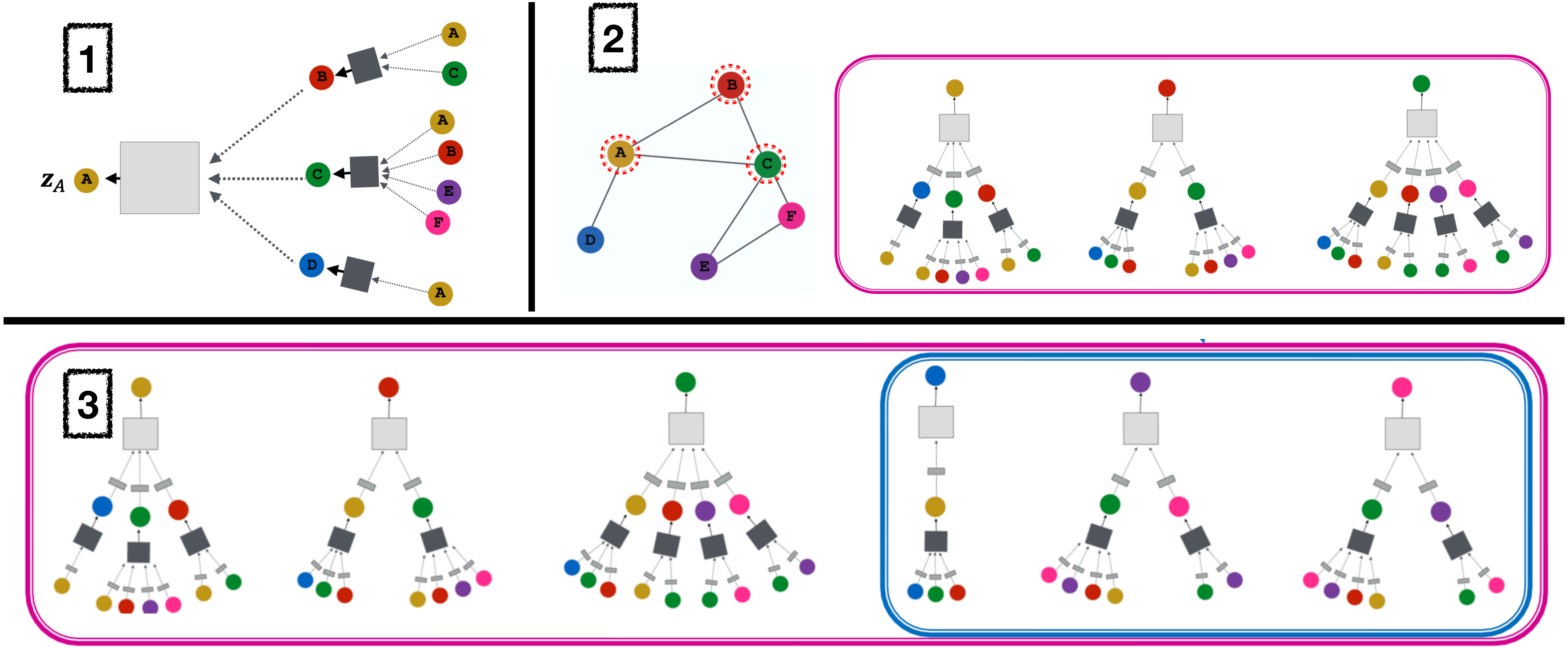
Unsupervised

$$J_{\mathcal{G}}\left(\mathbf{z}_u\right) = -\log\left(\sigma\left(\mathbf{z}_u^\top \mathbf{z}_v\right)\right) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log\left(\sigma\left(-\mathbf{z}_u^\top \mathbf{z}_{v_n}\right)\right) \tag{2}$$

Supervised

$$\mathscr{L}\left(h_A\right) = ||y_A - f\left(h_A\right)||^2 \tag{3}$$

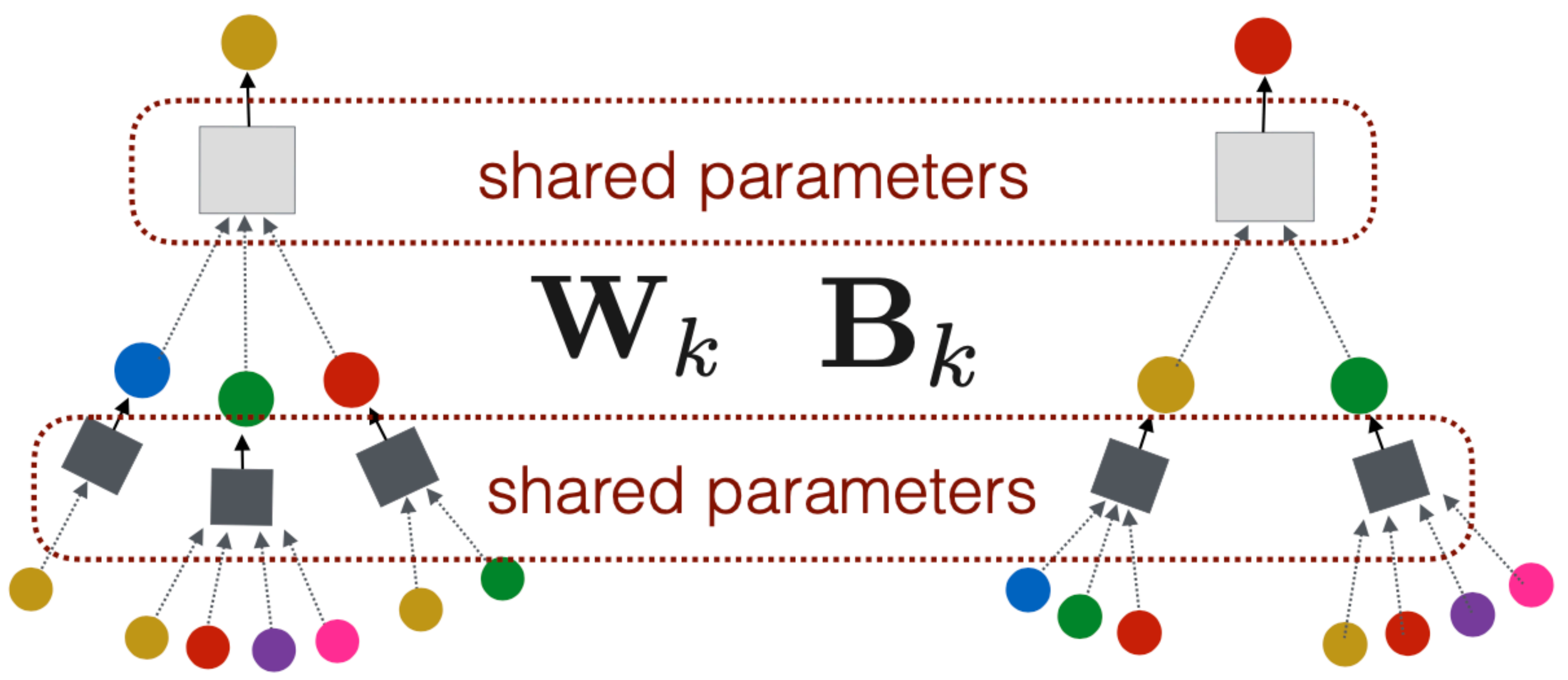
GraphSAGE : Overview



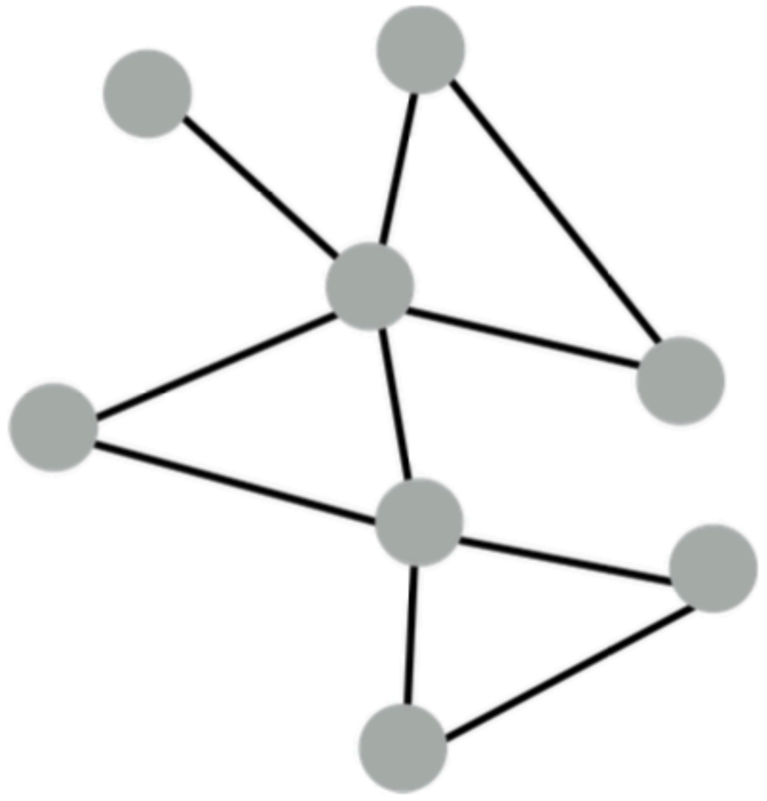
1. *Define a neighborhood aggregation function and a loss function on the embeddings*
2. *Train on a set of nodes, i.e., a batch of compute graphs*
3. *Generate embeddings for nodes as needed(Even for nodes we never trained on)*

GraphSAGE : Inductive

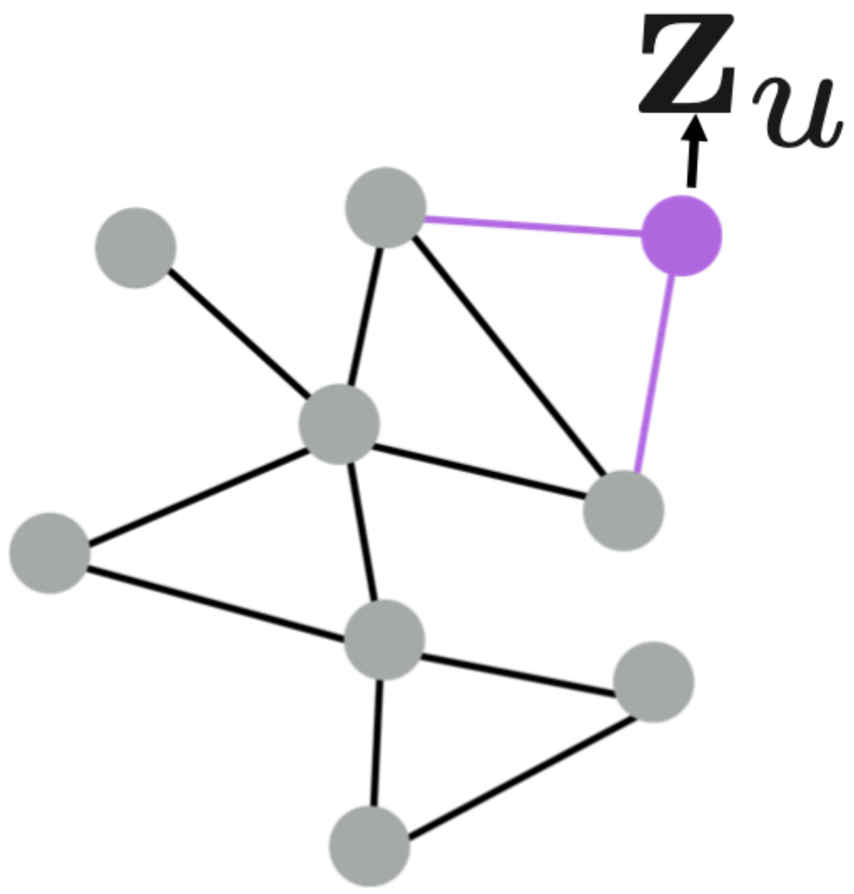
The same aggregation parameters
are shared for all nodes



New node

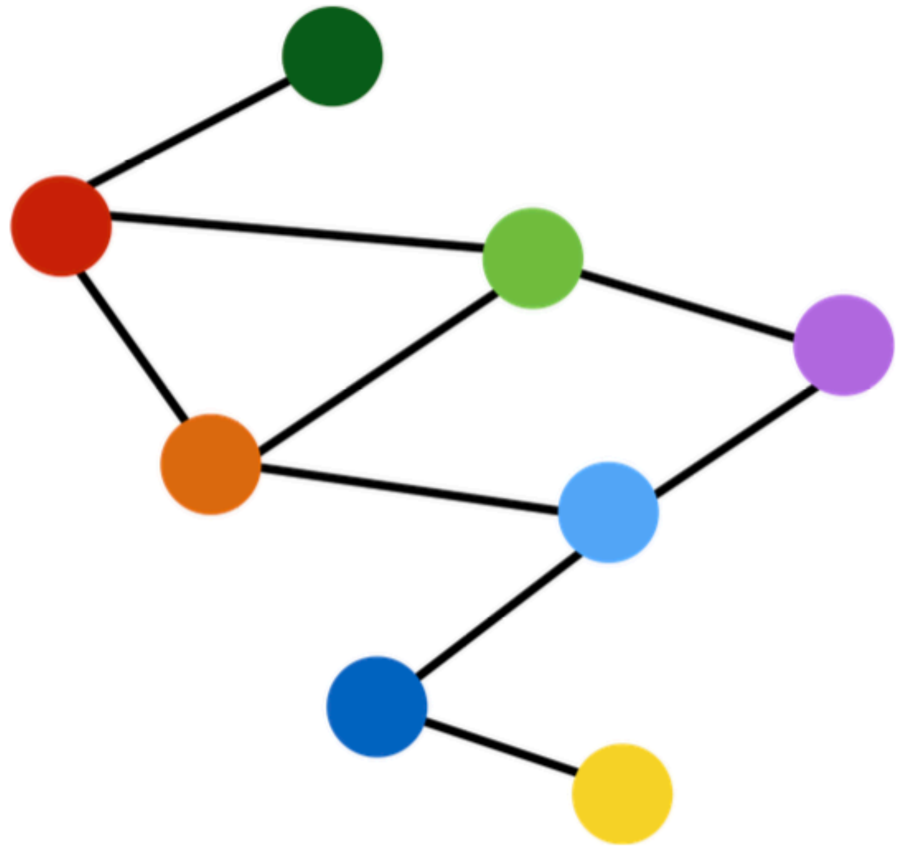


Train graph

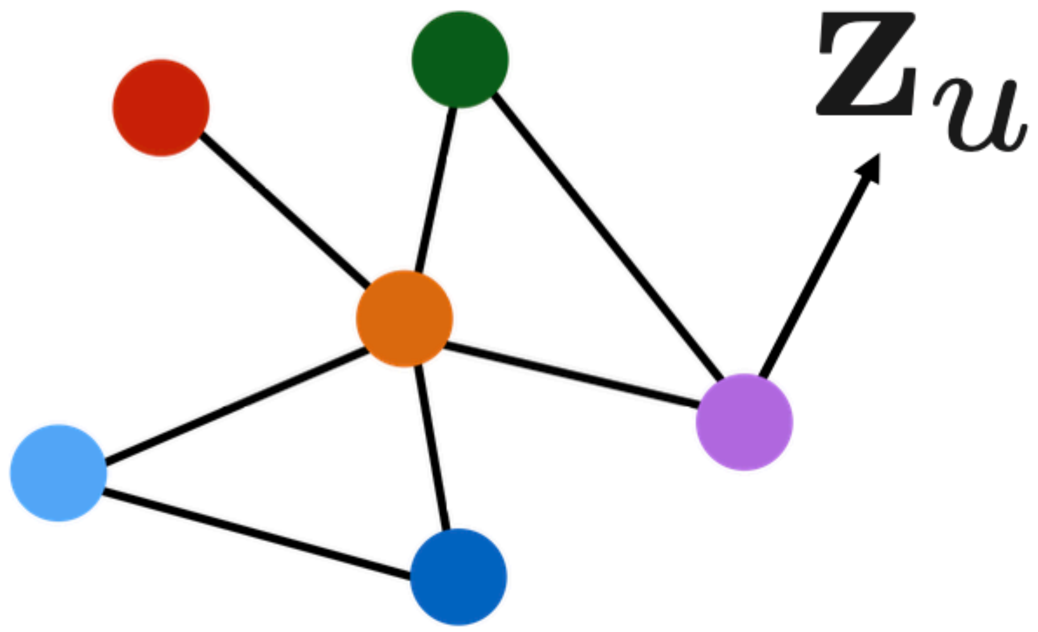


Generate embedding
for new node

New graph



Train graph



Generalize to new graph

Experiments

Set-up :

- Test the performance on **node-classification task.**
- Evaluation on three datasets : Citation, Reddit, PPI.
- For all the GraphSAGE, set $K = 2$,with neighborhood **sample sizes** $S_1 = 25$ and $S_2 = 10$.

Baselines :

- **Random :** a random classifier.
- **Logistic:** a logistic regression feature-based classifier.
- **DeepWalk :** a representative factorization-based approach.

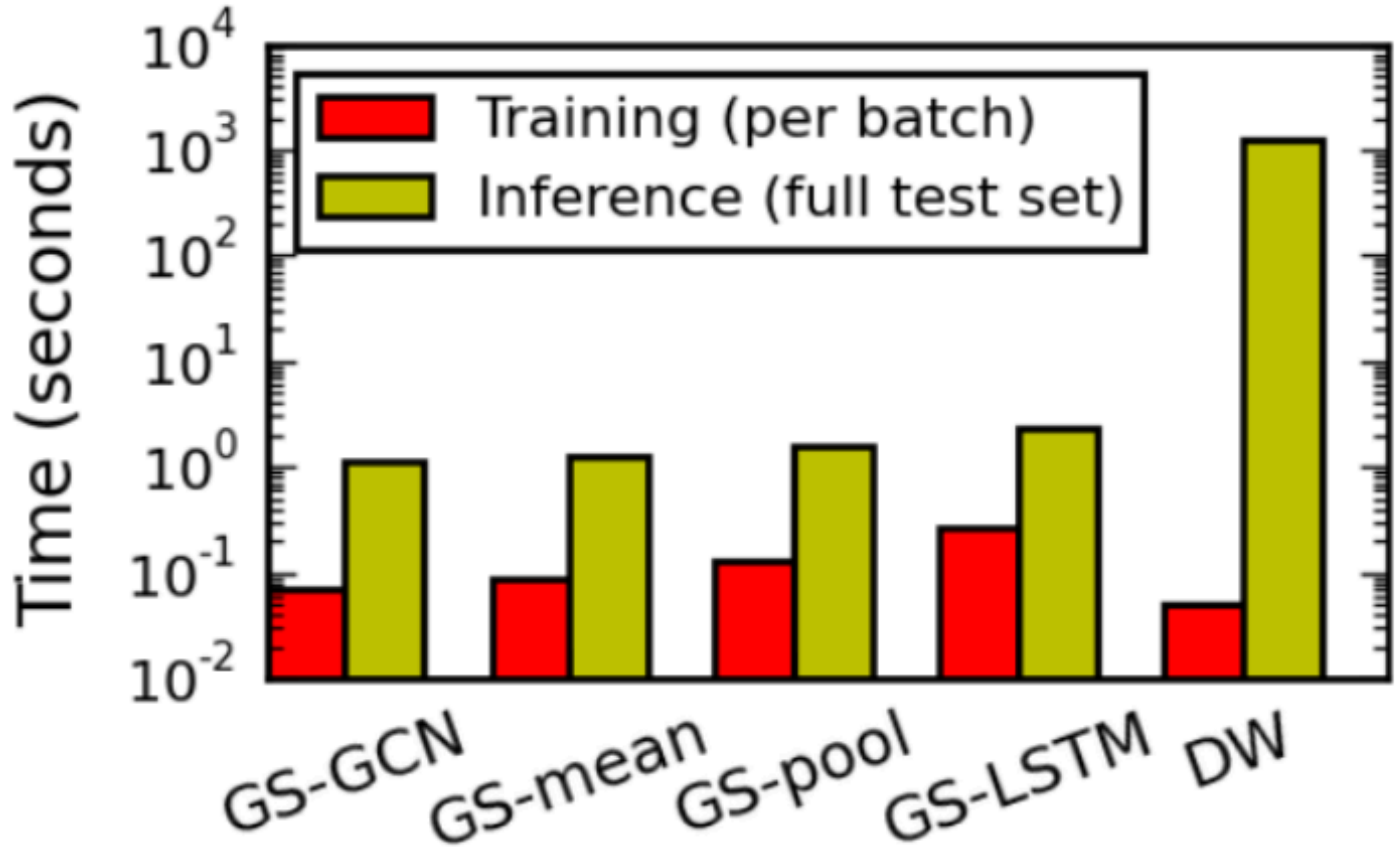
Results

Name	Citation Unsup. F1	Citation Sup. F1	Reddit Unsup. F1	Reddit Sup. F1	PPI Unsup. F1	PPI Sup. F1
Random	0.206	0.206	0.043	0.042	0.396	0.396
Raw features	0.575	0.575	0.585	0.585	0.422	0.422
DeepWalk	0.565	0.565	0.324	0.324	—	—
DeepWalk + features	0.701	0.701	0.691	0.691	—	—
GraphSAGE-GCN	0.742	0.772	**0.908**	0.930	0.465	0.500
GraphSAGE-mean	0.778	0.820	0.897	0.950	0.486	0.598
GraphSAGE-LSTM	0.788	0.832	**0.907**	**0.954**	0.482	**0.612**
GraphSAGE-pool	**0.798**	**0.839**	0.892	0.948	**0.502**	0.600
% gain over feat.	39%	46%	55%	63%	19%	45%

GraphSAGE significantly outperforms the baseline approaches

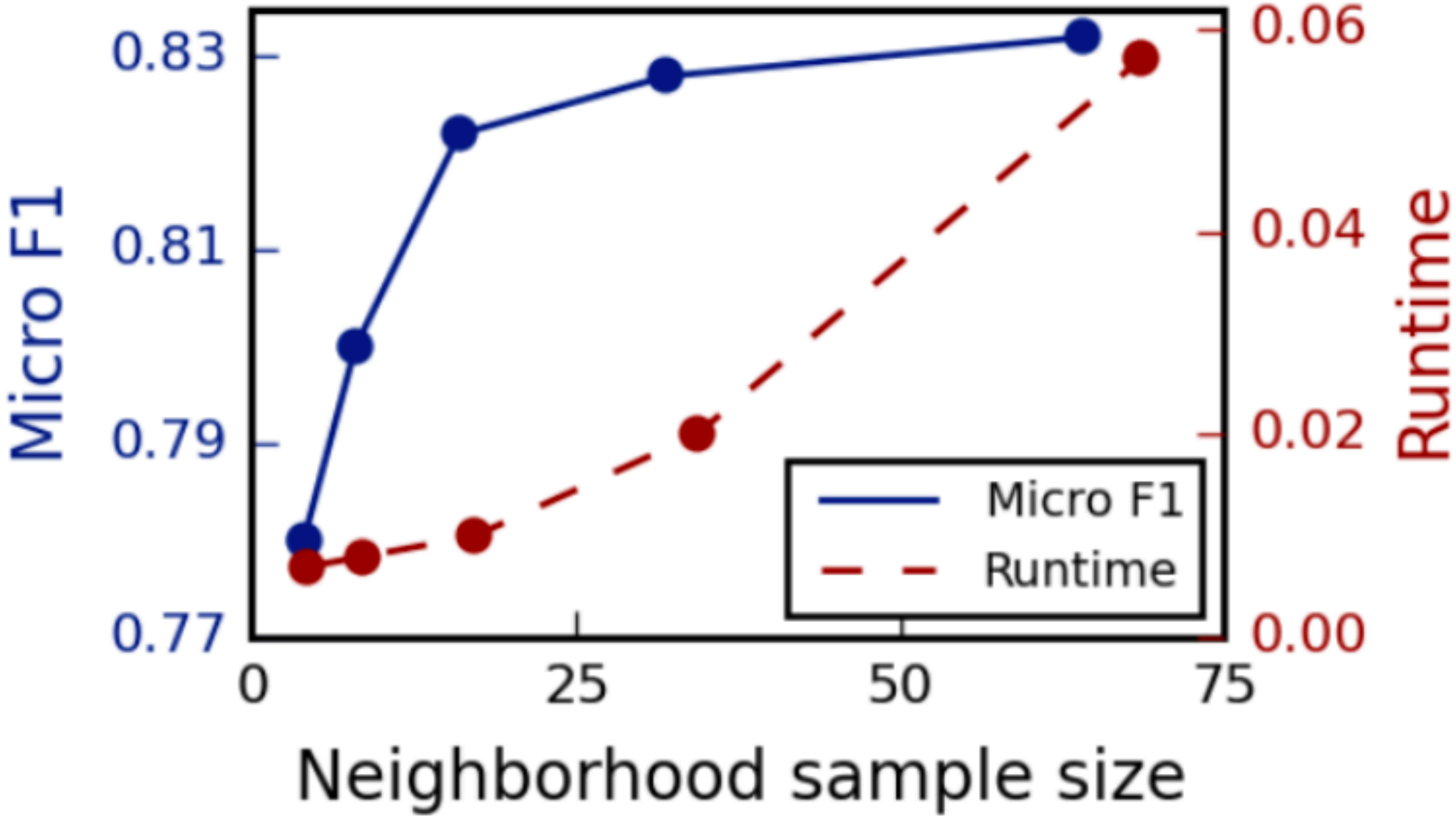
Results

Timing experiments



DeepWalk 100-500× slower at test time

Size of the sampled neighborhood



K = 2 with S1 = S2

Analyze

- ☑ A "**deep**" model rather than traditional "shallow" model(eg. DeepWalk).
- ☑ Leverages both node **feature information** and graph structure rather than only graph structure.
- ☑ An **inductive** model rather than a transduction model.
- ☑ Apply on **large** graphs(eg. 10 billion nodes).

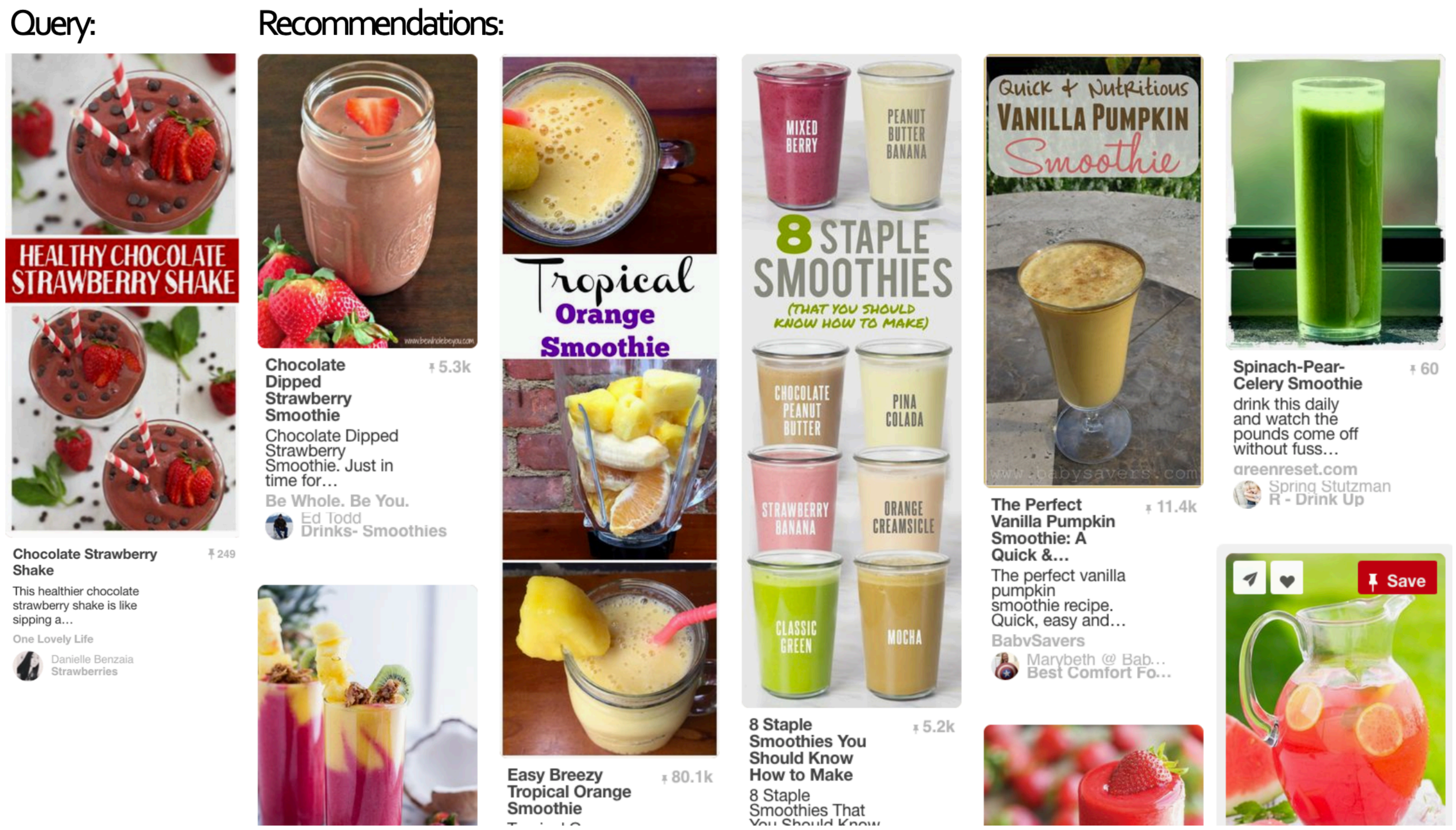
- ☐ Can't learn on **directed** or **multi-modal** graphs.
- ☐ Use a **uniform** neighborhood sampling functions.
- ☐ Apply the **same weights** to different neighborhoods.

Conclusion

- Present GraphSAGE to solve representation of nodes on large graphs.
- It's a general inductive framework that leverages node feature information and graph structure.
- It outperforms strong baselines on three node-classification benchmark.

https://github.com/williamleif/GraphSAGE

Application : PinSAGE

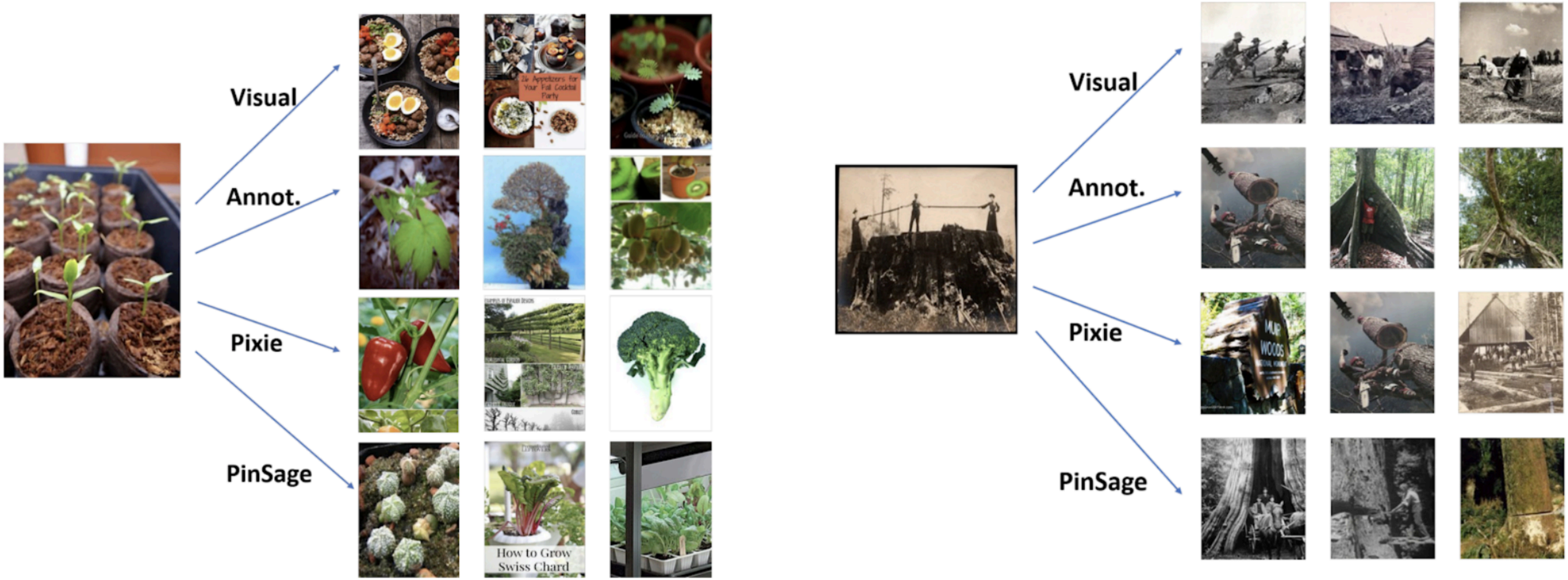


Pinterest + Stanford

3 billion nodes representing pins and boards, and 18 billion edges

Graph Convolutional Neural Networks for Web-Scale Recommender Systems , Jure Leskovec, 2018 KDD

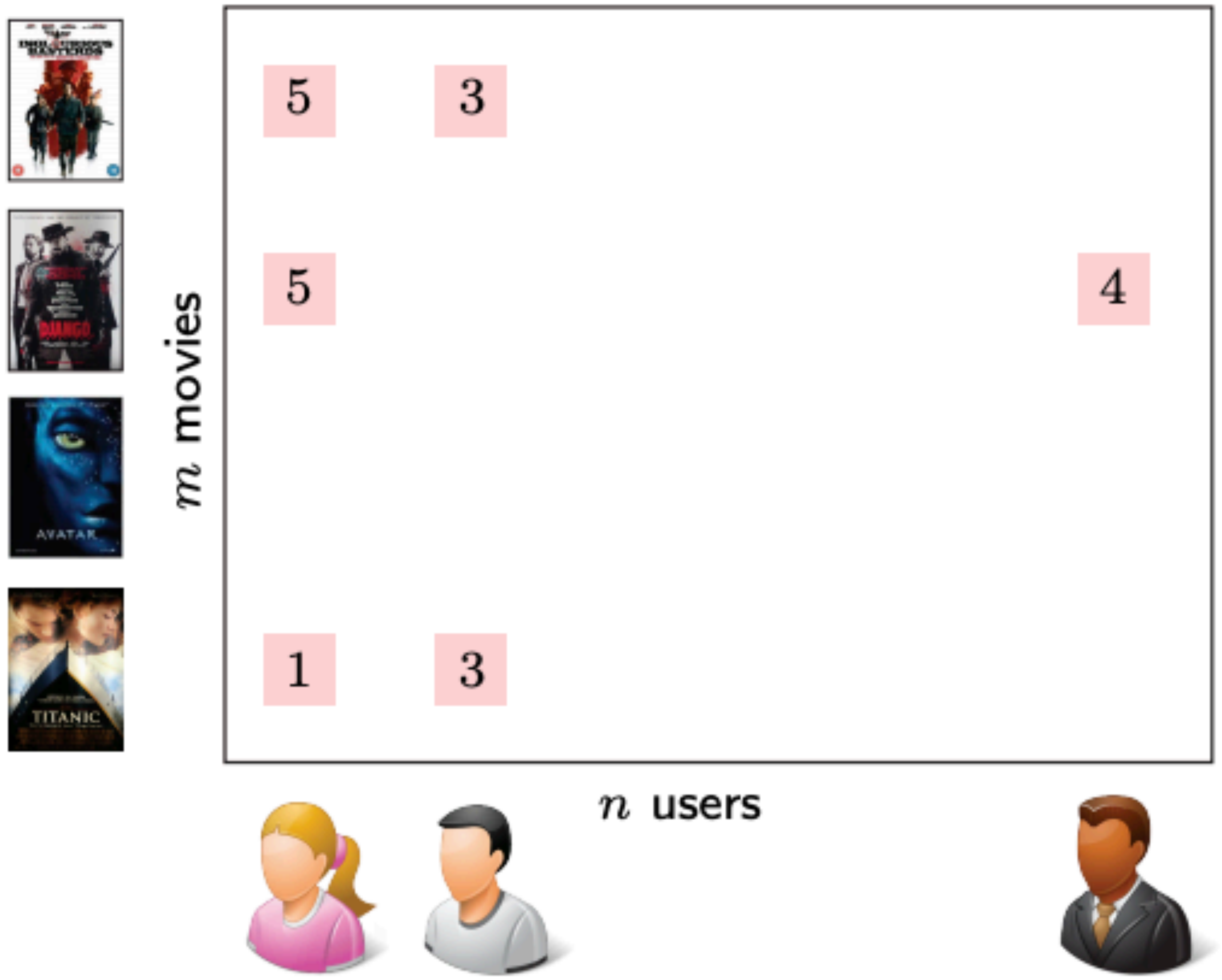
Application : PinSAGE

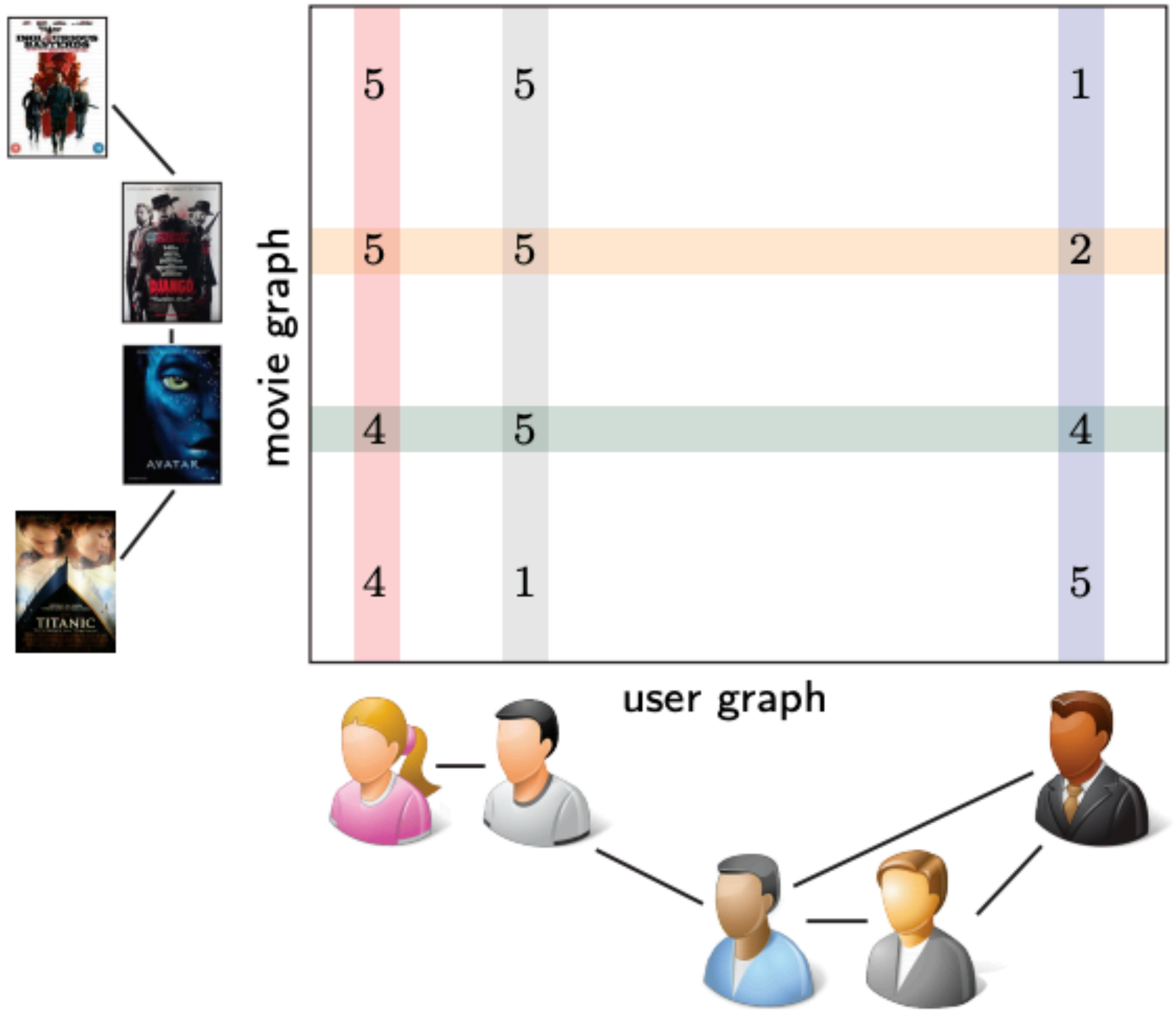


Graph Convolutional Neural Networks for Web-Scale Recommender Systems,Jure Leskovec, 2018 KDD

Application

Recommender System





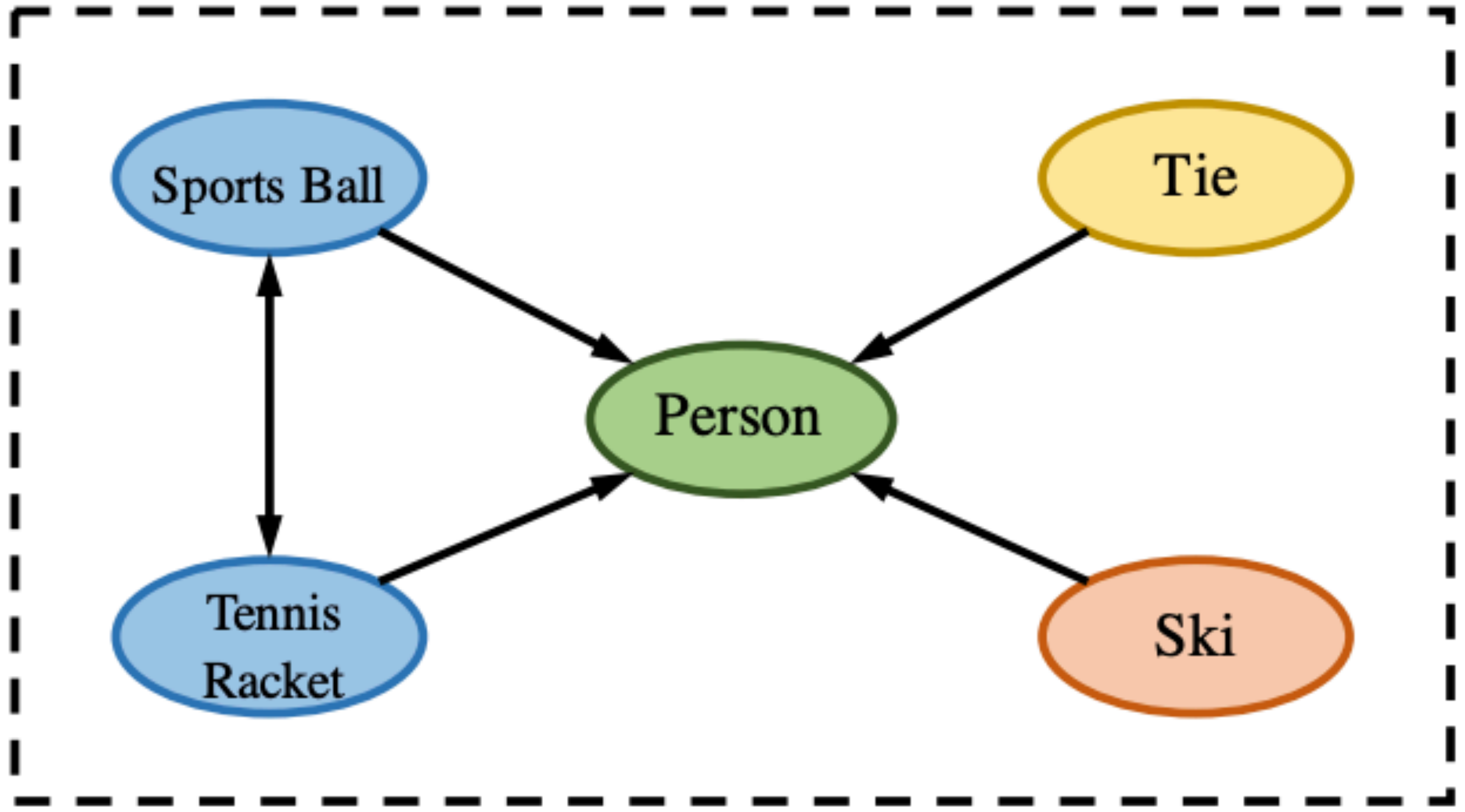
Application

Multi-Label Image Recognition

Person, Sports Ball, Tennis Racket

Person, Tie

Person, Ski



Multi-Label Image Recognition with Graph Convolutional Networks, Zhao-Min Chen, CVPR 2019

Application

3. Applications	
3.1 Physics	3.2 Chemistry and Biology
3.3 Knowledge Graph	3.4 Recommender Systems
3.5 Computer Vision	3.6 Natural Language Processing
3.7 Generation	3.8 Combinatorial Optimization
3.9 Adversarial Attack	3.10 Graph Clustering
3.11 Graph Classification	3.12 Reinforcement Learning
3.13 Traffic Network	3.14 Few-shot and Zero-shot Learning
3.15 Program Representation	3.16 Social Network

https://github.com/thunlp/GNNPapers