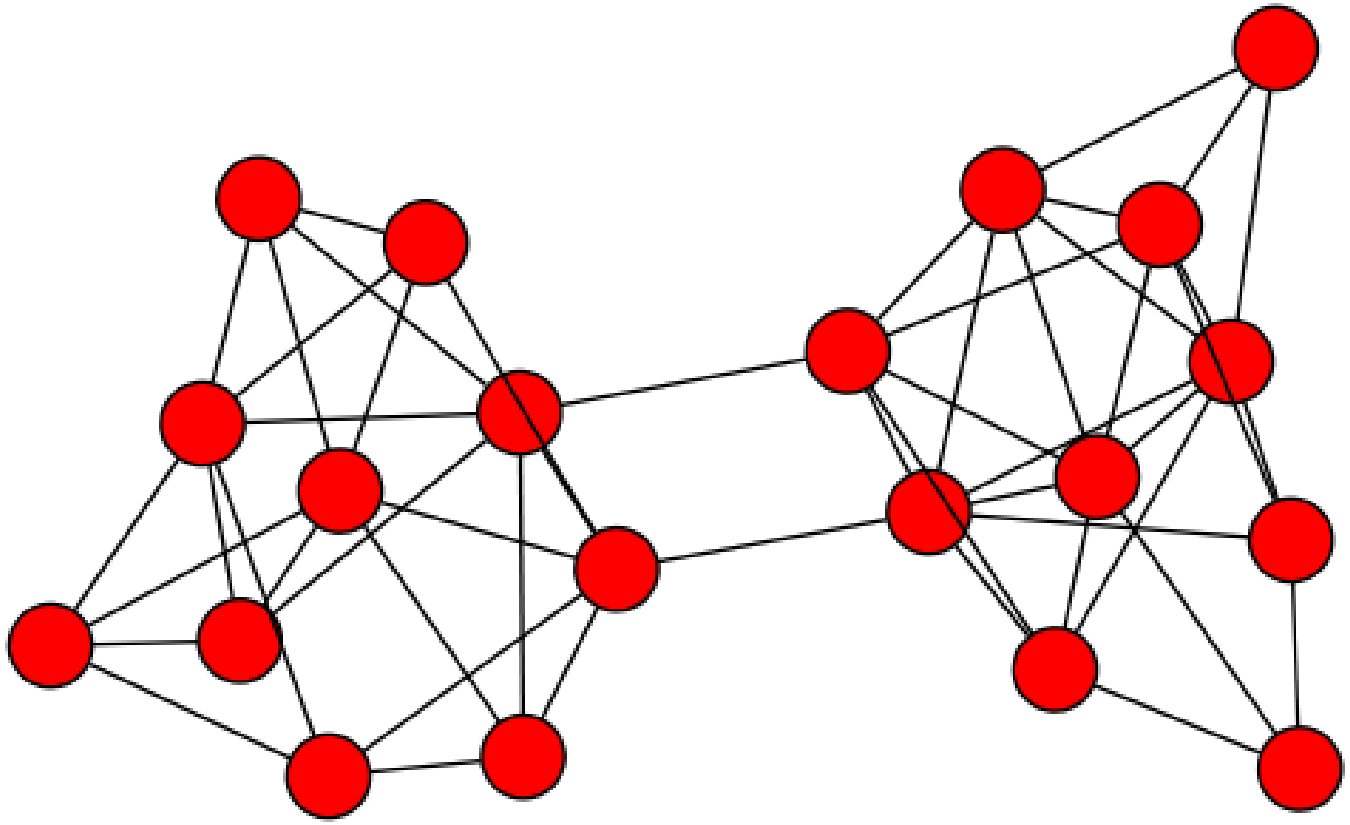
Frontiers in GNN and Network Embedding

Peng Cui, Ziwei Zhang

Tsinghua University

Network (Graph)

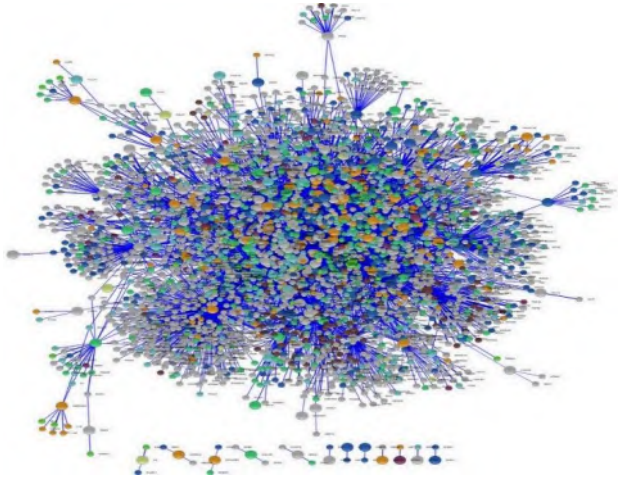
The general description of data and their relations.



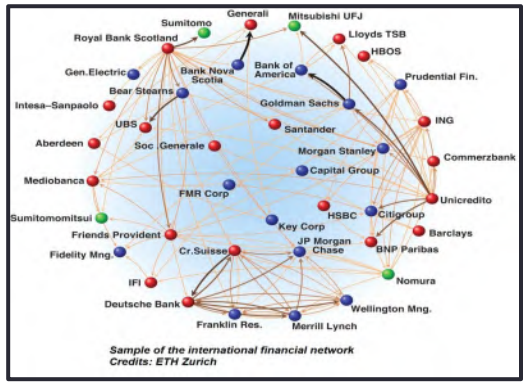
Many types of data are networks

Social Networks

Biology Networks



Finance Networks



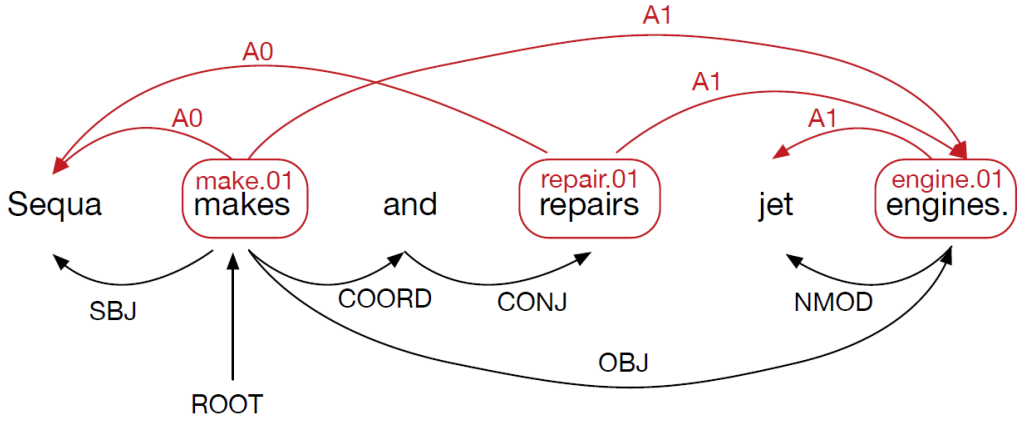
Internet of Things

Information Networks

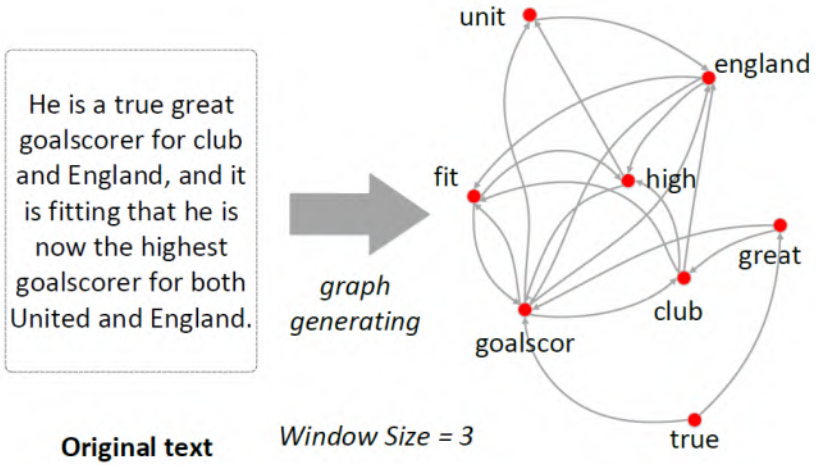
Logistic Networks

Graphs in NLP

Syntactic Dependency



Word Co-occurrences



Abstract Meaning Representation

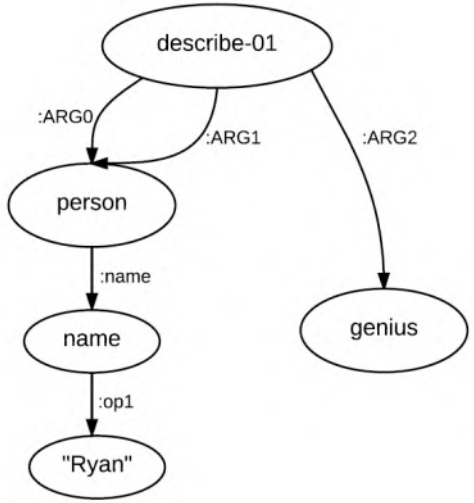
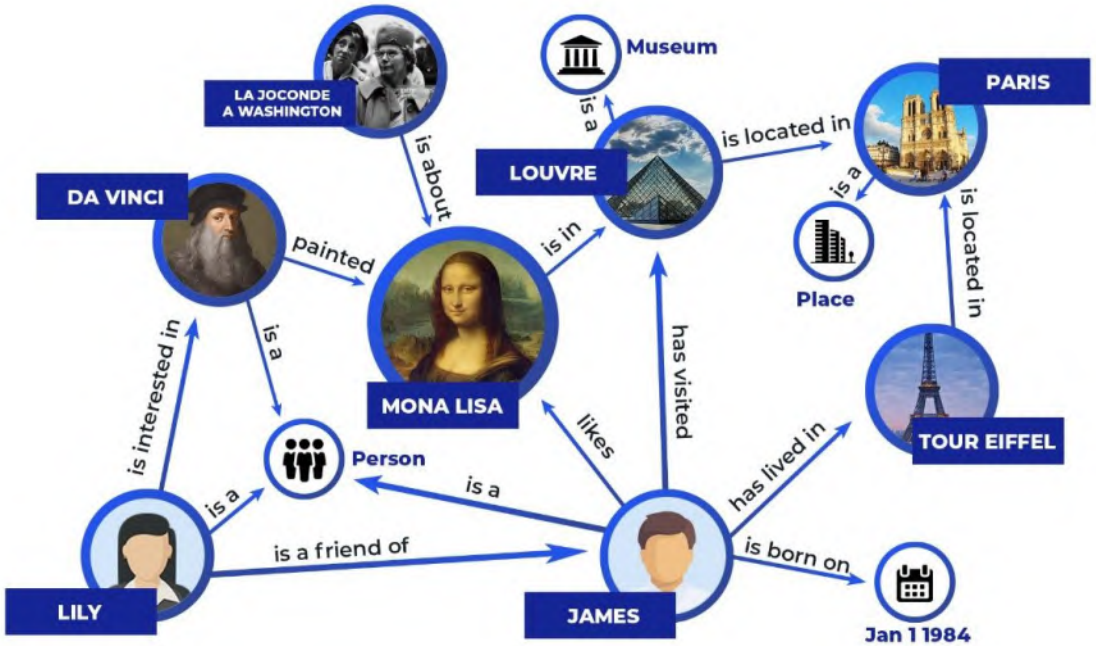


Figure 1: An example of AMR graph meaning "Ryan's description of himself: a genius."

Encoding Sentences with Graph Convolutional Networks for Semantic Role Labeling, *EMNLP 2017*
Large-Scale Hierarchical Text Classification with Recursively Regularized Deep Graph-CNN, *WWW 2018*
A Graph-to-Sequence Model for AMR-to-Text Generation, *ACL 2018*

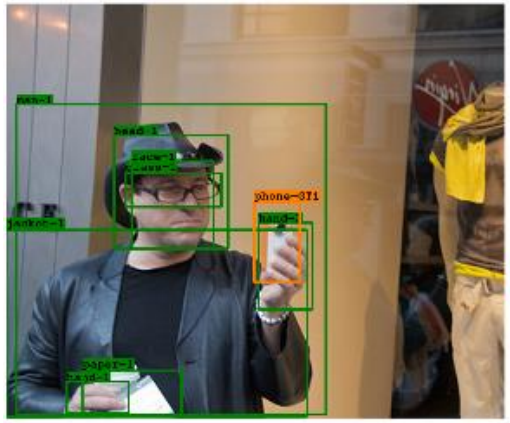
Knowledge Graph

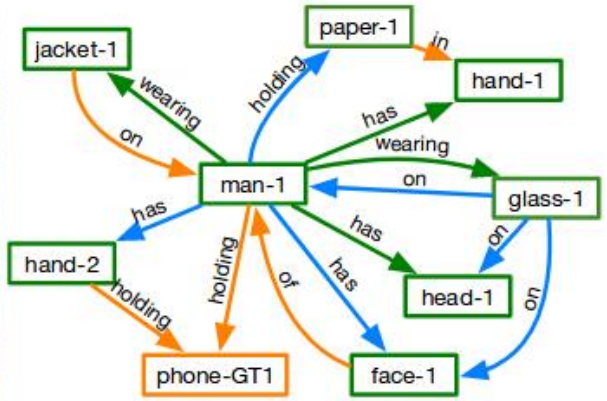


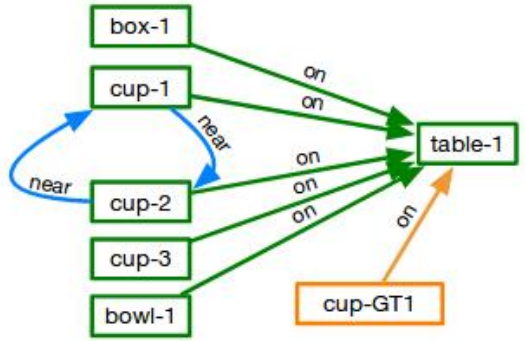
https://yashuseth.blog/2019/10/08/introduction-question-answering-knowledge-graphs-kgqa/
Building and Querying an Enterprise Knowledge Graph, *TSC 2019*

NLP + Computer Vision

Scene Graph



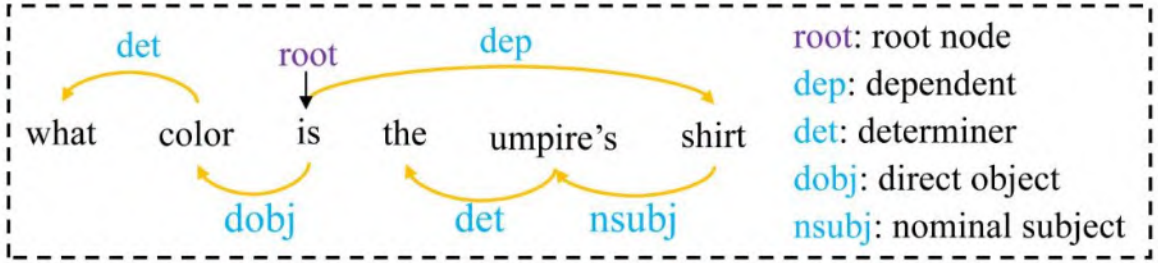




Visual Question Answering

(a) **Q**: What color is the umpire's shirt (b) **Q**: What color is the umpire's shirt

Ground True Answer: blue **Predicted Answer**: black



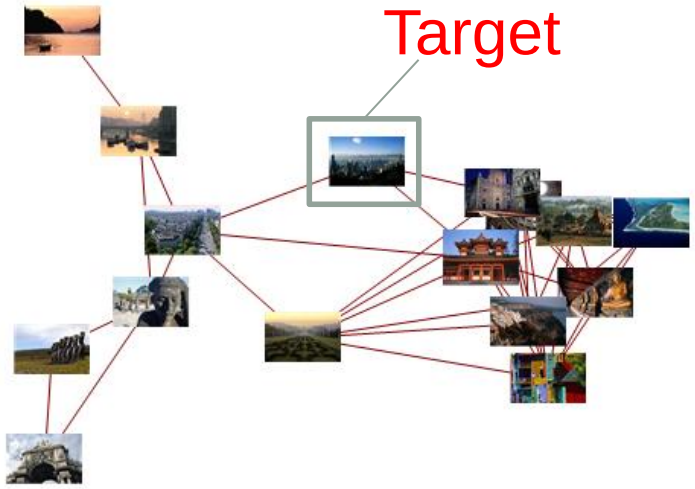
(c) Dependency parsing of the question

Neural Motifs: Scene Graph Parsing with Global Context, *CVPR 2018*
Aligned Dual Channel Graph Convolutional Network for Visual Question Answering, *ACL 2020*

Why network is important?

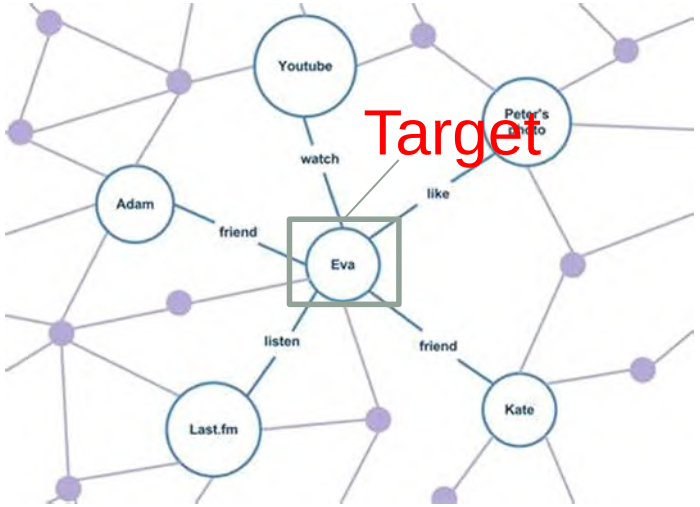
In few cases, you only care about a subject but not its relations with other subjects.

Image Characterization



Reflected by relational subjects

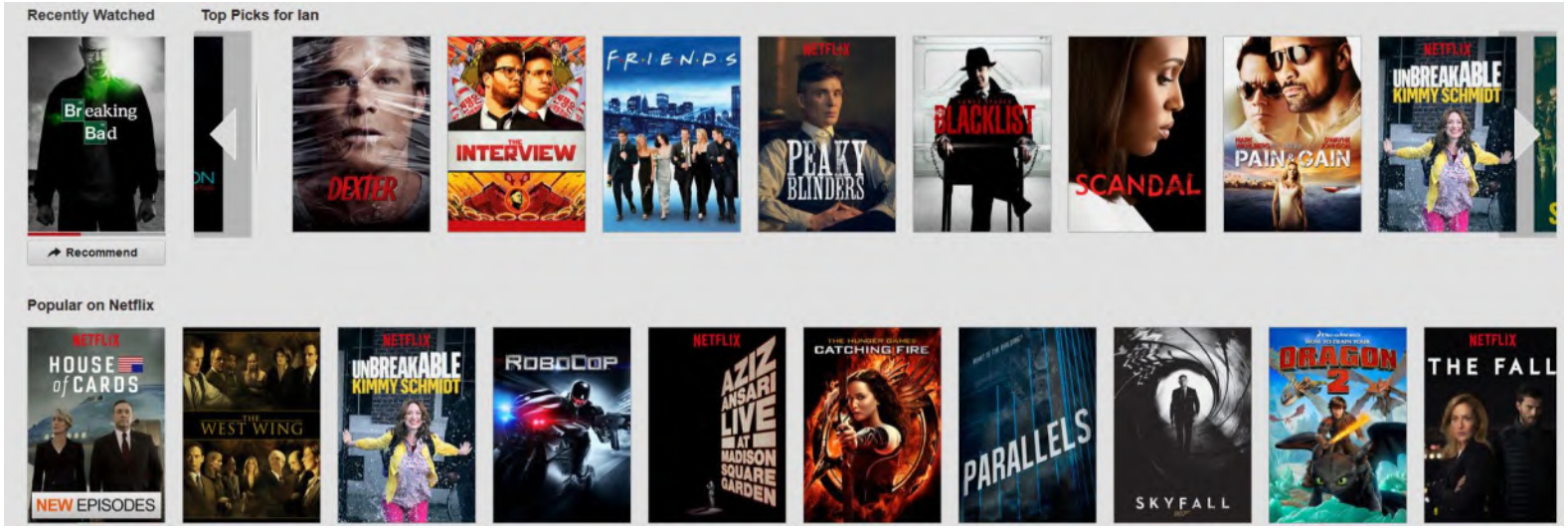
Social Capital



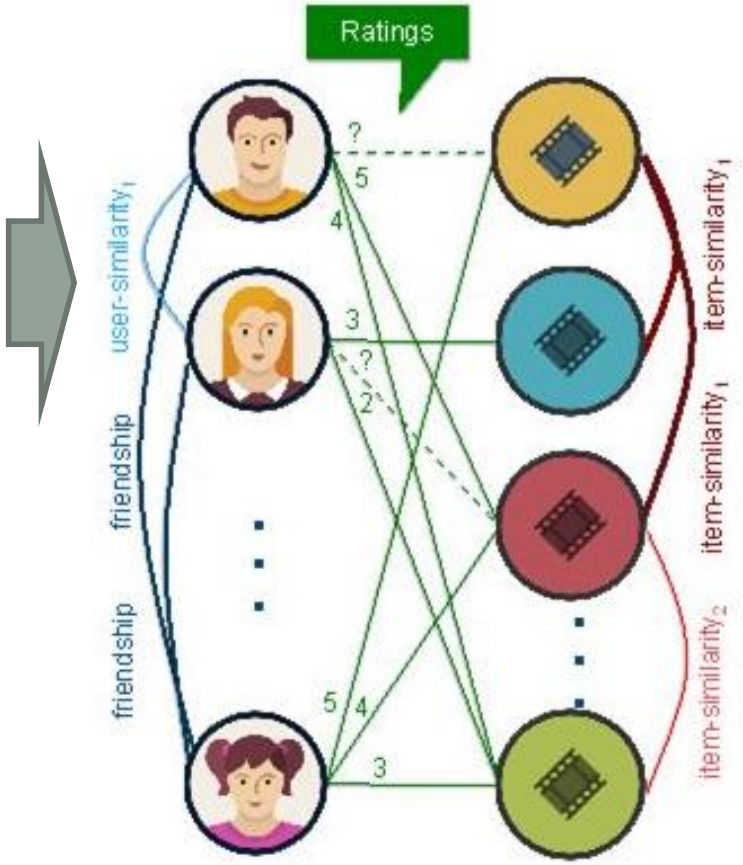
Decided by relational subjects

Many applications are intrinsically network problems

Recommendation Systems



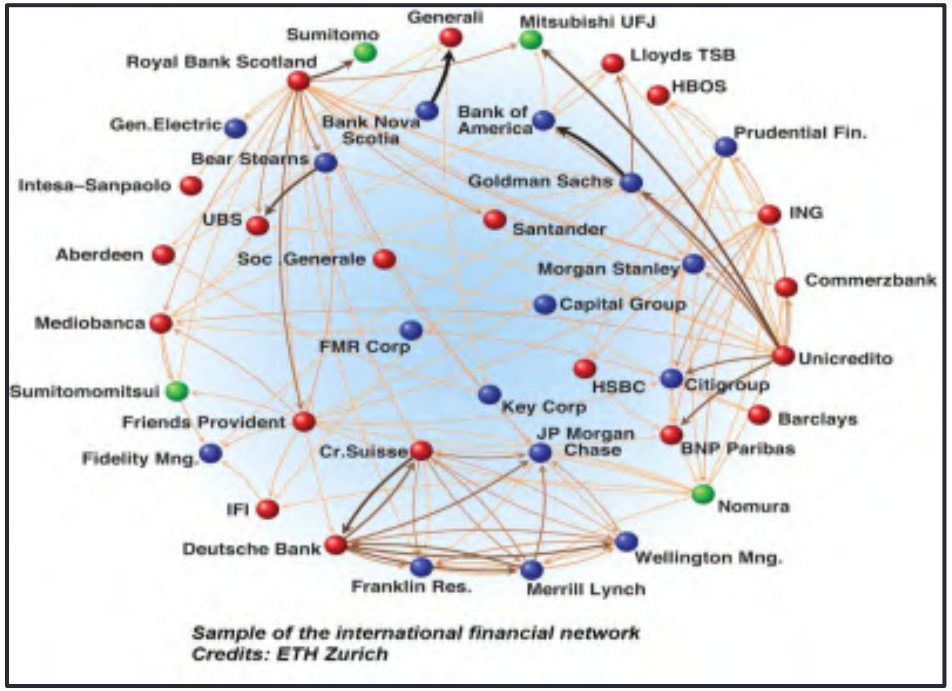
Link prediction in bipartite graphs



Many applications are intrinsically network problems

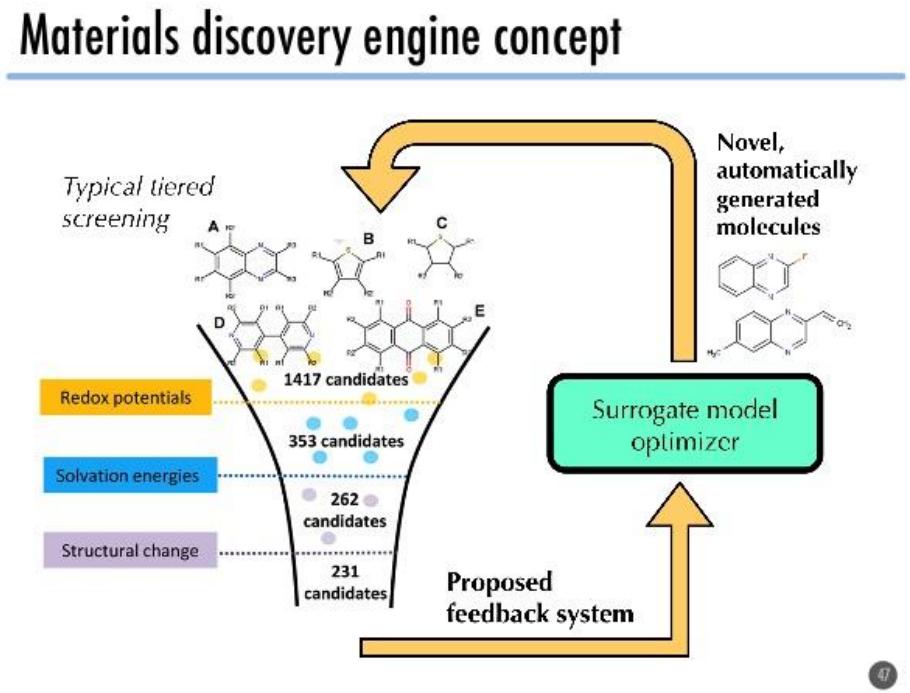
Financial credit & risk management

Node importance & classification

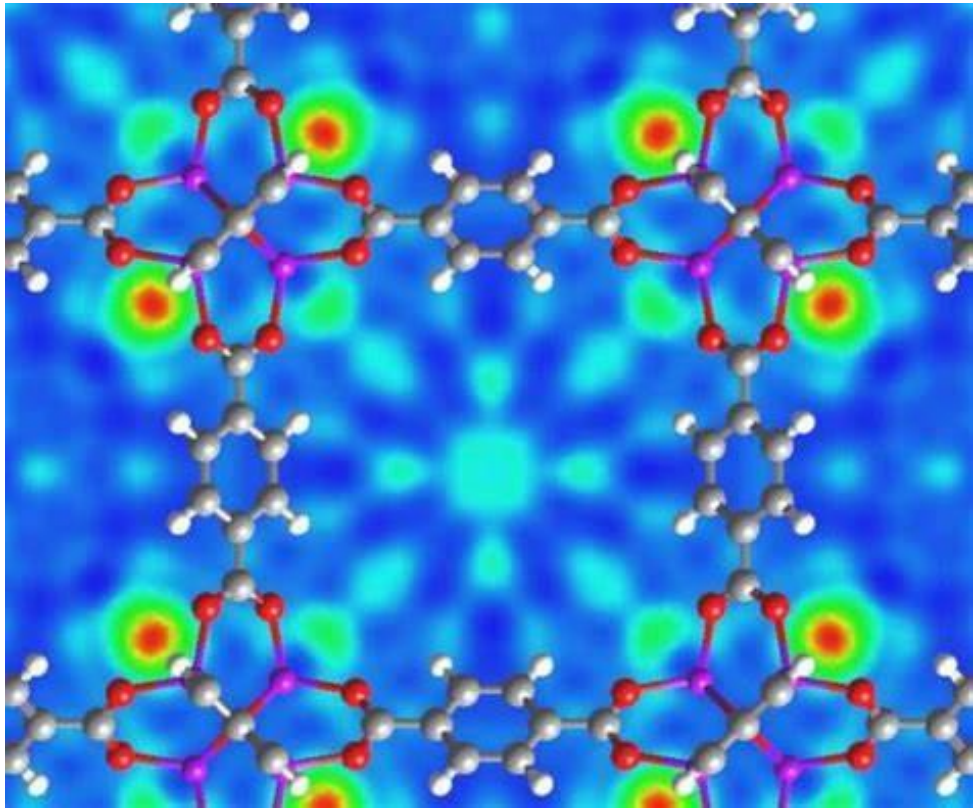


Many applications are intrinsically network problems

New material discovery



Subgraph pattern discovery

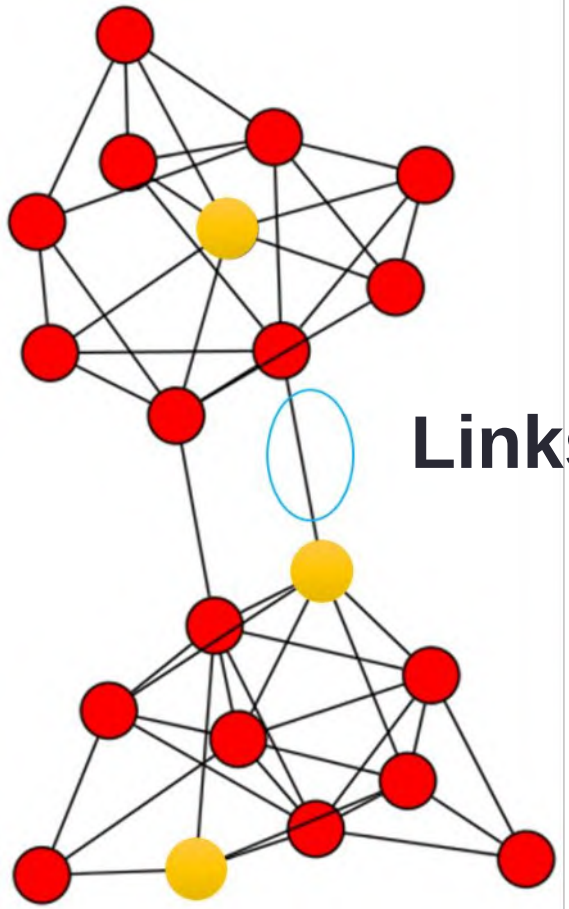


Graph as a data model

- The last resort for the curse of complexity in real applications
  - Geographical networks, relationships, etc.
- Divide-and-conquer in modeling
  - Individual nodes and edges are well structured
  - Global structures are weakly organized

Networks are not *learning-friendly*

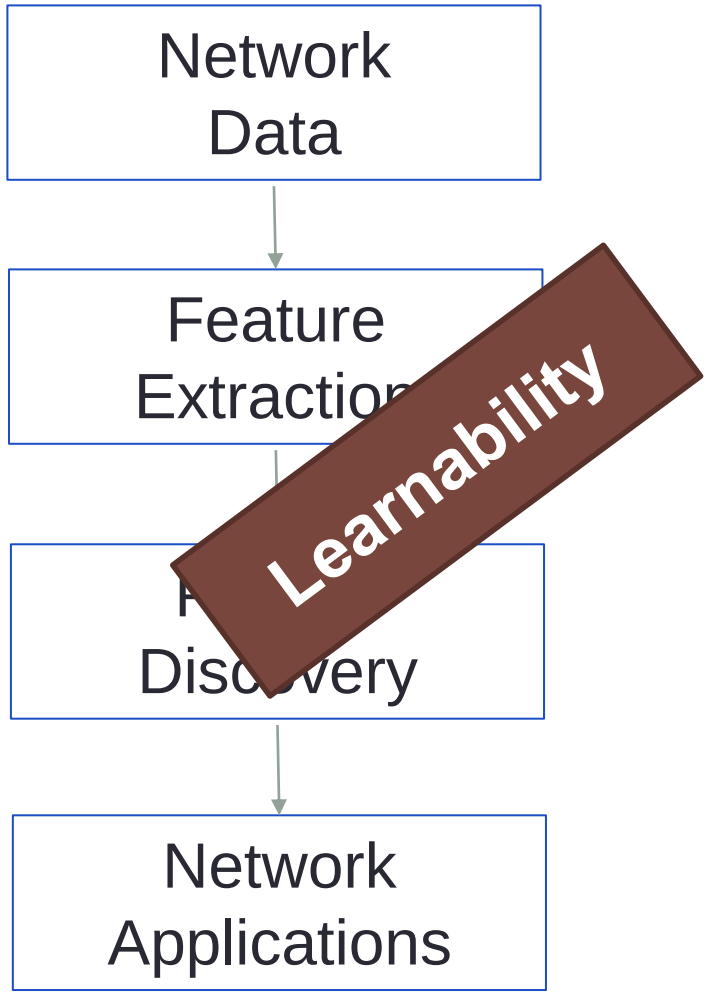
G = (V, E)



Links ⇨ **Topology**

Inapplicability of ML methods

Pipeline for network analysis

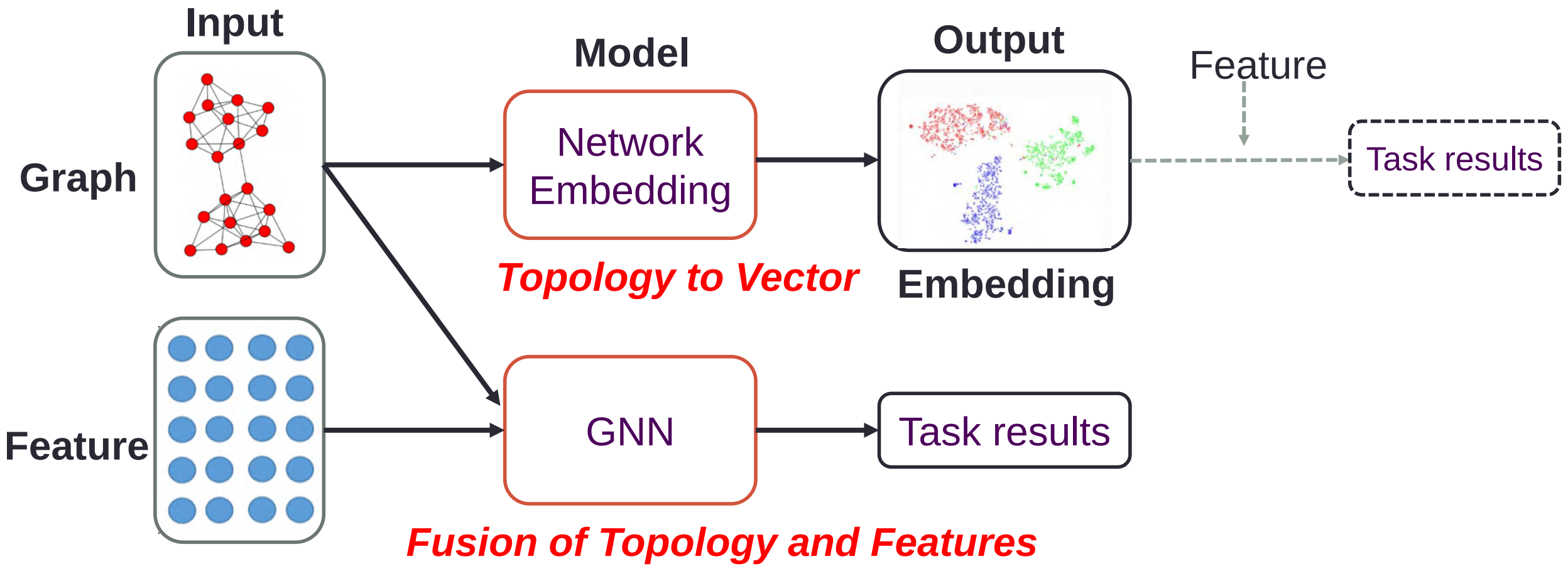


Learning from networks

Network Embedding

GNN

Network Embedding and GNN



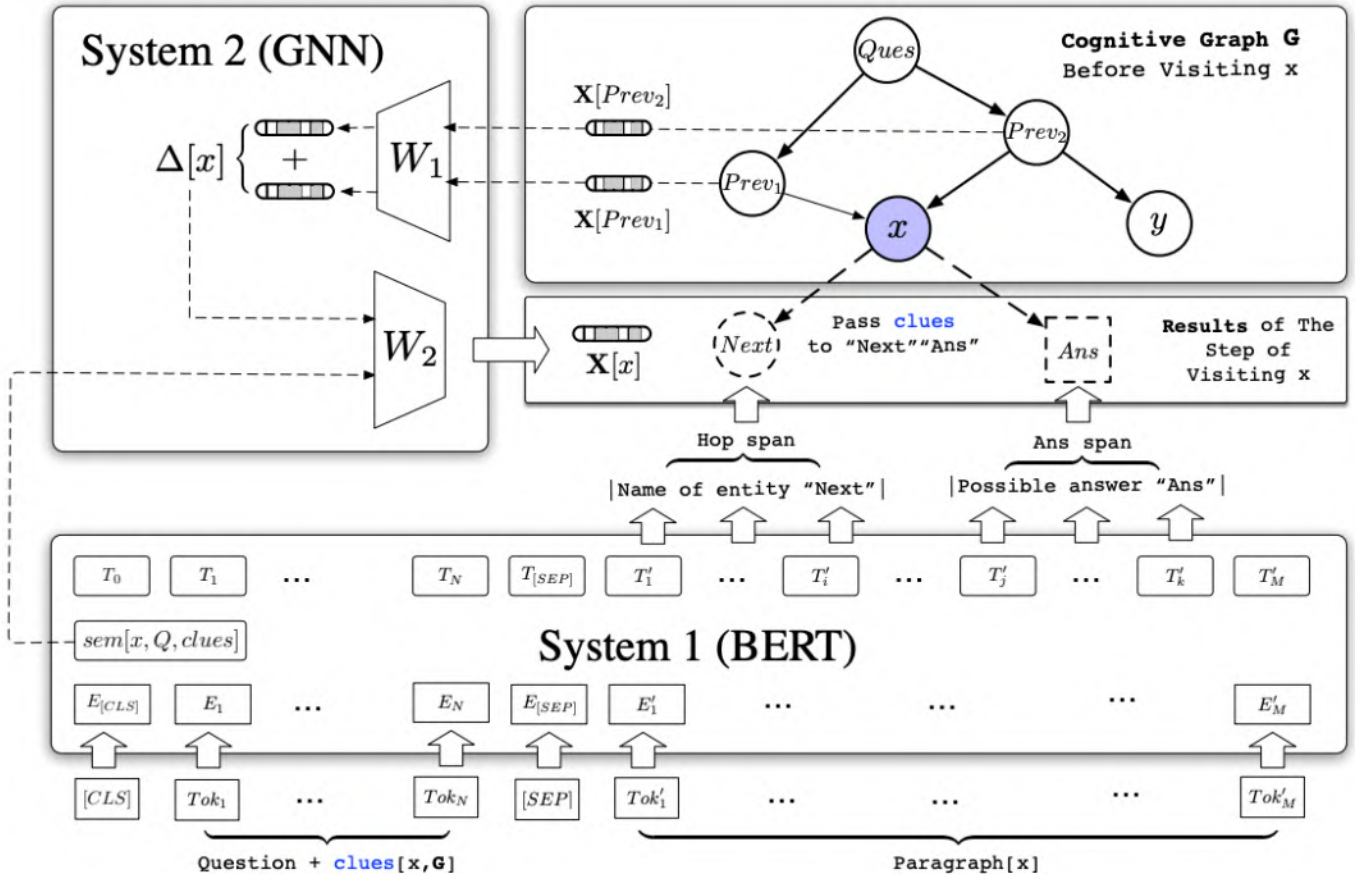
Unsupervised vs. (Semi-)Supervised

Graph Neural network vs. Network embedding

- In some sense, they are different
- **Graphs** exist in ***mathematics*** (Data Structure)
  - Mathematical structures used to model pairwise relations between objects
- **Networks** exist in the ***real world*** (Data)
  - Social networks, logistic networks, biology networks, transaction networks, etc.
- A network can be represented by a graph
- A dataset that is not a network can also be represented by a graph

GNN for Natural Language Processing

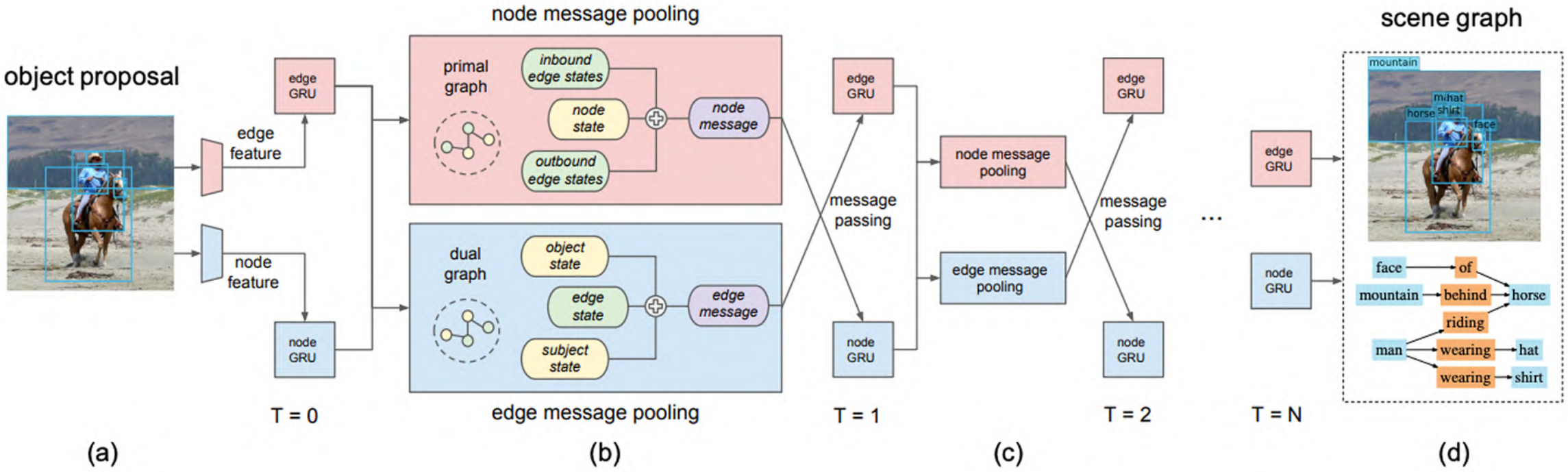
- Many papers on BERT + GNN.
- BERT is for retrieval.
  - It creates an initial graph of relevant entities and the initial evidence.
- GNN is for reasoning.
  - It collects evidence (i.e., old messages on the entities) and arrive at new conclusions (i.e., new messages on the entities), by passing the messages around and aggregating them.



Cognitive Graph for Multi-Hop Reading Comprehension at Scale. Ding et al., *ACL 2019.*
Dynamically Fused Graph Network for Multi-hop Reasoning. Xiao et al., *ACL 2019.*

GNN for Computer Vision

- A popular trend in CV is to construct a graph during the learning process
  - To process multiple objects or parts in a scene, and to infer their relationships
- Example: Scene graphs



Scene Graph Generation by Iterative Message Passing. Xu et al., *CVPR 2017.*
Image Generation from Scene Graphs. Johnson et al., *CVPR 2018.*

GNN for Symbolic Reasoning

- We can view the process of symbolic reasoning as a directed acyclic graph
- Many recent efforts use GNNs to perform symbolic reasoning

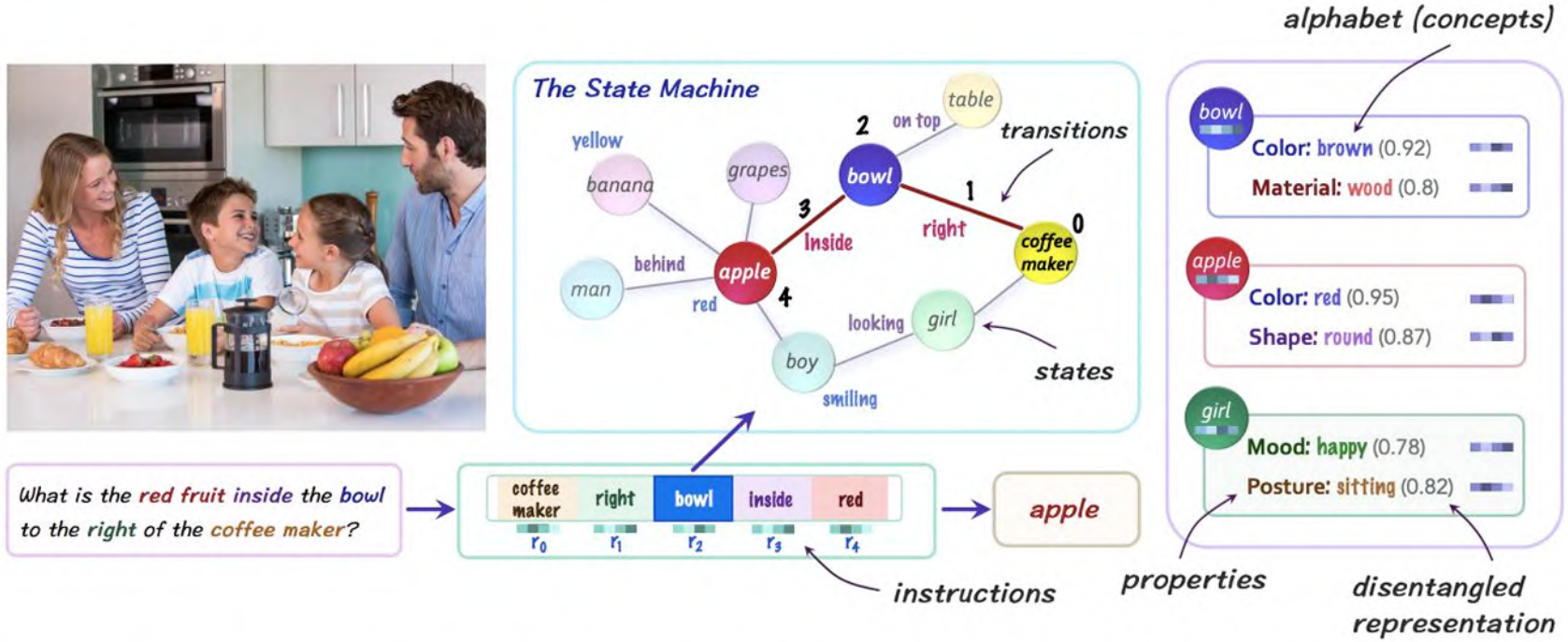
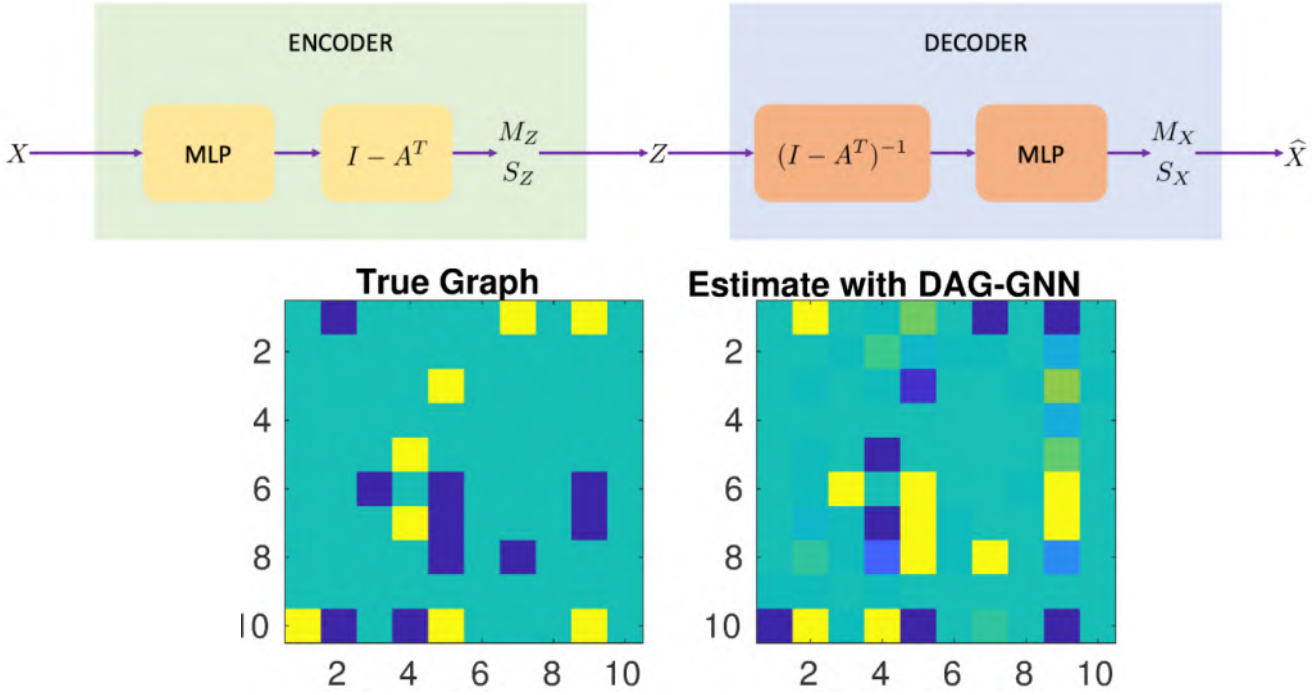


Figure 1: The Neural State Machine is a graph network that simulates the computation of an automaton.

Learning by Abstraction: The Neural State Machine. Hudson & Manning, *NeurIPS 2019*
Can Graph Neural Networks Help Logic Reasoning? Zhang et al., *arXiv 1906.02111*
Symbolic Graph Reasoning Meets Convolutions. Liang et al., *NeurIPS 2018*

GNN for Structural Equation Modeling

- Structural equation modeling, a form of causal modeling, tries to describe the relationships between the variables as a directed acyclic graph (DAG)
- GNN can be used to represent a nonlinear structural equation and help find the DAG, after treating the adjacency matrix as parameters



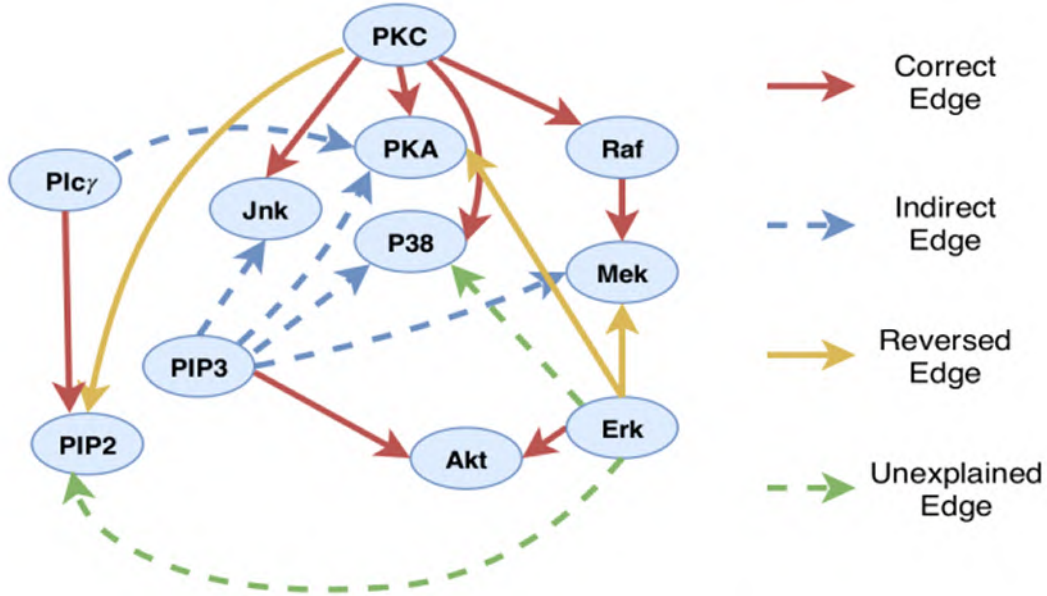
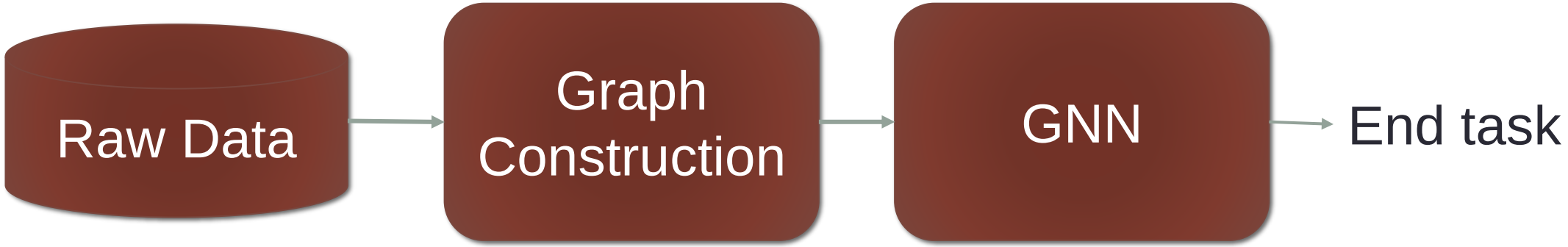


Figure 8. Estimate protein signaling network.

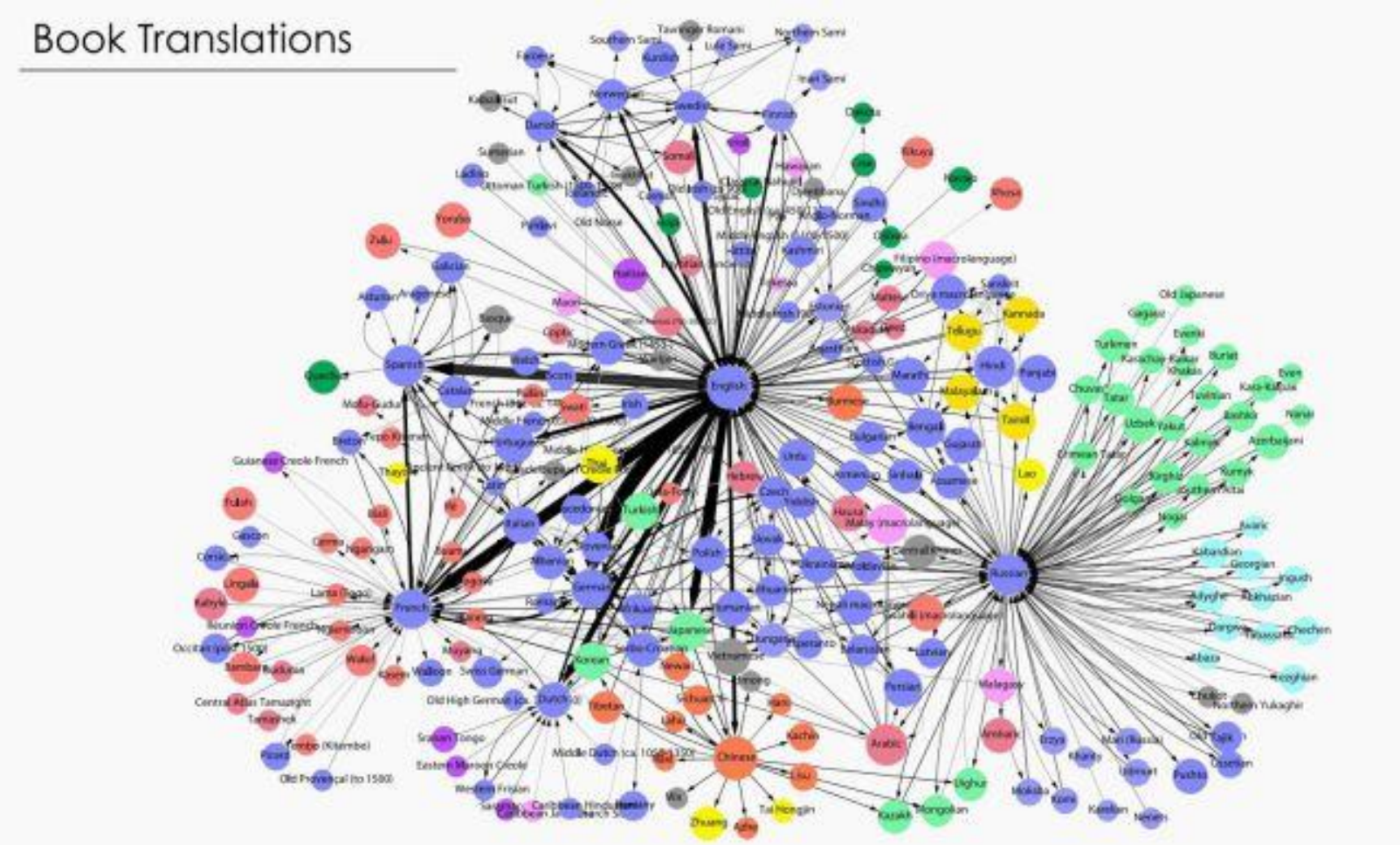
DAG-GNN: DAG Structure Learning with Graph Neural Networks. Yu et al., *ICML 2019*

Pipeline for (most) GNN works



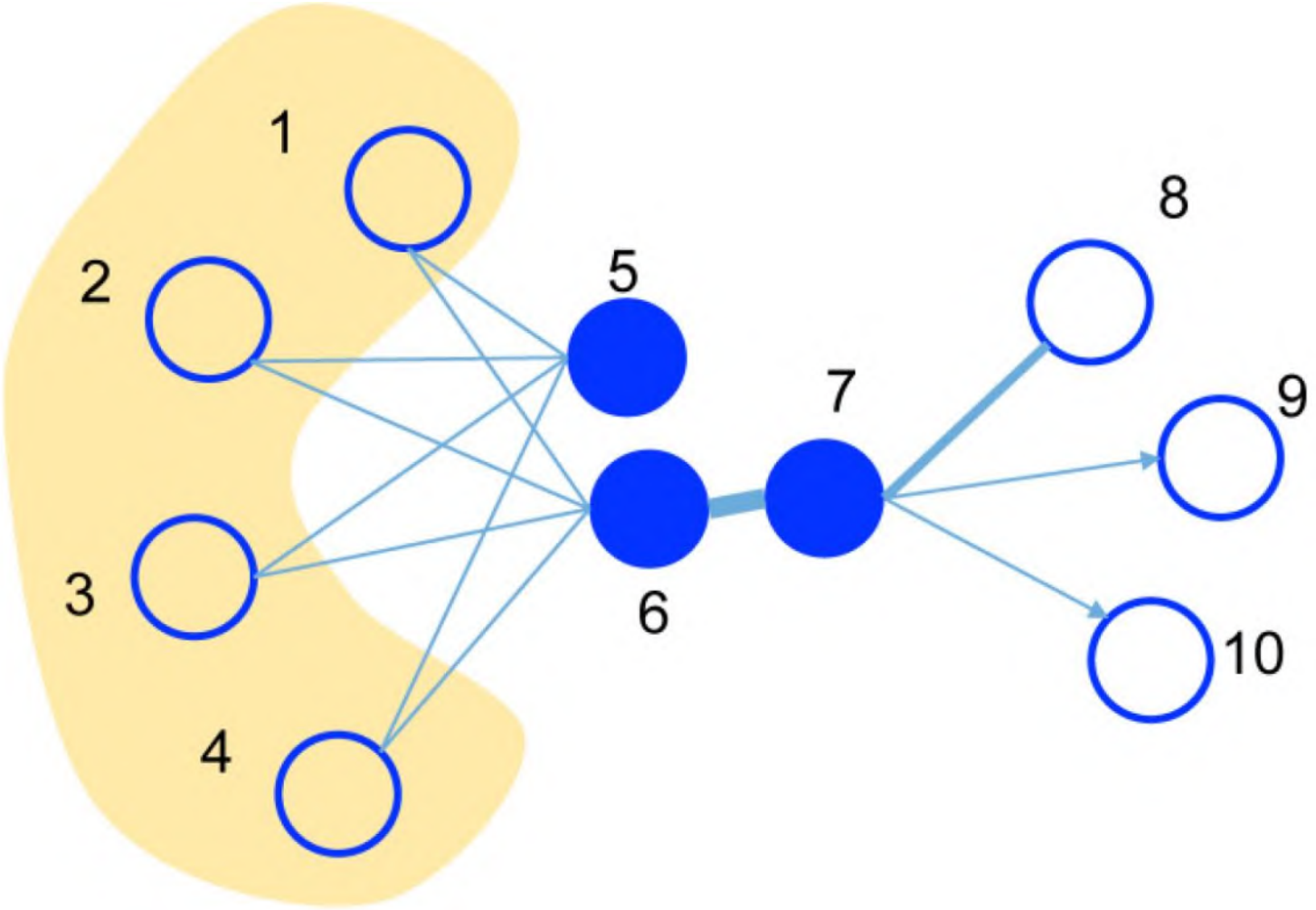
Network embedding: topology to vector

- Co-occurrence (neighborhood)



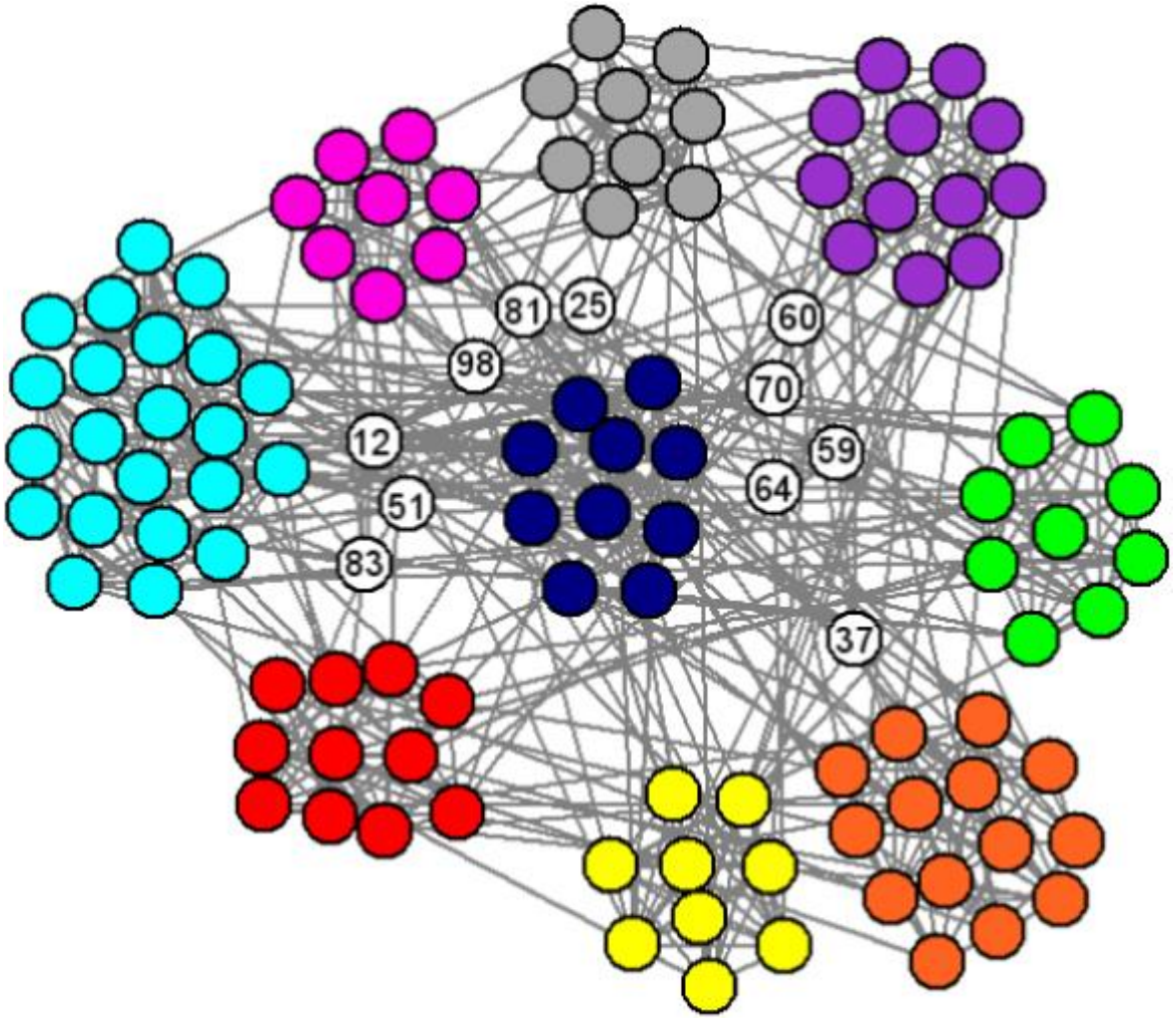
Network embedding: topology to vector

- High-order proximities



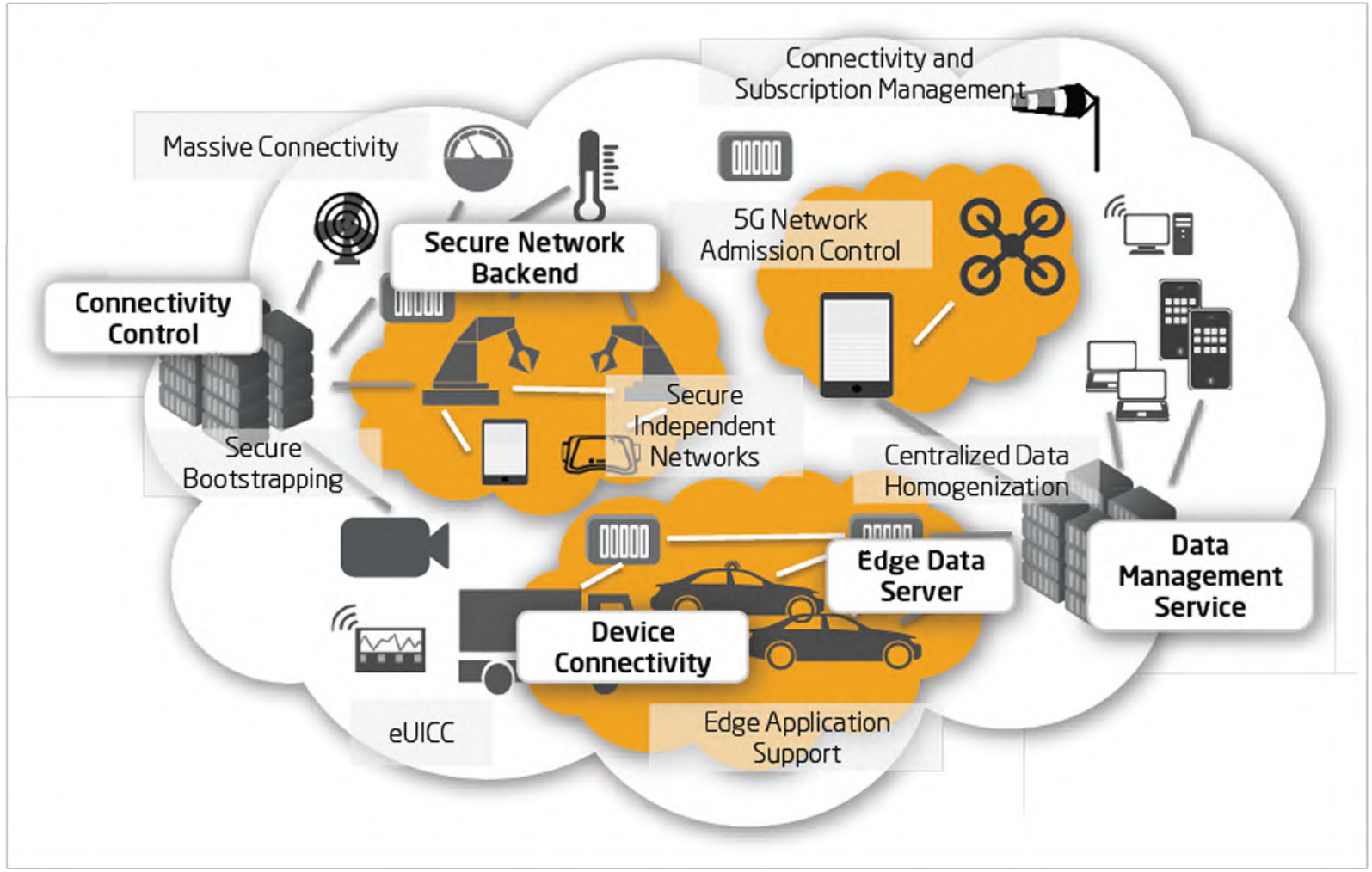
Network embedding: topology to vector

- Communities

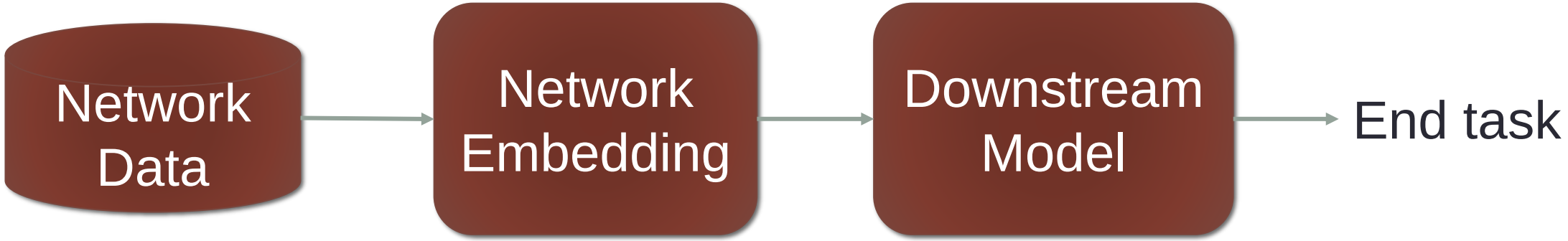


Network embedding: topology to vector

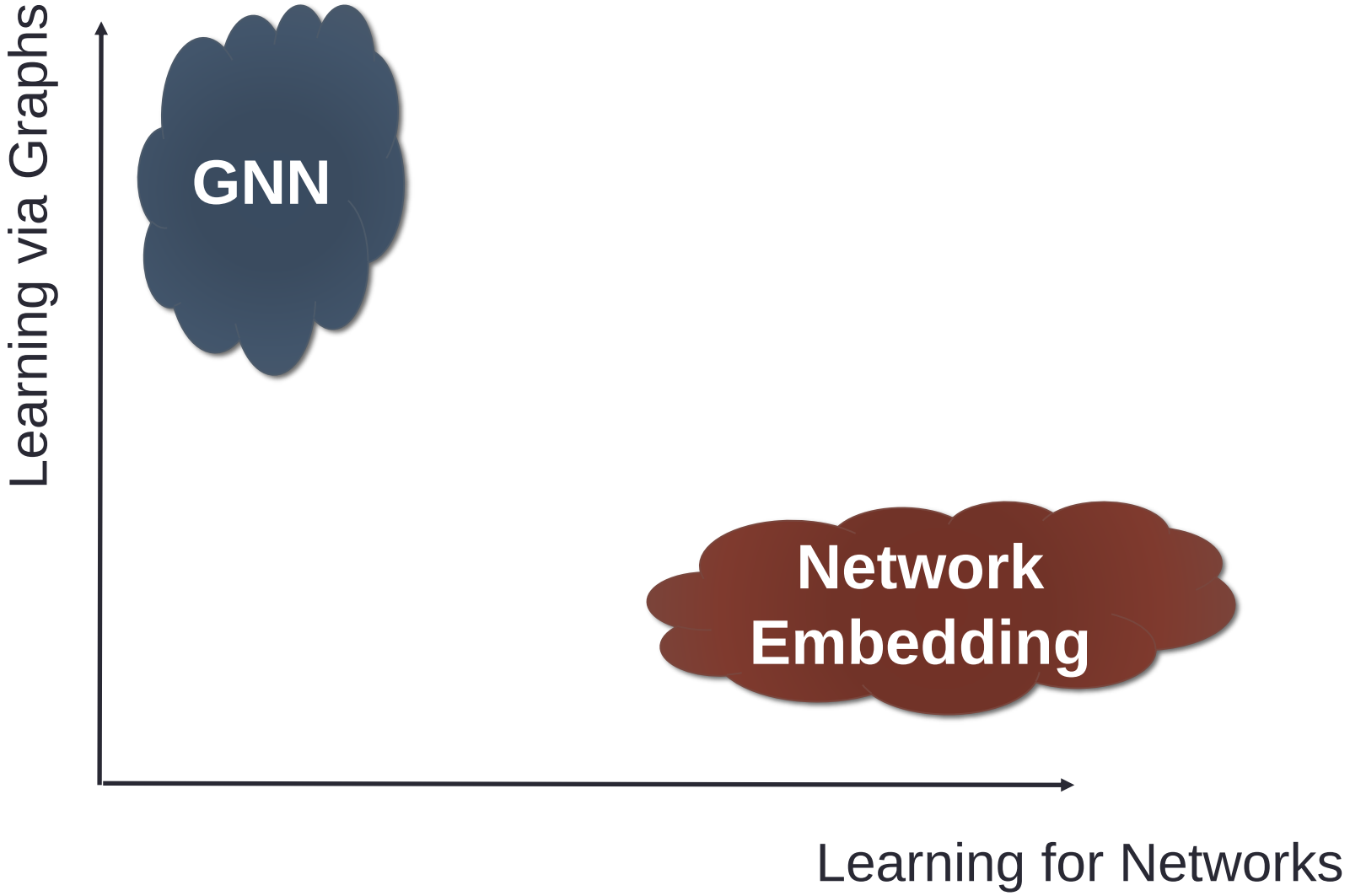
- Heterogeneous networks



Pipeline for (most) Network Embedding works



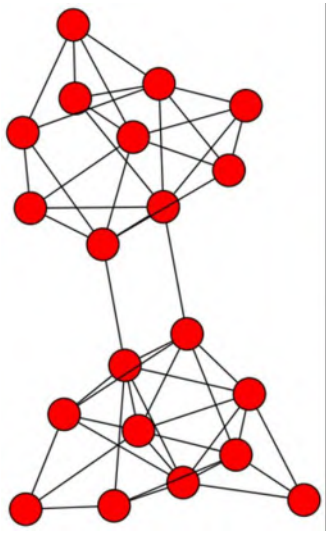
Learning for Networks vs. Learning via Graphs



The intrinsic problems NE is solving

Reducing representation dimensionality while preserving necessary topological **structures** and **properties**.

- Nodes & Links
- Node Neighborhood
- Pair-wise Proximity
- Community
- Hyper Edges
- Global Structure



generate

embed

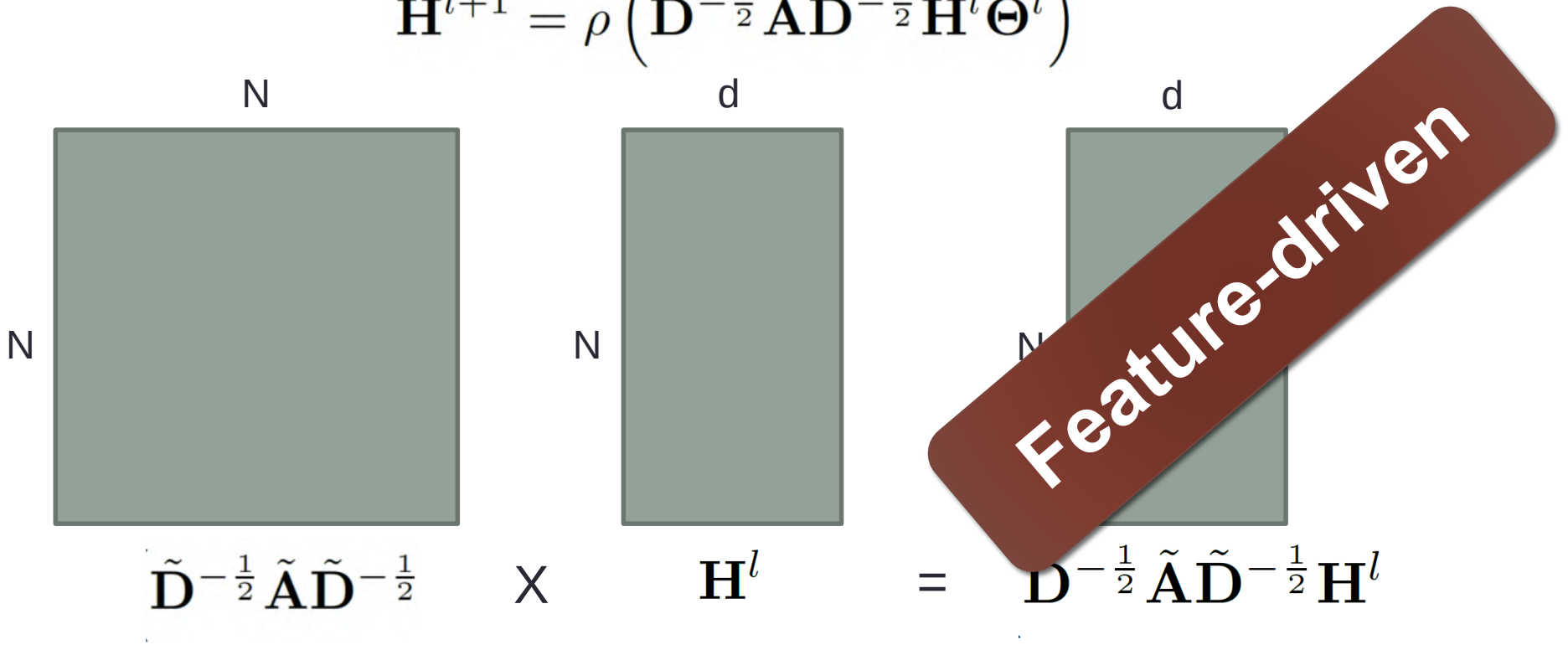
Topology-driven

- Non-transitivity
- [illegible]etric [illegible]tivity
- [illegible]ncertainty
- Dynamic
- Heterogeneity
- Interpretability

The intrinsic problem GNN is solving

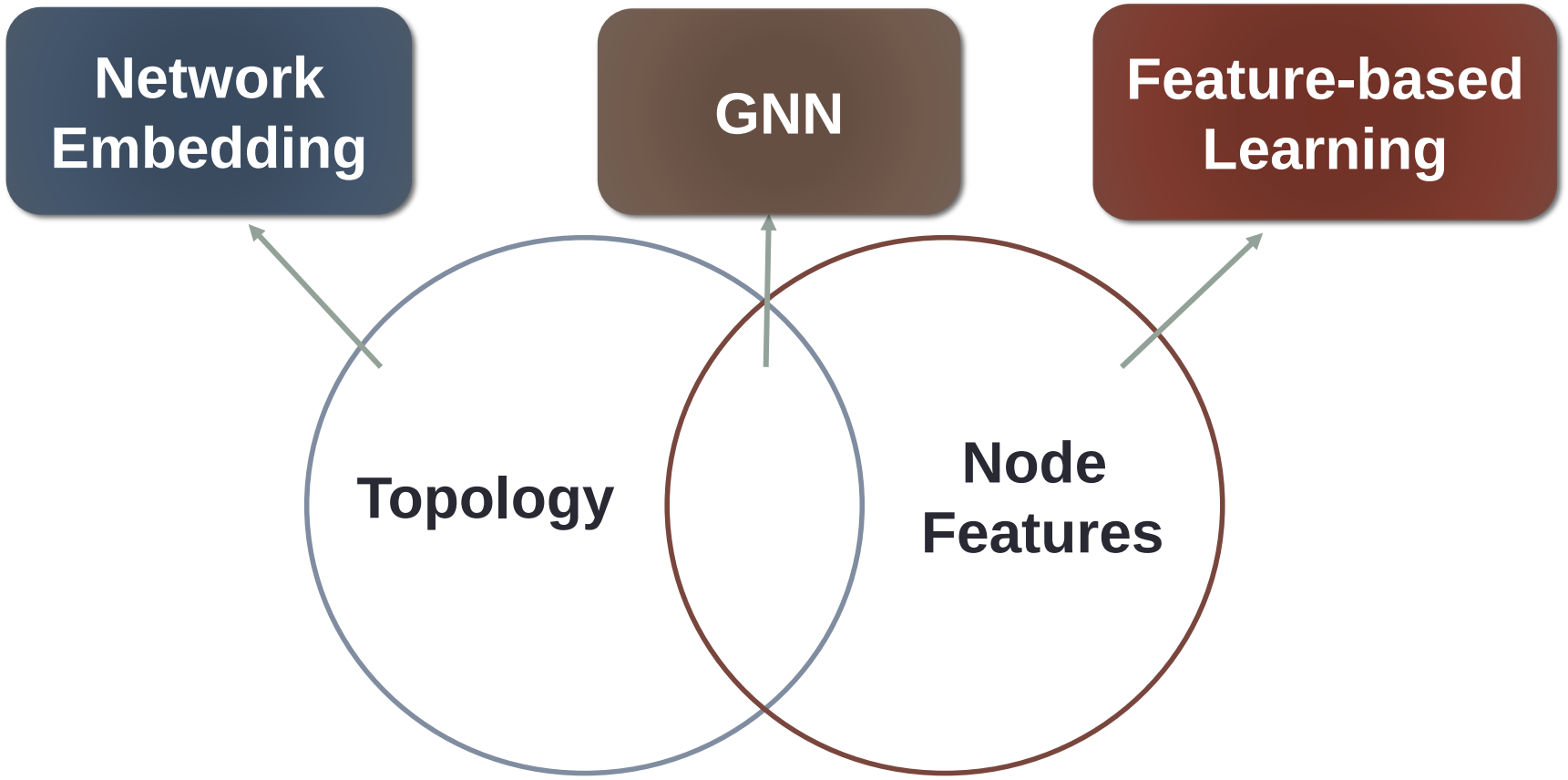
Fusing topology and features in the way of smoothing features with the assistance of topology.

$$\mathbf{H}^{l+1} = \rho\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{l}\mathbf{\Theta}^{l}\right)$$



Network Embedding vs. GNN

There is no better one, but there is more proper one.

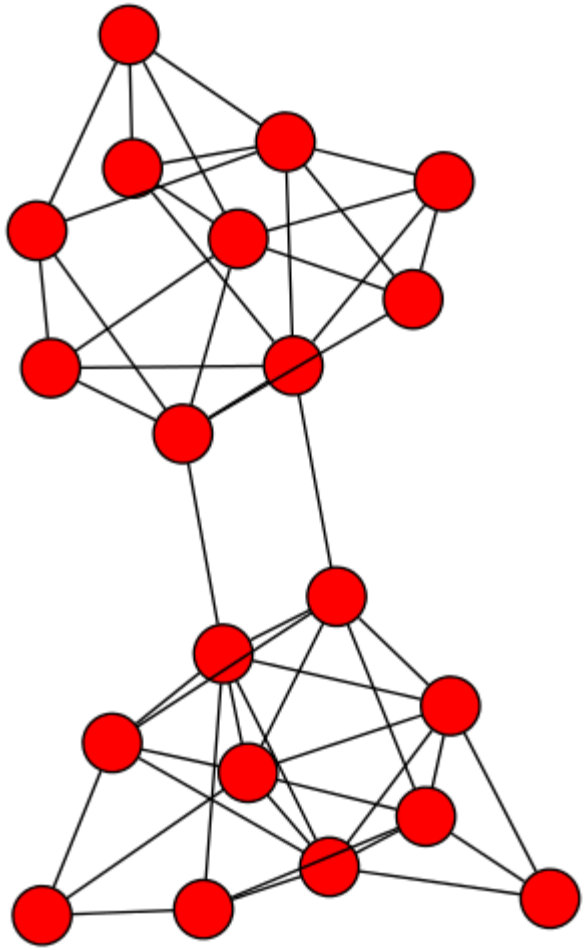


Learning from networks

Network Embedding

Network Embedding

G = (V, E)

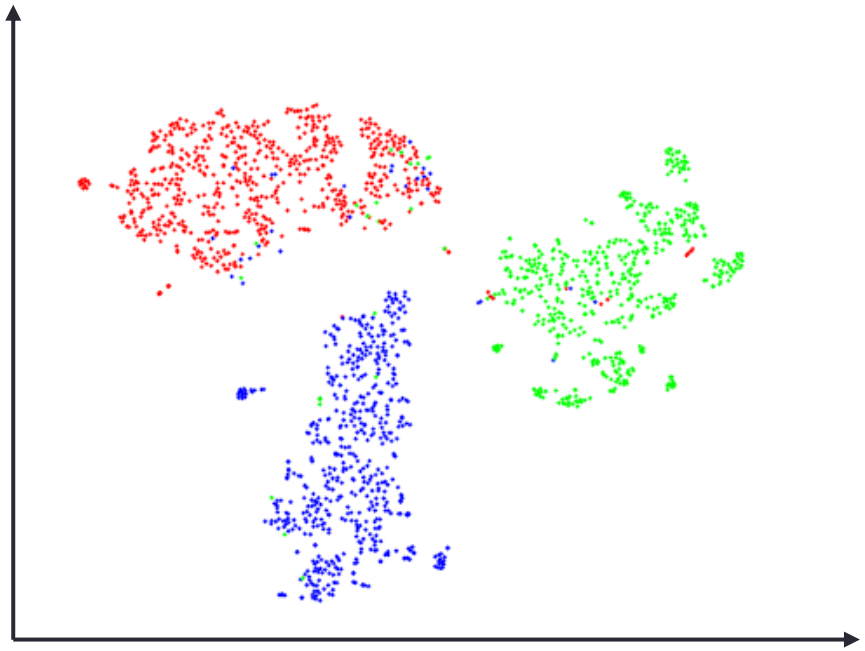


generate

embed

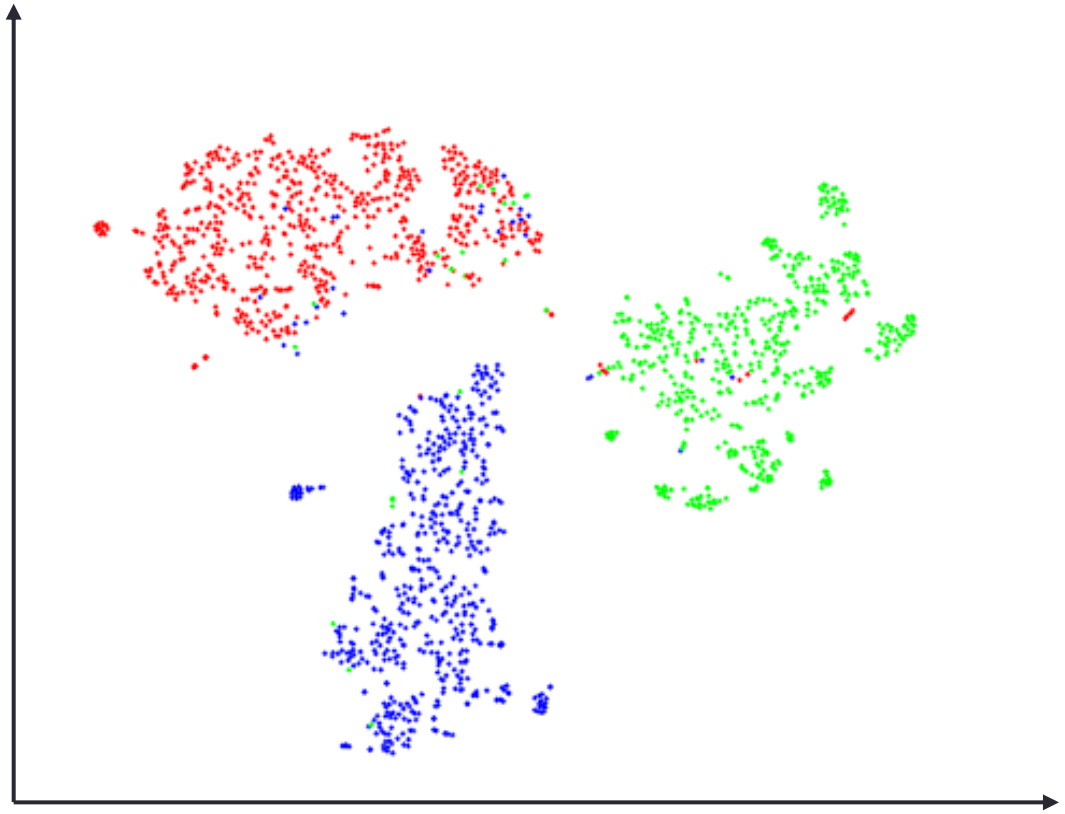
G = (V)

Vector Space



- Easy to parallel
- Can apply classical ML methods

The ultimate goal



Network Inference

- Node importance
- Community detection
- Network distance
- Link prediction
- Node classification
- Network evolution
- …

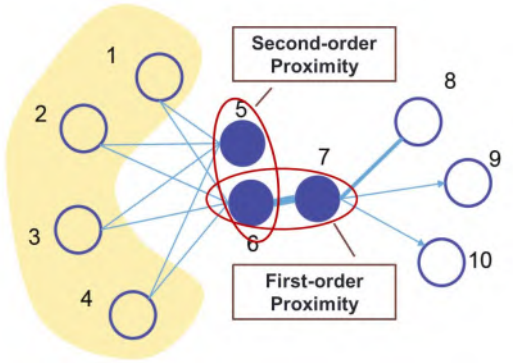
in *Vector Space*

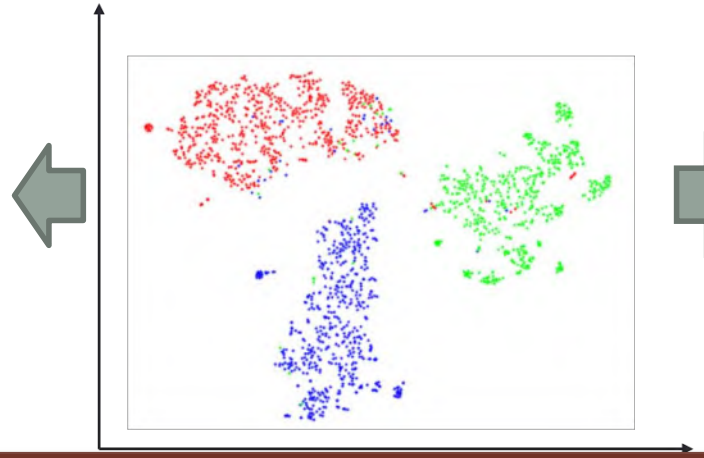
The goal of network embedding

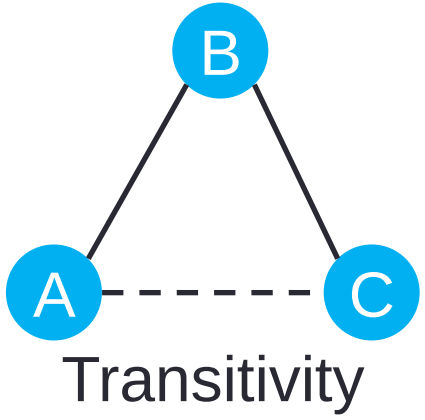
Goal Support network inference in vector space

Reflect network structure

Maintain network properties







Transform network nodes into vectors that are fit for off-the-shelf machine learning models

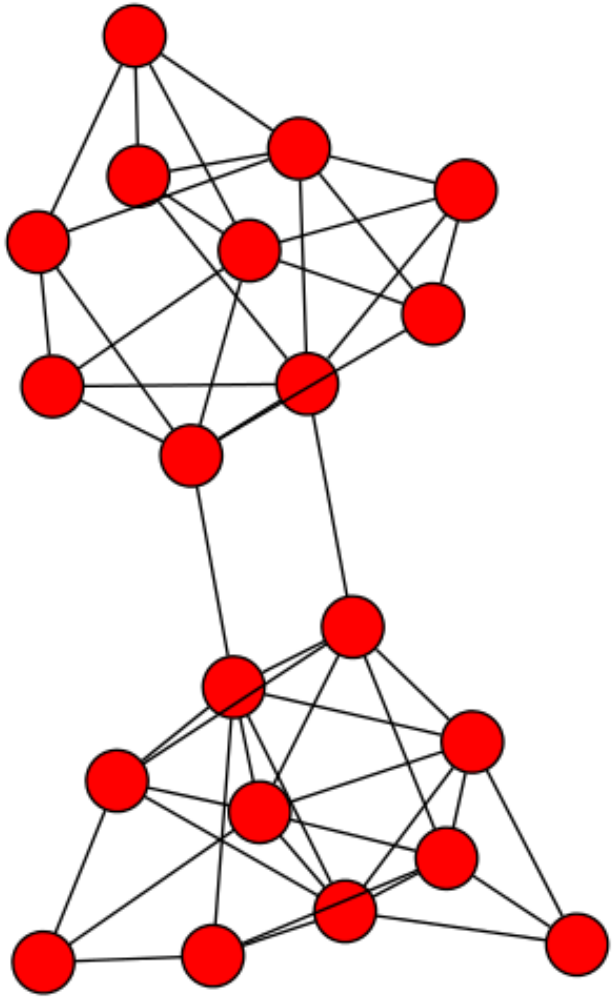
Outline

- **Structure-preserved network embedding**
- **Property-preserved network embedding**
- **Dynamic network embedding**

Outline

- **Structure-preserved network embedding**
- **Property-preserved network embedding**
- **Dynamic network embedding**

Network Structures



Nodes & Links

Pair-wise Proximity

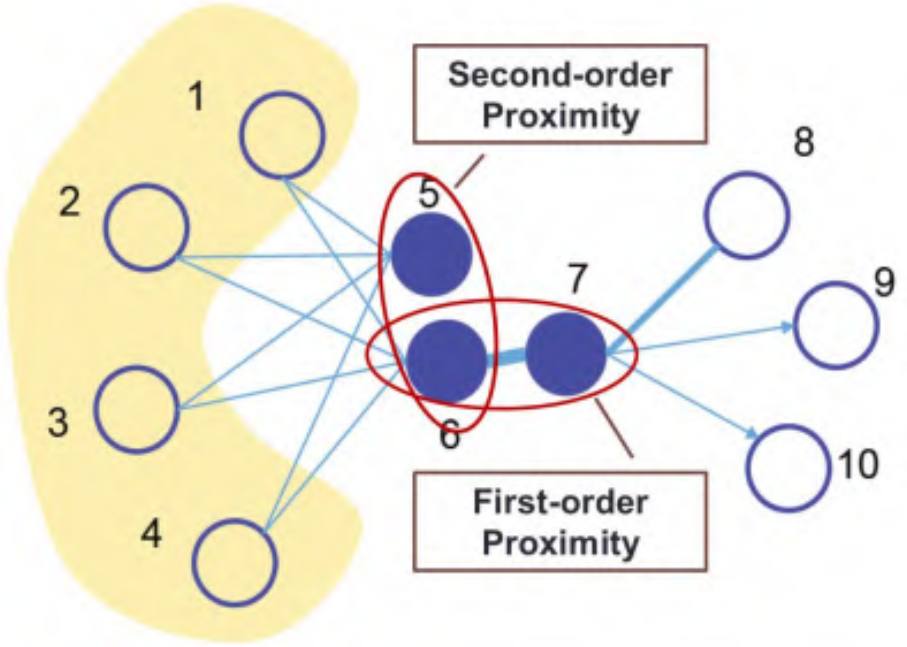
Community Structures

Hyper Edges

Global Structure

High-Order Proximity

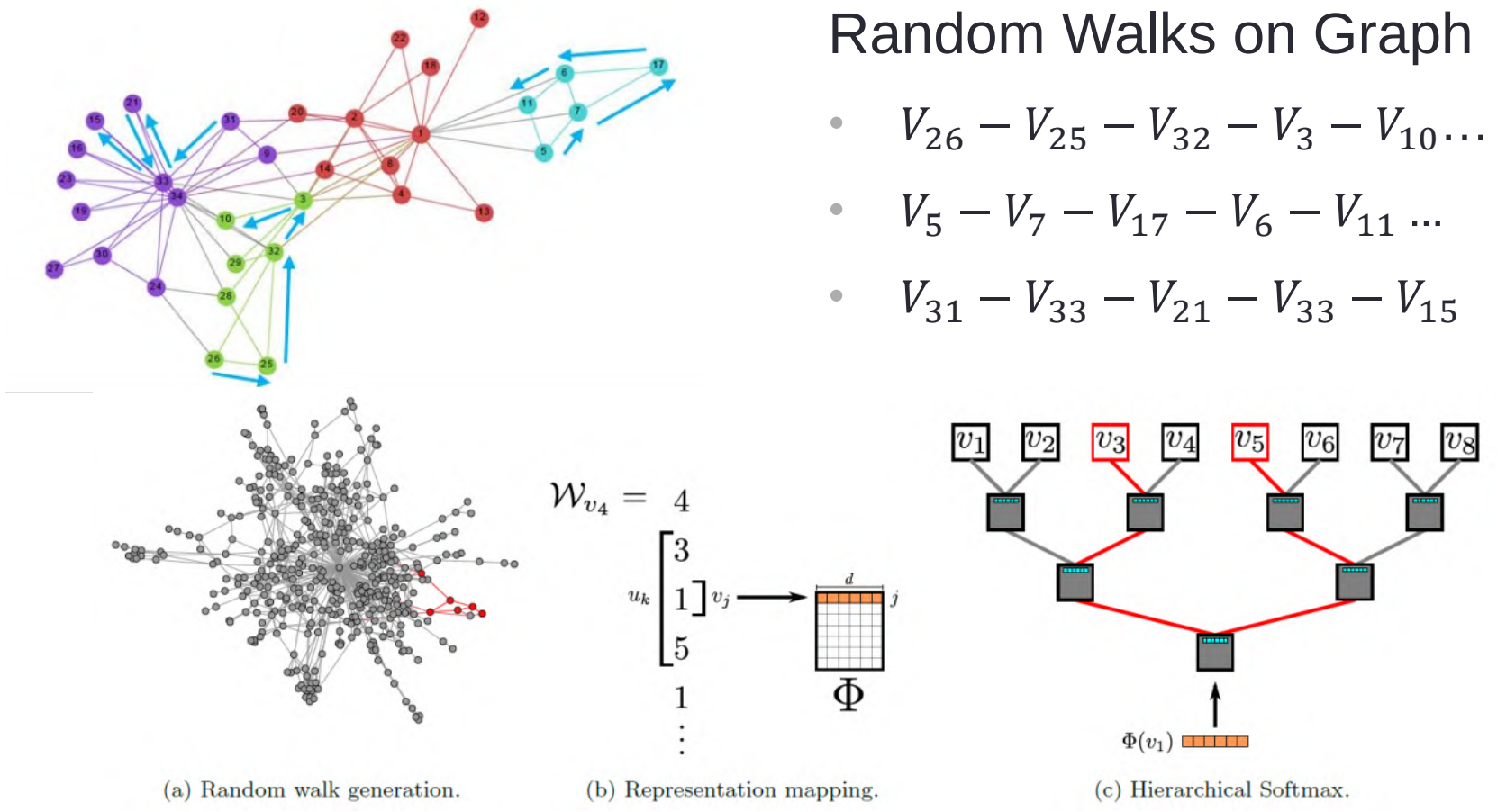
- Capturing the underlying structure of networks



- Advantages:
  - Solve the sparsity problem of network connections
  - Measure indirect relationship between nodes

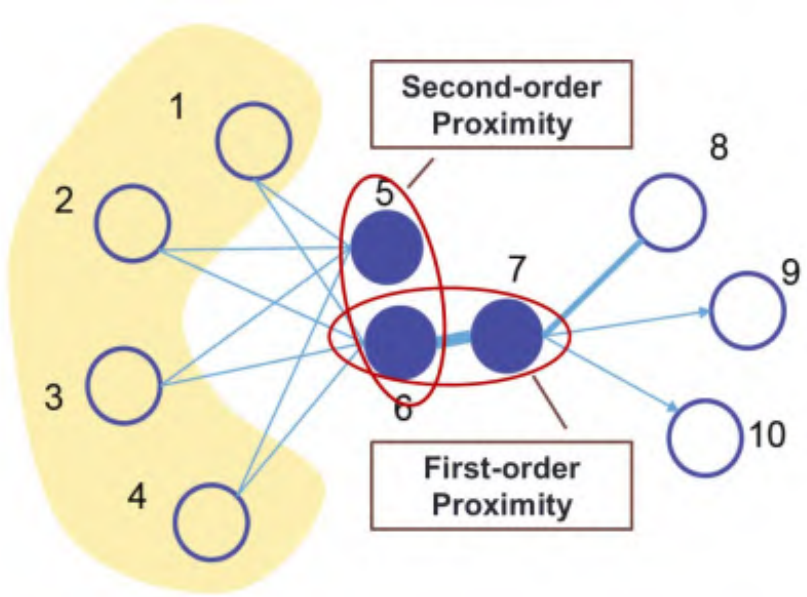
DeepWalk

- Exploit truncated random walks to define neighborhoods of a node.



B. Perozzi et al. Deepwalk: Online learning of social representations. *KDD 2014.*

LINE



LINE with First-order Proximity: local pairwise

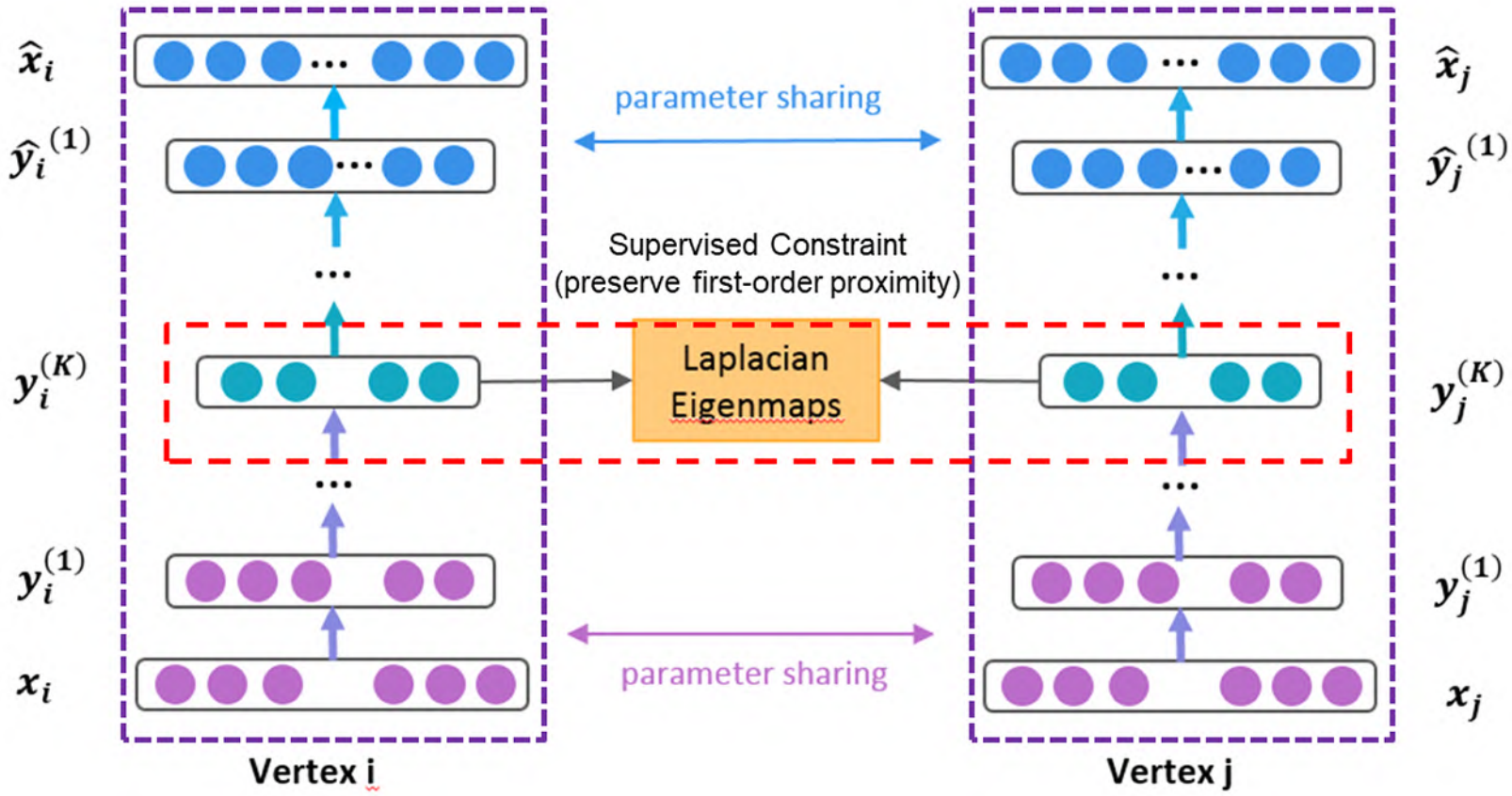
$$O_1 = -\sum_{(i,j)\in E} w_{ij} \log p_1(v_i, v_j)$$

LINE with Second-order Proximity: neighborhood structures

$$O_2 = \sum_{i\in V} \lambda_i d(\hat{p}_2(\cdot|v_i), p_2(\cdot|v_i))$$

Jian Tang et al. LINE: Large-scale Information Network Embedding. *WWW 2015*.

SDNE – Structural Deep Network Embedding

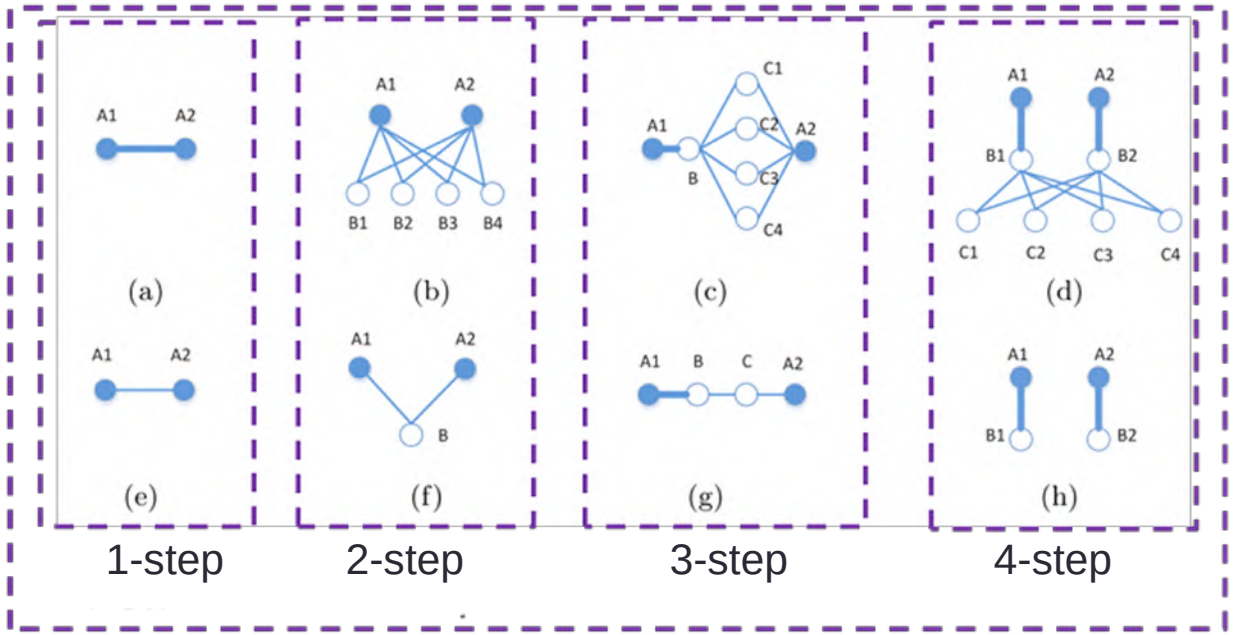


Unsupervised Autoencoder
(preserve second-order proximity)

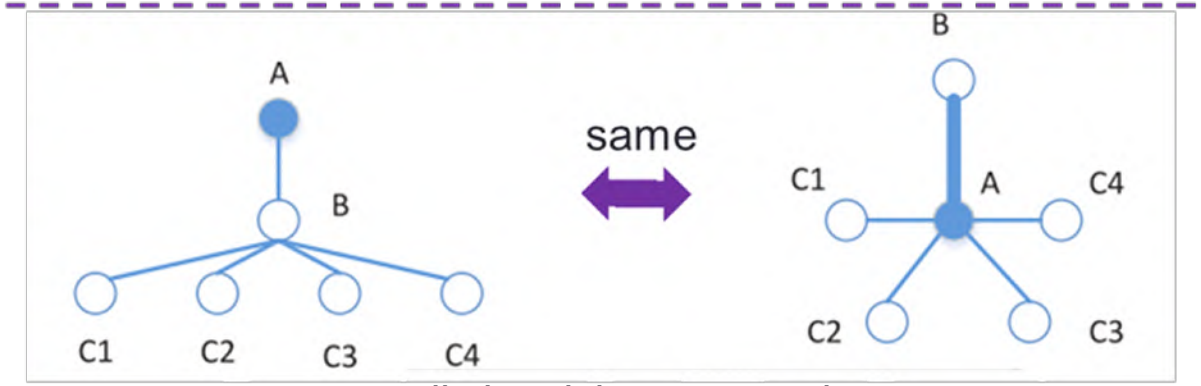
Unsupervised Autoencoder
(preserve second-order proximity)

Daixin Wang et al. Structural Deep Network Embedding. *KDD*, 2016.

GraRep



capturing different k-step information

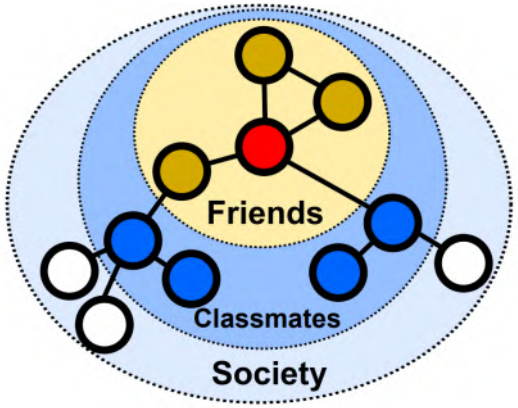


maintaining different k-step information separately

Shaosheng Cao et al. GraRep: Learning Graph Representations with Global Structural Information. *CIKM 2015*.

What is the *right* order?

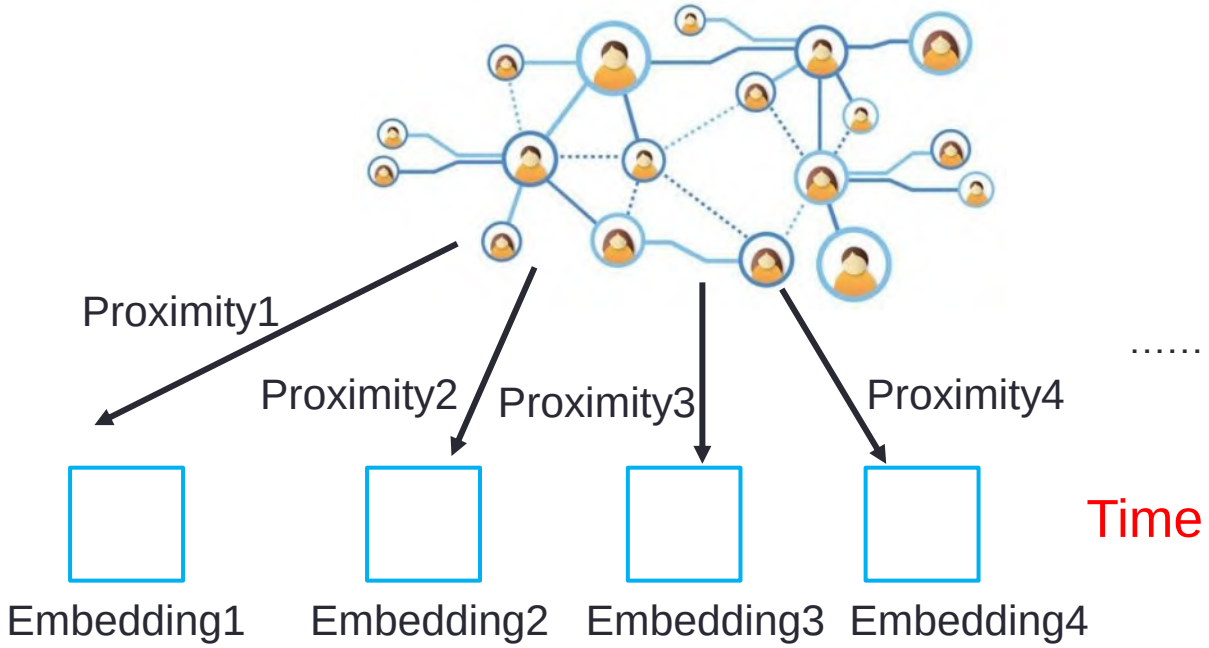
- Different networks/tasks require different high-order proximities
  - E.g., multi-scale classification (Bryan Perozzi, et al, 2017)



  - E.g., networks with different scales and sparsity
- Proximities of different orders can also be arbitrarily weighted
  - E.g., equal weights, exponentially decayed weights (Katz)

What is the *right* order?

- Existing methods can only preserve one fixed high-order proximity
  - Different high-order proximities are calculated separately



→ How to preserve arbitrary-order proximity while guaranteeing accuracy and efficiency?

Problem Formulation

- High-order proximity: a polynomial function of the adjacency matrix

$$S = f(A) = w_1 A^1 + w_2 A^2 + \cdots + w_q A^q$$

  - q: order; $w_1 \ldots w_q$: weights, assuming to be non-negative
  - A: could be replaced by other variations (such as the Laplacian matrix)
- Objective function: matrix factorization

$$\min_{U^*, V^*} \left\| S - U^* {V^*}^T \right\|_F^2$$

  - $U^*, V^* \in \mathbb{R}^{N \times d}$: left/right embedding vectors
  - d: dimensionality of the space
- Optimal solution: Singular Value Decomposition (SVD)
  - $[U, \Sigma, V]$: top-d SVD results

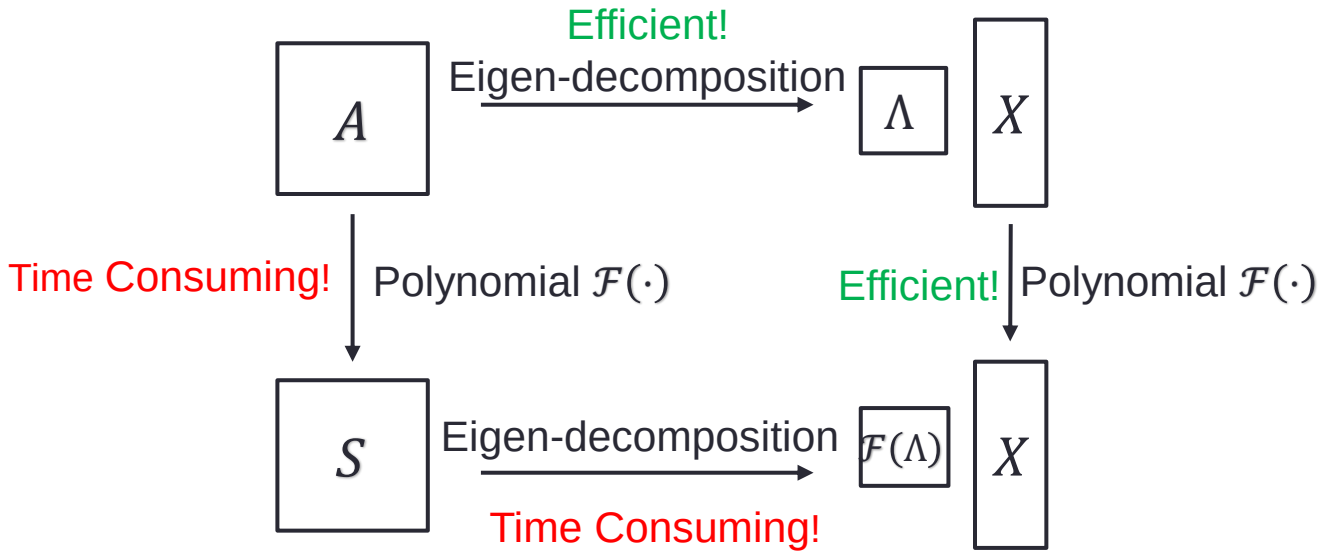
$$U^* = U\sqrt{\Sigma}, \qquad V^* = V\sqrt{\Sigma}$$

Ziwei Zhang, et al. Arbitrary-Order Proximity Preserved Network Embedding. *KDD, 2018.*

Eigen-decomposition Reweighting

- Eigen-decomposition reweighting

THEOREM 4.2 (EIGEN-DECOMPOSITION REWEIGHTING). *If* $[\lambda, \mathbf{x}]$ *is an eigen-pair of* $\mathbf{A}$, *then* $[\mathcal{F}(\lambda), \mathbf{x}]$ *is an eigen-pair of* $\mathbf{S} = \mathcal{F}(\mathbf{A})$.



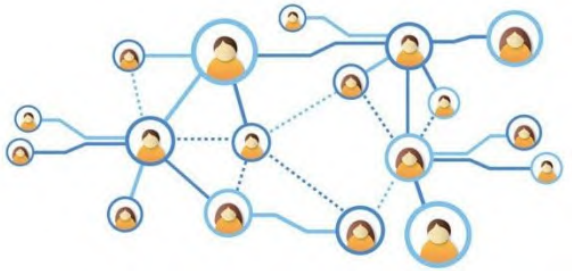
- **Insights**: high-order proximity is simply re-weighting dimensions!

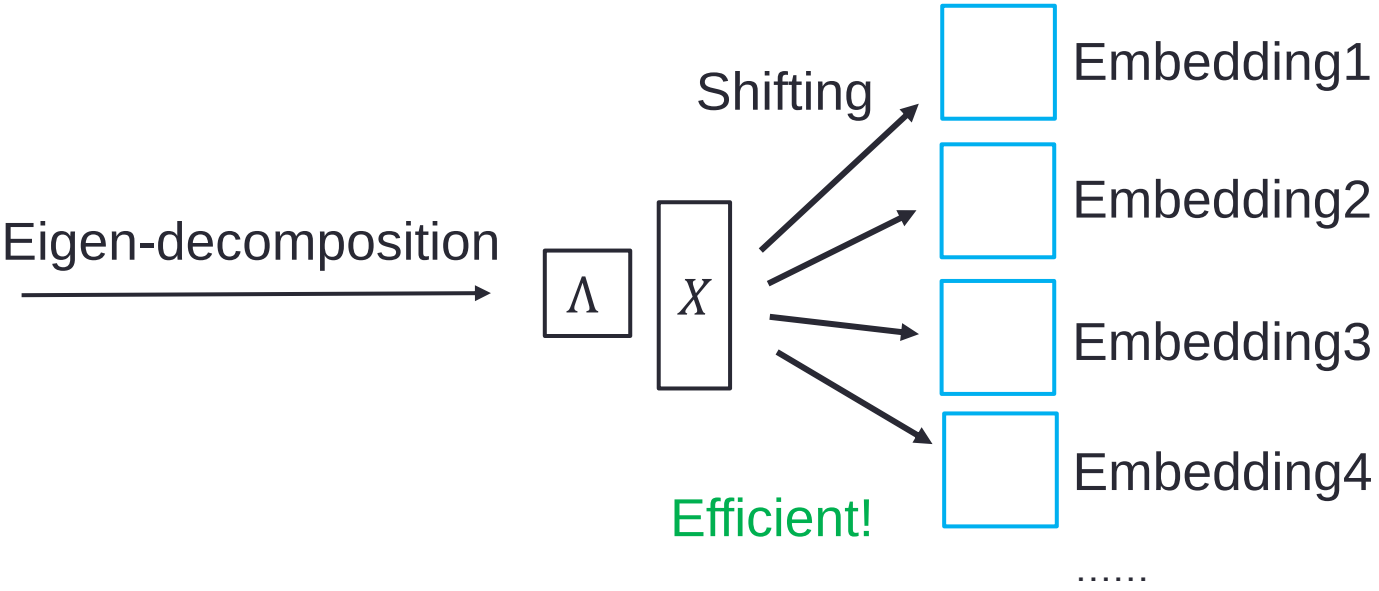
$$U^* = U\sqrt{\Sigma}, V^* = V\sqrt{\Sigma}$$

Ziwei Zhang, et al. Arbitrary-Order Proximity Preserved Network Embedding. *KDD, 2018.*

Preserving Arbitrary-Order Proximity

- Shifting across different orders/weights:



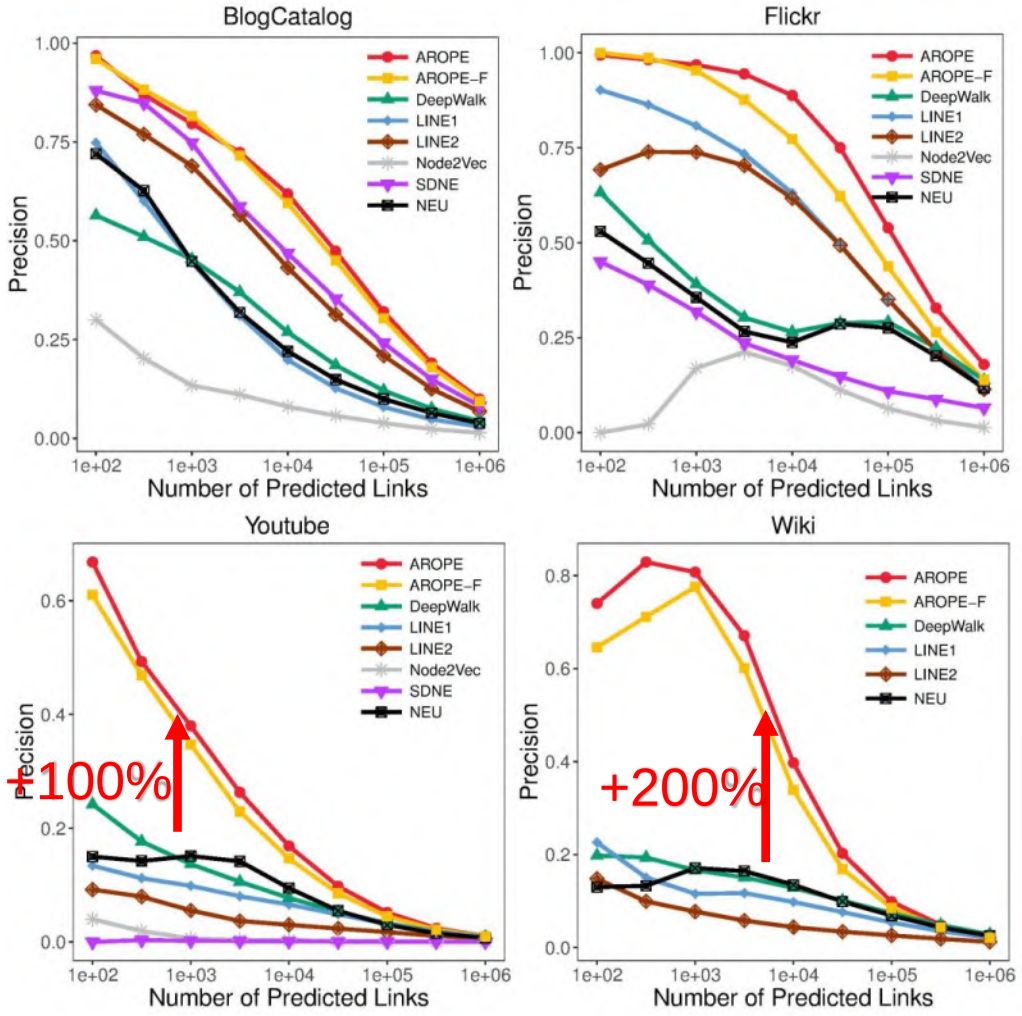


- Preserving arbitrary-order proximity
- Low marginal cost
- Accurate and efficient

Ziwei Zhang, et al. Arbitrary-Order Proximity Preserved Network Embedding. *KDD, 2018.*

Experimental Results

- Link Prediction



Ziwei Zhang, et al. Arbitrary-Order Proximity Preserved Network Embedding. *KDD, 2018.*

Billion-Scale Networks

- Existing network embeddings (e.g., AROPE) can handle million-scale networks
- But real graphs can have billions of nodes and edges
  - Social Networks
    - WeChat: 1 billion monthly active users (March, 2018)
    - Facebook: 2 billion active users (2017)
  - E-commerce Networks
    - Amazon: 353 million products, 310 million users, 5 billion orders (2017)
  - Citation Networks
    - 130 million authors, 233 million publications, 754 million citations (Aminer, 2018)

How to scale embedding methods to billion-scale networks?

Random Projection for Matrix Factorization

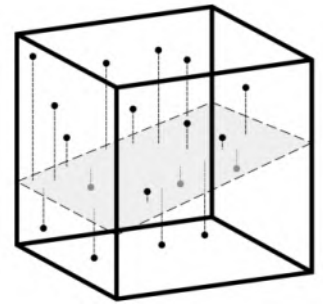
- Objective function: matrix factorization of preserving high-order proximity

$$\min_{U,V} \|S - UV^T\|_p^2$$

$$S = f(A) = \alpha_1 A^1 + \alpha_2 A^2 + \cdots + \alpha_q A^q$$

- Essentially a dimensionality reduction problem
- Random projection: optimization-free for dimensionality reduction
  - Basic idea: randomly project data into a low-dimensional subspace
  - Extremely efficient and friendly to distributed computing
  - Denote $R \in \mathbb{R}^{N \times d}$ as a Gaussian random matrix

$$R_{ij} \sim \mathcal{N}\left(0, \frac{1}{d}\right)$$



  - Surprisingly simple result:

$$U = SR$$

Ziwei Zhang, et al. Billion-scale Network Embedding with Iterative Random Projection. *ICDM, 2018.*

Theoretical Guarantee

- Slight modification: assuming positive semi-definite and using 2-norm

$$\min_{U} \|SS^T - UU^T\|_2$$

$$S = f(A) = \alpha_1 A^1 + \alpha_2 A^2 + \cdots + \alpha_q A^q$$

- Theoretical guarantee

Theorem 1. *For any similarity matrix* $\mathbf{S}$*, denote its rank as* $r_{\mathbf{S}}$*. Then, for any* $\epsilon \in \left(0, \frac{1}{2}\right)$*, the following equation holds:*

$$P\left[\left\|\mathbf{S}\cdot\mathbf{S}^T - \mathbf{U}\cdot\mathbf{U}^T\right\|_2 > \epsilon\left\|\mathbf{S}^T\cdot\mathbf{S}\right\|_2\right] \leq 2r_{\mathbf{S}}e^{-\frac{\left(\epsilon^2-\epsilon^3\right)d}{4}},$$

where $\mathbf{U} = \mathbf{S}\cdot\mathbf{R}$ *and* $\mathbf{R}$ *is a Gaussian random matrix.*

- Basically, random projection can effectively minimize the objective function
- However, calculating S is still very time consuming

Ziwei Zhang, et al. Billion-scale Network Embedding with Iterative Random Projection. *ICDM, 2018.*

RandNE: Iterative Projection

- Iterative projection: can be calculated iteratively

$$U = SR = \left(\alpha_1 A^1 + \alpha_2 A^2 + \cdots + \alpha_q A^q\right) R$$
$$= \alpha_1 A^1 R + \alpha_2 A^2 R + \cdots + \alpha_q A^q R$$

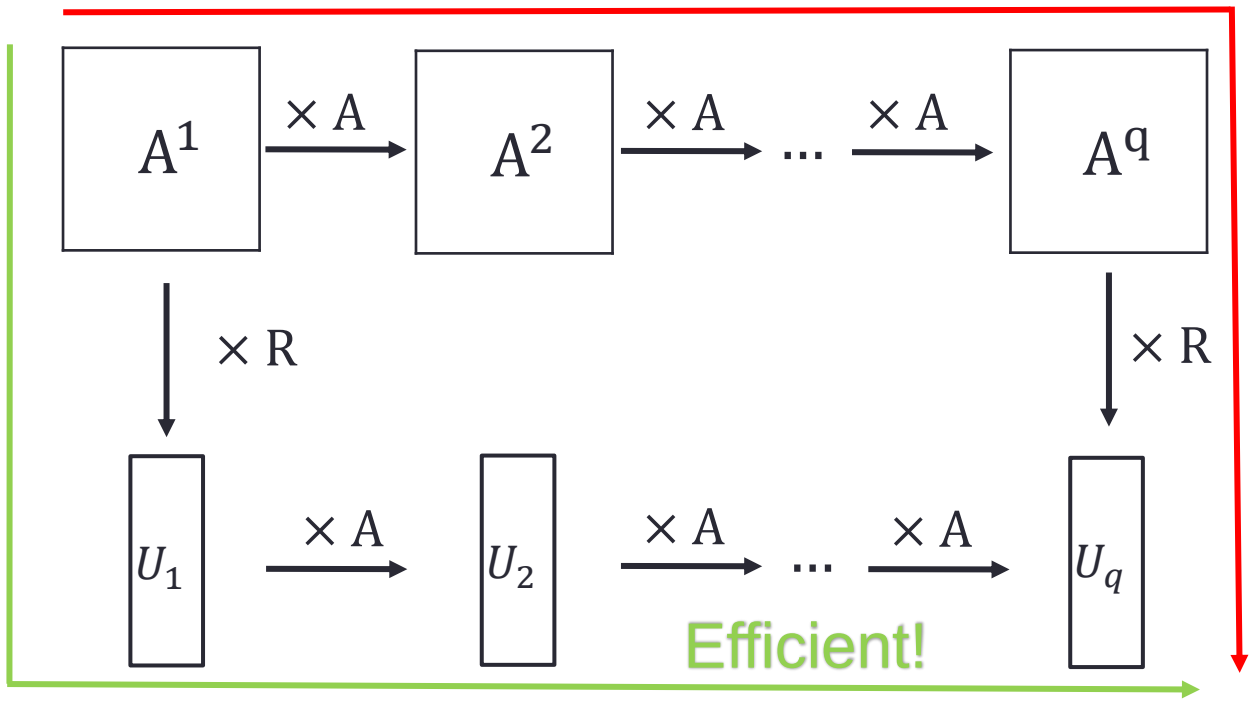
$\times A$ $\times A$ $\times A$

- Why efficient?
  - A: $N \times N$ sparse adjacency matrix
  - R: $N \times d$ low-dimensional matrix
  - Associative law of matrix multiplication

Sparse
$AA \ldots AAAR$ ← Low-dimensional
Sparse
$AA \ldots AA(AR)$ ← Low-dimensional
Sparse
$AA \ldots A(AAR)$ ← Low-dimensional

Sparse matrix multiplication!

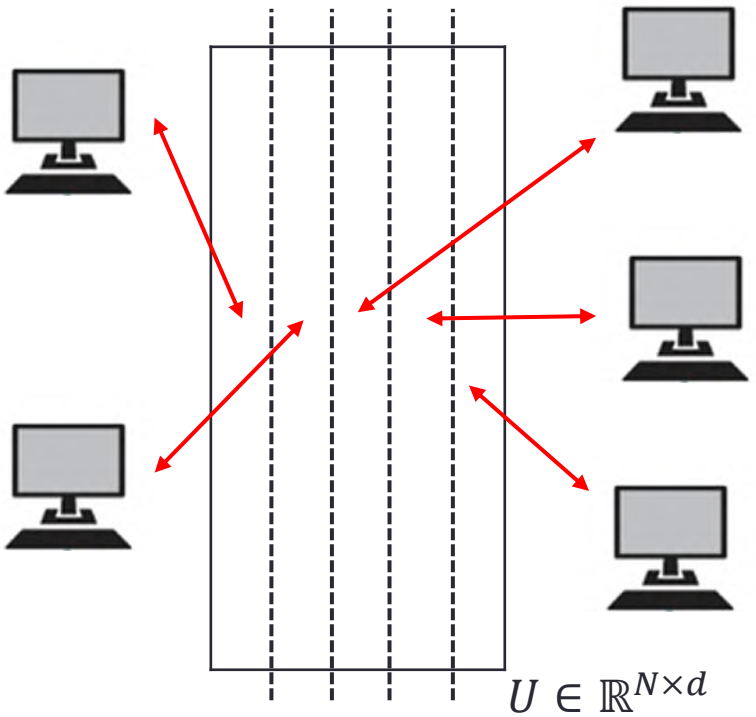
Time Consuming!



Ziwei Zhang, et al. Billion-scale Network Embedding with Iterative Random Projection. *ICDM, 2018.*

Distributed Calculation

- Iterative random projection only involves matrix multiplication $U_i = AU_{i-1}$
  - Each dimension can be calculated separately
    - Property of sparse matrix multiplication
  - No communication is needed during calculation!



Algorithm 2 Distributed Calculation of RandNE

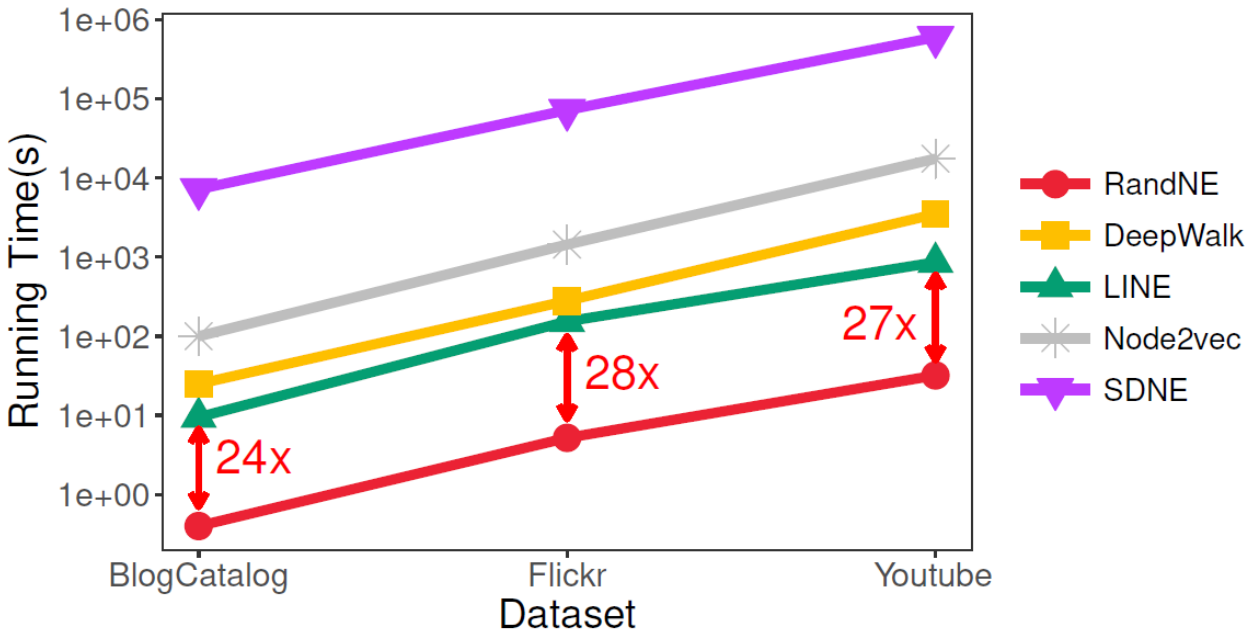
Require: Adjacency matrix $\mathbf{A}$, Initial Projection $\mathbf{U}_0$, Parameters of RandNE, K Distributed Servers
Ensure: Embedding Results $\mathbf{U}$

```
Broadcast A, U_0 and parameters into K servers
Set i = 1
repeat
    if There is an idle server k then
        Calculate U(i, :) in server k
        i = i + 1
        Gather U(i, :) from server k after calculation
    end if
until i > d
Return U
```

Ziwei Zhang, et al. Billion-scale Network Embedding with Iterative Random Projection. *ICDM, 2018.*

Experimental Results

Running time



At least dozens of times faster

Link Prediction

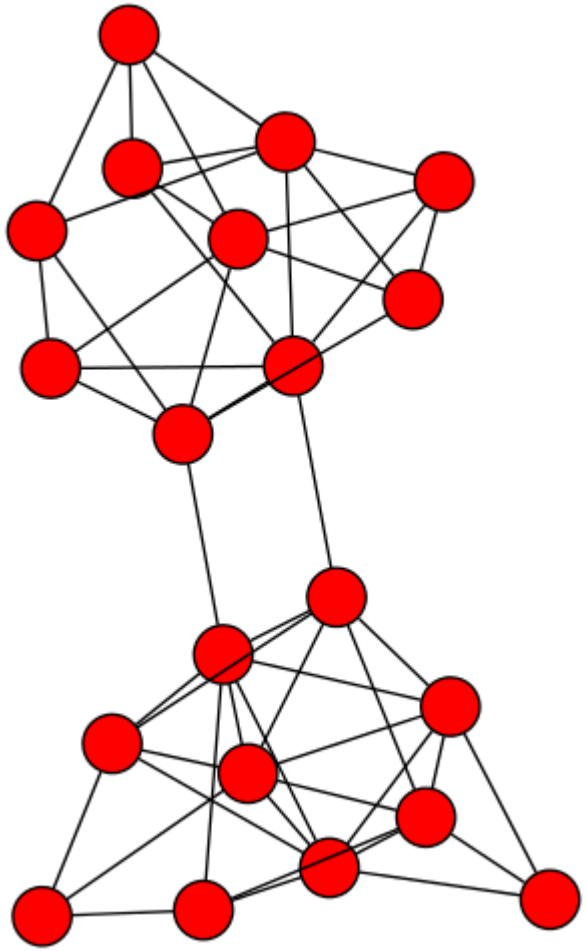
AUC SCORES OF LINK PREDICTION.

Dataset	BlogCatalog	Flickr	Youtube
RandNE	**0.944**	**0.940**	0.887
DeepWalk	0.760	0.938	0.909
LINE_{1st}	0.667	0.909	0.847
LINE_{2nd}	0.762	0.932	**0.959**
Node2vec	0.650	0.865	0.778
SDNE	0.940	0.926	-

Superior or comparable performance

Ziwei Zhang, et al. Billion-scale Network Embedding with Iterative Random Projection. *ICDM, 2018.*

Section Summary



Nodes & Links

↓

Pair-wise Proximity

↓

Community Structures

↓

Hyper Edges

↓

Global Structure

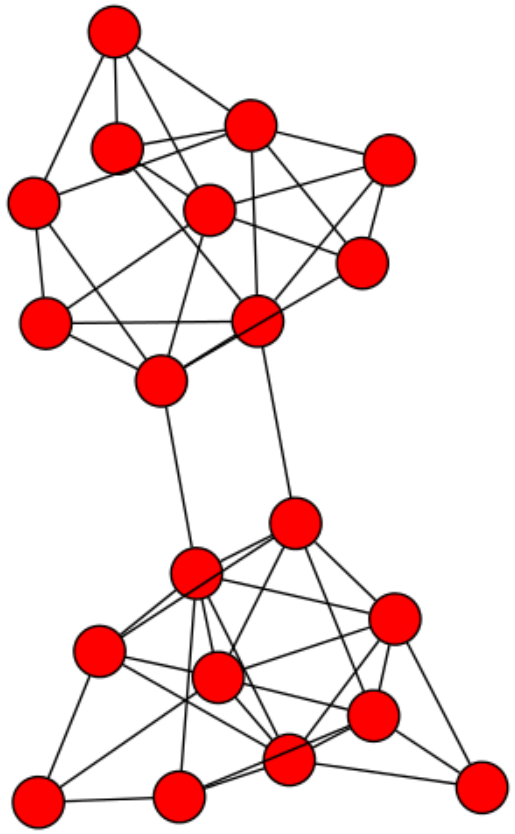
Network Characteristics

Application Characteristics

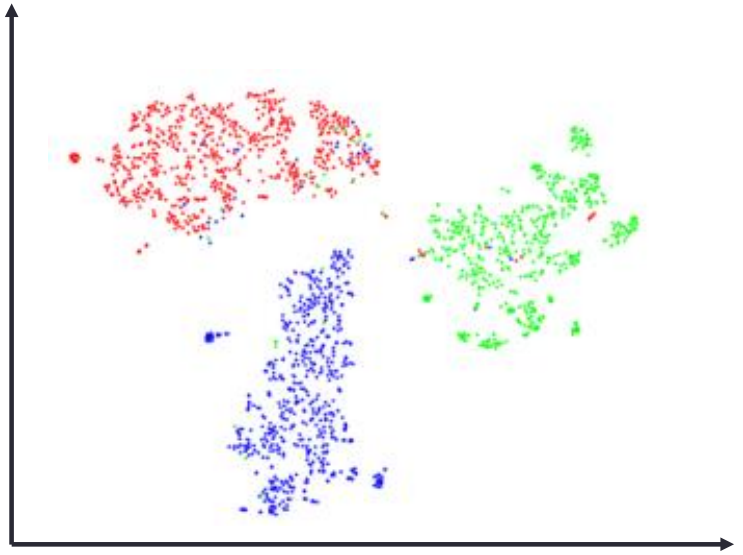
Outline

- Structure-preserved network embedding
- Property-preserved network embedding
- Dynamic network embedding

Why preserve network properties?



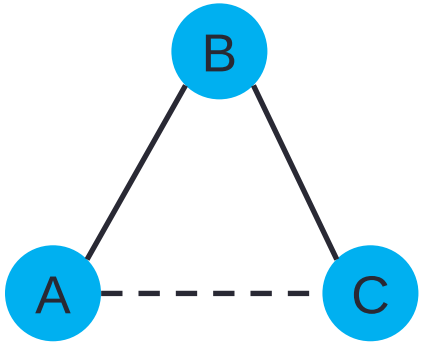
Heterogeneity



Transitivity

The Transitivity Phenomenon

Network



Embedding Space

Triangle Inequality: $D(A,B) + D(B,C) > D(A,C)$

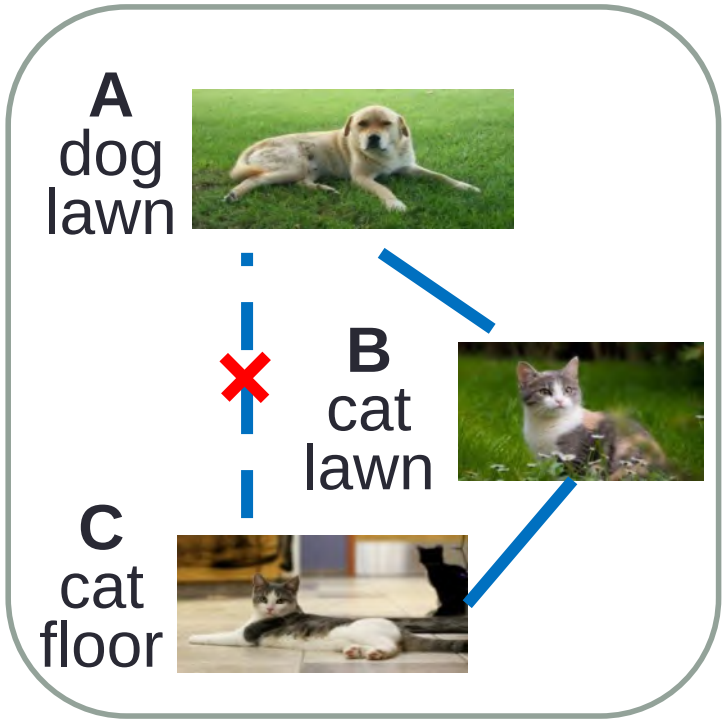
A close to ***B***, ***B*** close to ***C*** $\longrightarrow$ ***A*** relatively close to ***C***

However, real network data is complex…

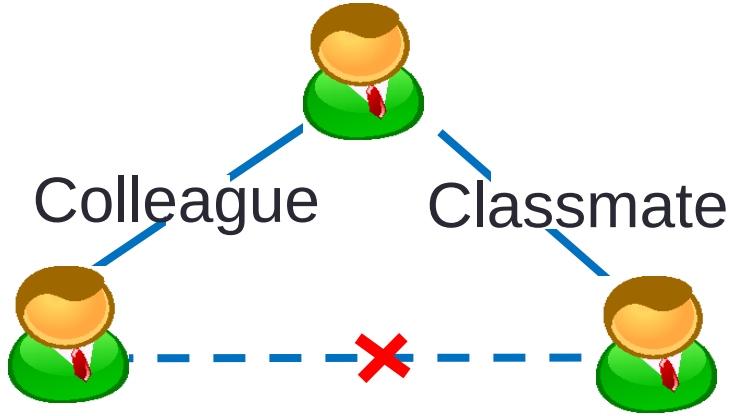
Non-transitivity

The Co-existence of *Transitivity* and *Non-transitivity*

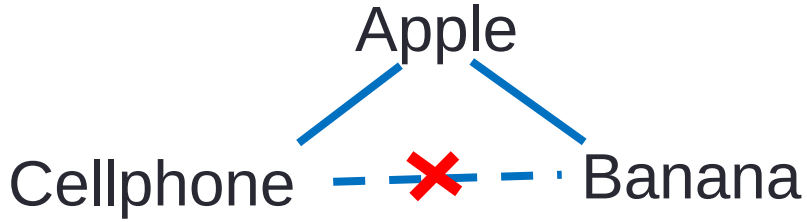
Image network



Social network



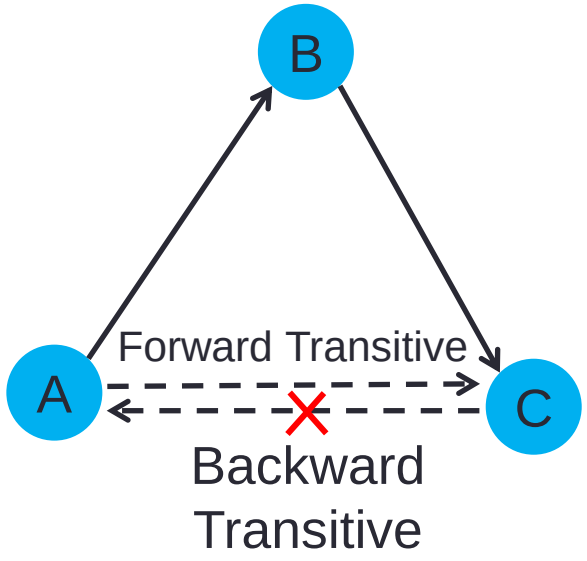
Word network



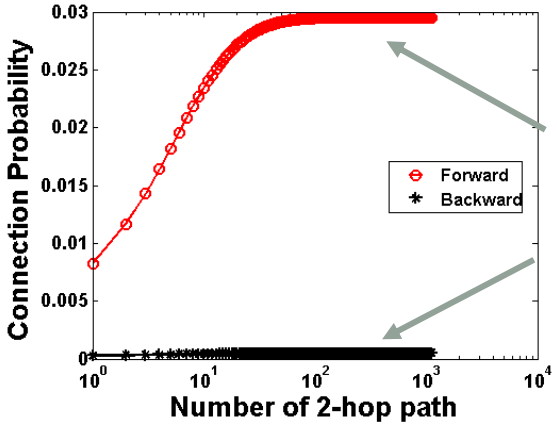
How to incorporate non-transitivity in the embedding space?

Asymmetric Transitivity

Directed Network



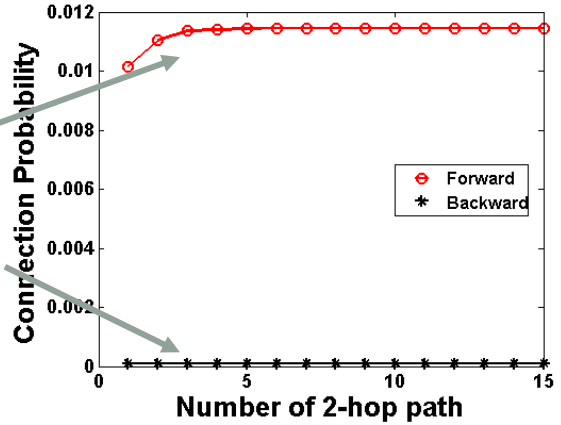
A→B, B → C => A → C, but not C →A



Tencent Microblog

Forward

Backward



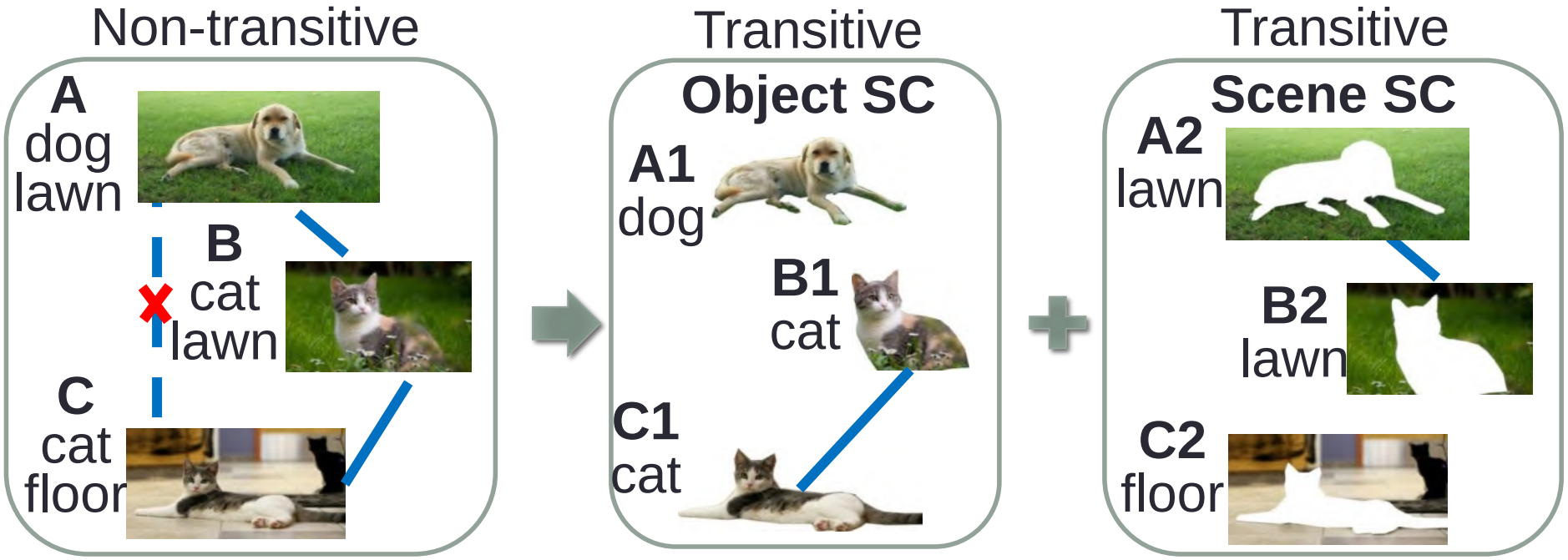
Twitter

Distance metric in the embedding space is symmetric.
How to incorporate *Asymmetric Transitivity*?

Non-transitivity

The source of non-transitivity:

Each node has multiple similarity components

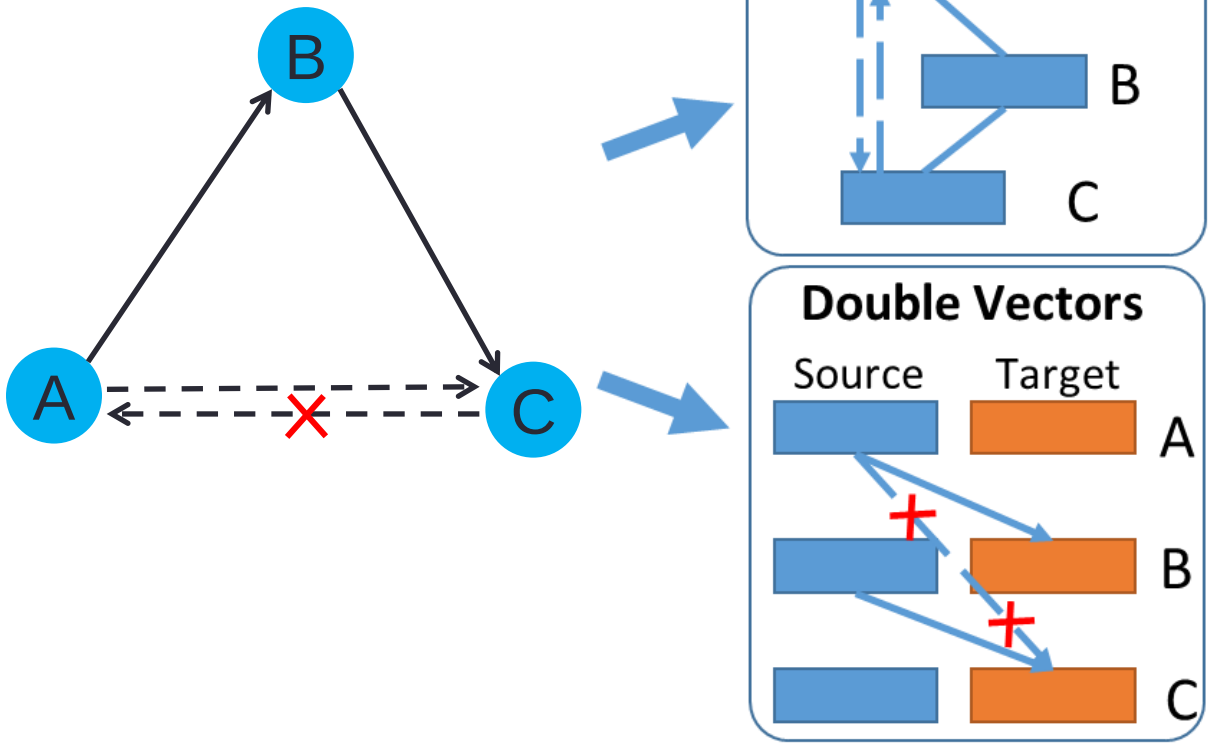


Non-transitive Embedding: represent non-transitive data with multiple latent similarity components

M. Ou, et al. Non-transitive Hashing with Latent Similarity Components. *KDD*, 2015.

Asymmetric Transitivity

All existing methods fail..



Asymmetric fails

Transitivity fails

Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, Wenwu Zhu. Asymmetric Transitivity Preserving Graph Embedding. *KDD*, 2016.

Section Summary

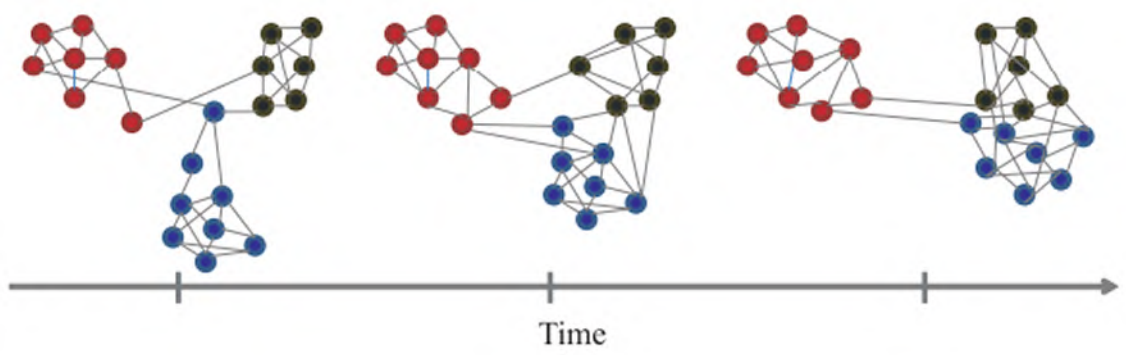
- Compared with network structures, **network properties** have a large space to explore in network embedding
- Transitivity is important for network inference.
- Uncertainty provides evidence in making network inference.
- Many other property issues:
  - The right embedding space: Euclidean space?
  - Power-law distribution
  - …

Outline

- Structure-preserved network embedding
- Property-preserved network embedding
- Dynamic network embedding

Dynamic Networks

- Networks are dynamic in nature
  - New (old) nodes are added (deleted)
    - New users, products, etc.
  - The edges between nodes evolve over time
    - Users add or delete friends in social networks, or neurons establish new connections in brain networks.
- How to efficiently incorporate the dynamic changes when networks evolve?

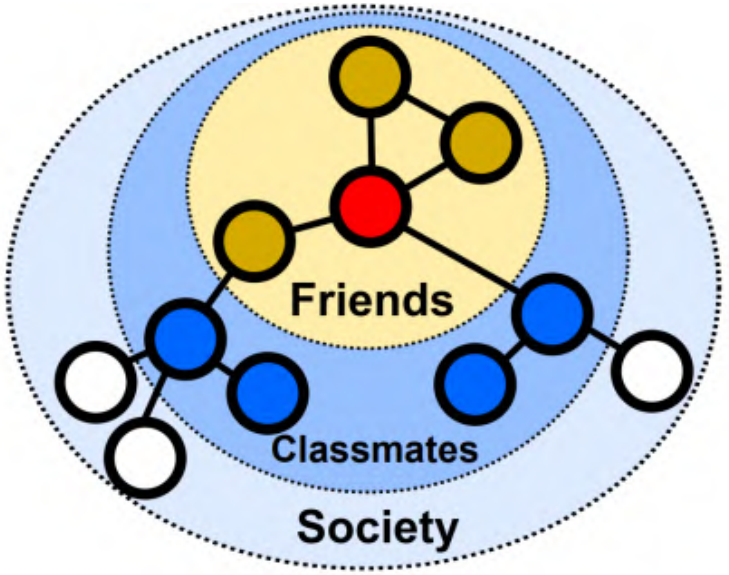


Key problems in dynamic network embedding

- I : Out-of-sample nodes
- II : Incremental edges
- III: Aggregated error
- IV: Scalable optimization

Challenge: High-order Proximity

- **High-order proximity**
  - Critical structural property of networks
  - Measure indirect relationship between nodes
  - Capture the structure of networks with different scales and sparsity



Network Embedding vs. Traditional Graph Embedding

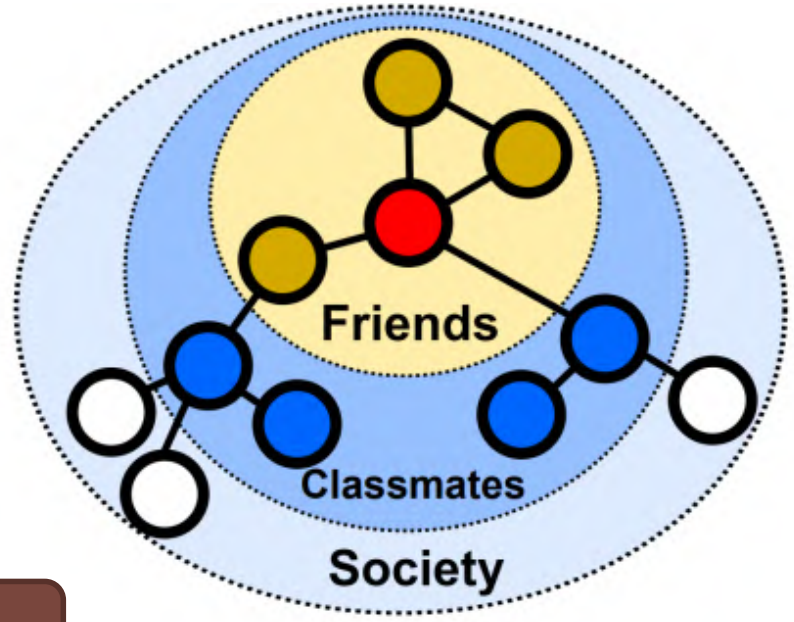
Challenge: High-order Proximity

I : Out-of-sample nodes

II : Incremental edges

III: Aggregated error

IV: Scalable optimization



Preserve High-order Proximities

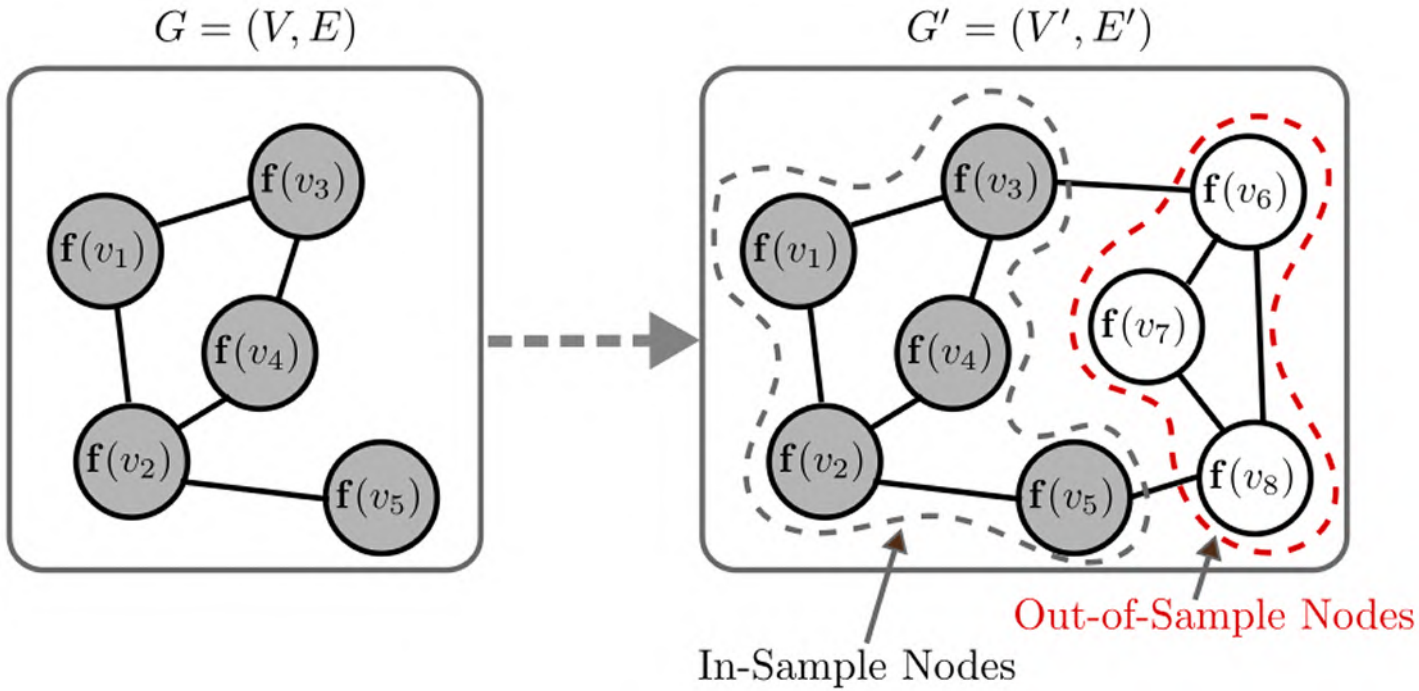
Local Change leads to Global Updating

Key problems in dynamic network embedding

- **I : Out-of-sample nodes**
- **II : Incremental edges**
- **III: Aggregated error**
- **IV: Scalable optimization**

Problem

- To infer embeddings for out-of-sample nodes



- G=(V, E) evolves into G'=(V', E'), where V' = V ∪ V*.
- n old nodes: V = $\{v_1,\ldots,v_n\}$, m new nodes: V* = $\{v_{n+1},\ldots,v_{n+m}\}$
- Network embedding $f: V \rightarrow \mathbb{R}^d$
- We know $f(V)$ for old nodes, want to infer $f(V')$ for new nodes.

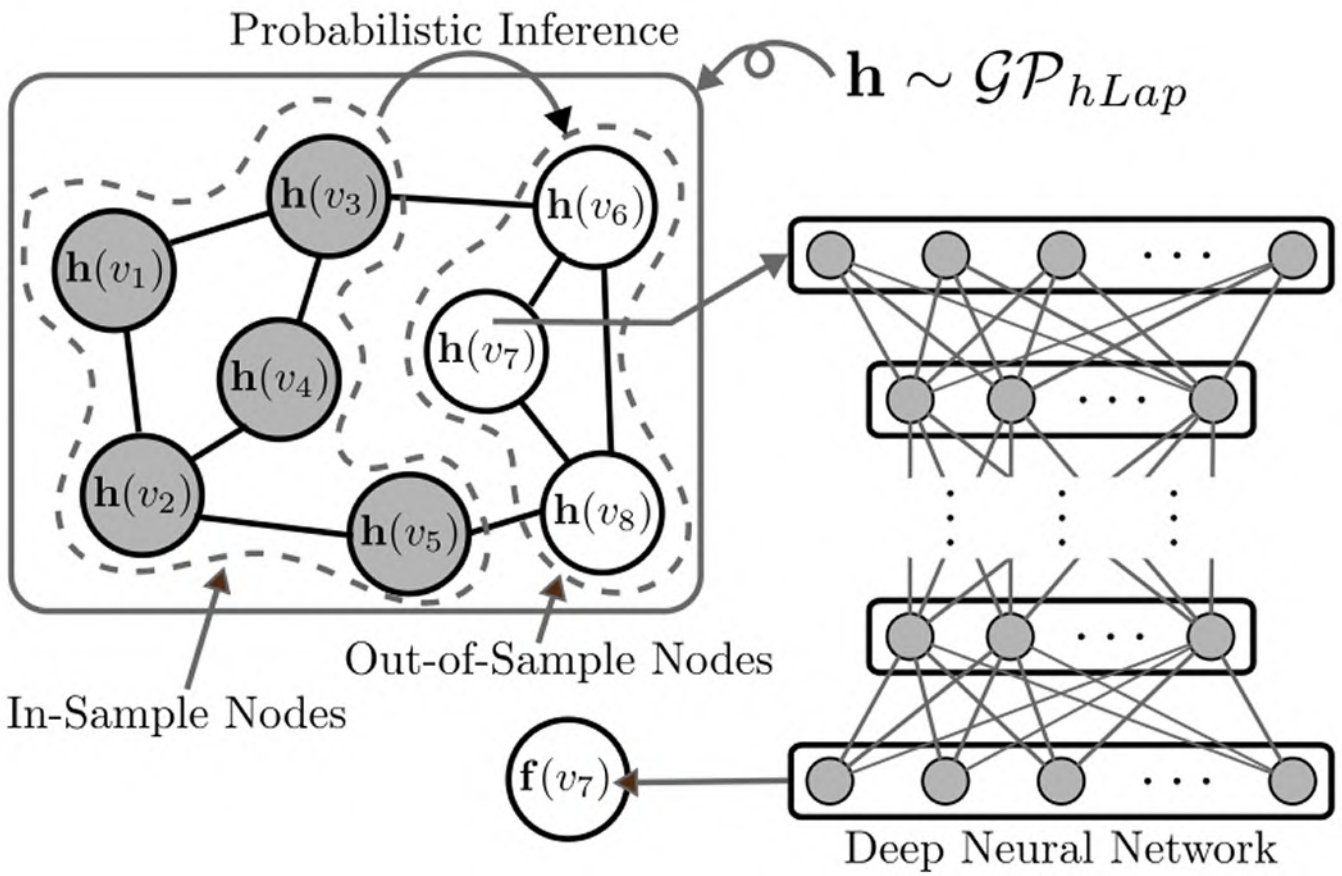
Challenges

- Preserve network structures
  - E.g., high-order proximity
  - Need to incorporate prior knowledge on networks
- Share similar characteristics with in-sample embeddings
  - E.g. magnitude, mean, variance
  - Requires a model with great expressive power to fit the data well
- Low computational cost

Jianxin Ma, et al. DepthLGP: Learning Embeddings of Out-of-Sample Nodes in Dynamic Networks. ***AAAI***, 2018.

DepthLGP

- Nonparametric probabilistic modeling + Deep Learning



Jianxin Ma, et al. DepthLGP: Learning Embeddings of Out-of-Sample Nodes in Dynamic Networks. ***AAAI***, 2018.

DepthLGP

- Design a kernel for the *k*th (*k*=*1,…,s*) dimension of *h*(·)

First-order Proximity

Second-order Proximity

$$\mathbf{K}_k \triangleq \left[\mathbf{I} + \eta_k \mathbf{L}(\hat{\mathbf{A}}_k) + \zeta_k \mathbf{L}(\hat{\mathbf{A}}_k \hat{\mathbf{A}}_k)\right]^{-1},$$

$$\hat{\mathbf{A}}_k \triangleq \mathrm{diag}(\boldsymbol{\alpha}_k)\mathbf{A}'\mathrm{diag}(\boldsymbol{\alpha}_k),$$

$$\boldsymbol{\alpha}_k \triangleq [a_{v_1}^{(k)}, a_{v_2}^{(k)}, \ldots, a_{v_{n+m}}^{(k)}]^\top,$$

Node Weights

(to prune uninformative nodes)

[1]The matrix inversion can be bypassed without approximation.

[2]$a^{(k)}_v$ indicates how much attention we pay to a node. It is learned for an in-sample node, but fixed to one for an out-of-sample node, as we are always interested in out-of-sample nodes.

Experimental Results: Classification

Metric	Embedding	Network	Baselines			This Work		Upper Bound
			LocalAvg	MRG	LabelProp	hLGP	DepthLGP	(rerunning)
Macro-F1(%)	LINE	DBLP	37.89	42.15	40.83	47.33	**48.25**	(49.07)
		PPI	10.52	10.02	12.42	13.42	**13.72**	(13.91)
		BlogCatalog	13.25	11.30	17.07	17.41	**18.03**	(18.90)
	GraRep	DBLP	50.61	55.79	55.02	57.43	**58.67**	(62.92)
		PPI	13.65	13.75	12.38	14.80	**14.84**	(15.33)
		BlogCatalog	14.76	14.80	14.71	15.94	**18.45**	(20.15)
	node2vec	DBLP	53.83	59.34	59.25	60.89	**62.63**	(64.87)
		PPI	15.05	13.43	13.78	15.85	**16.54**	(16.81)
		BlogCatalog	15.10	14.04	19.16	19.77	**20.32**	(20.82)
Micro-F1(%)	LINE	DBLP	49.58	50.49	50.88	54.01	**54.94**	(55.84)
		PPI	18.10	15.71	18.81	20.71	**21.42**	(21.43)
		BlogCatalog	27.40	23.21	30.79	31.36	**31.90**	(32.20)
	GraRep	DBLP	60.17	60.62	60.48	61.44	**62.29**	(65.44)
		PPI	20.23	20.35	20.23	20.79	**21.44**	(21.88)
		BlogCatalog	36.44	30.79	33.90	37.57	**38.14**	(38.37)
	node2vec	DBLP	60.54	62.29	62.52	62.83	**64.56**	(65.63)
		PPI	19.70	18.25	18.25	22.63	**23.11**	(23.41)
		BlogCatalog	34.83	25.82	36.94	37.96	**39.64**	(40.34)

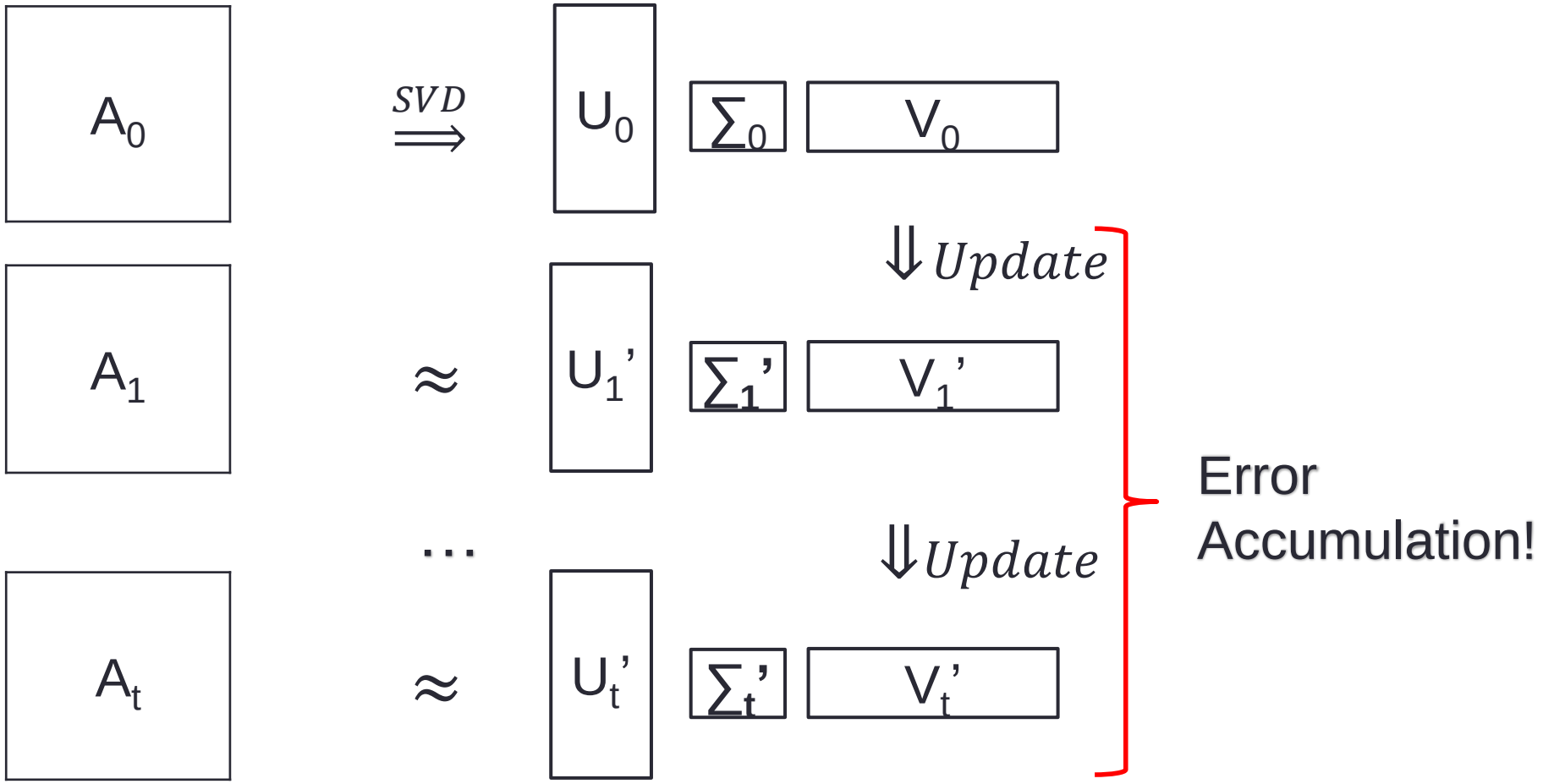
Jianxin Ma, et al. DepthLGP: Learning Embeddings of Out-of-Sample Nodes in Dynamic Networks. ***AAAI***, 2018.

Key problems in dynamic network embedding

- I : Out-of-sample nodes
- II : Incremental edges
- III: Aggregated error
- IV: Scalable optimization

Problem: Error Accumulation

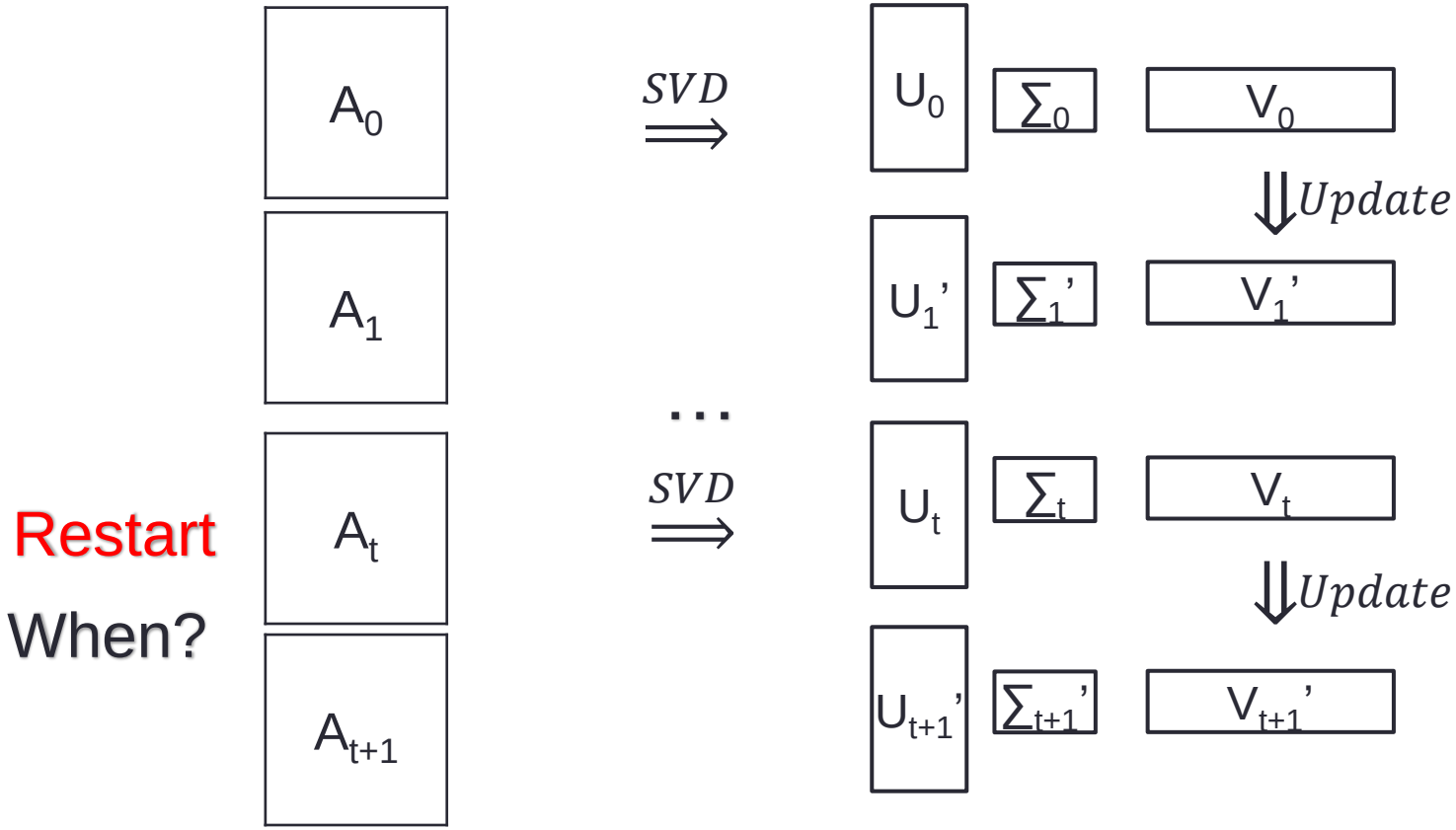
- Eigen perturbation is at the cost of inducing approximation



- Problem: error accumulation is inevitable

Solution: SVD Restarts

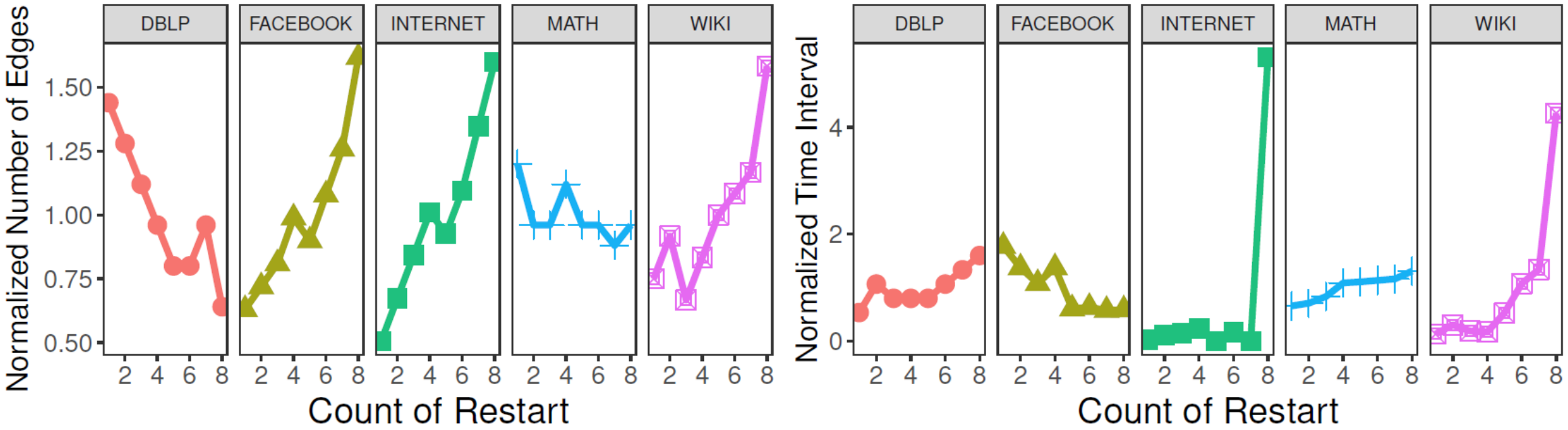
- Solution: restart SVD occasionally



- What are the appropriate time points?
  - Too early restarts: waste of computation resources
  - Too late restarts: serious error accumulation

Naïve Solution

- Naïve solution: fixed time interval or fixed number of changes
- Difficulty: error accumulation is not uniform



Ziwei Zhang, et al. TIMERS: Error-Bounded SVD Restart on Dynamic Networks. ***AAAI***, 2018.

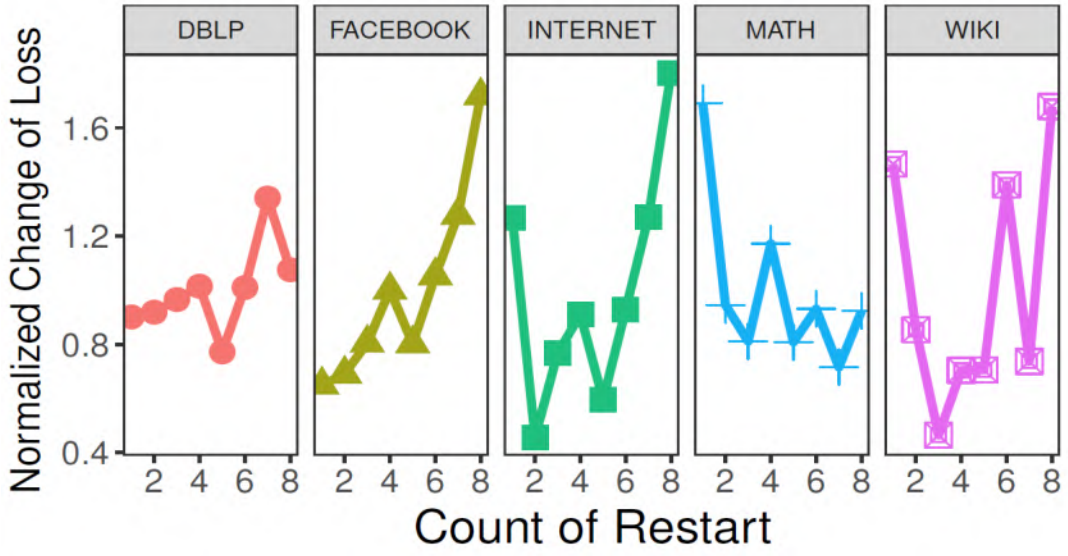
Existing Method

- Existing method: monitor loss (Chen and Candan, KDD 2014)
- Loss in SVD:

$$\mathcal{J} = \|S - U\Sigma V^T\|_F^2$$

S: target matrix, $[U, \Sigma, V]$: results of SVD

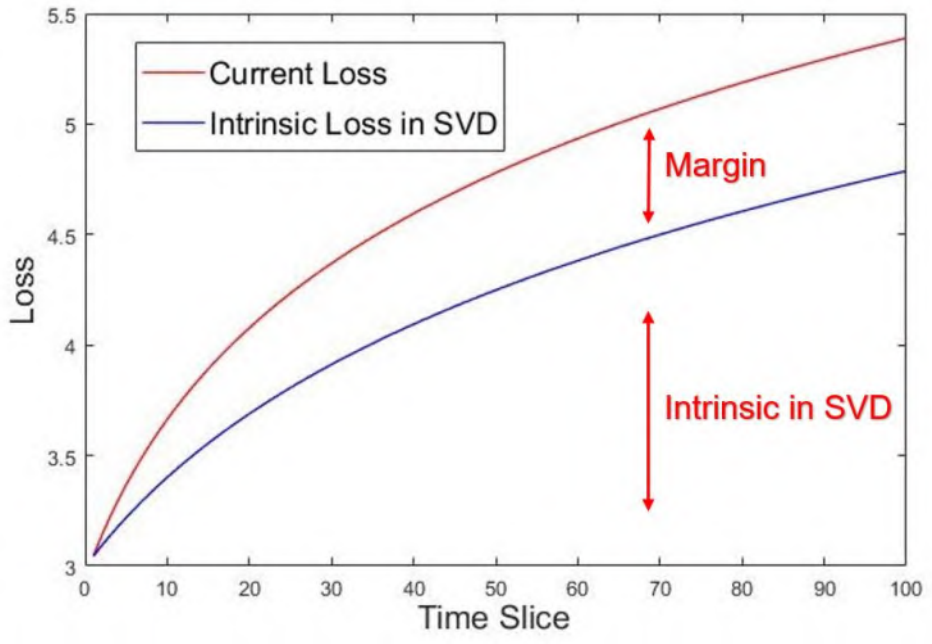
- Problem: loss includes approximation error and intrinsic loss in SVD



Ziwei Zhang, et al. TIMERS: Error-Bounded SVD Restart on Dynamic Networks. ***AAAI***, 2018.

Framework: Monitor Margin

- Observation: the margin between the current loss and the intrinsic loss in SVD is the actual accumulated error
  - Current loss: $\mathcal{J} = \|S - U\Sigma V^T\|_F^2$
  - Intrinsic loss: $\mathcal{L}(S,k) = \min_{U^*,\Sigma^*,V^*} \left\|S - U^*\Sigma^*V^{*T}\right\|_F^2, k: dimensionality$



Ziwei Zhang, et al. TIMERS: Error-Bounded SVD Restart on Dynamic Networks. ***AAAI***, 2018.

Solution: Lazy Restarts

- Lazy restarts: restart only when the margin exceeds the threshold
- Problem: intrinsic loss is hard to compute
  - Direct calculation has the same time complexity as SVD
- Relaxation: an upper bound on margin
  - A lower bound on intrinsic loss $\mathcal{L}(S,k)$

$$\mathcal{L}(\mathbf{S}_t, k) \geq B(t) \Rightarrow \frac{\mathcal{J}(t) - \mathcal{L}(\mathbf{S}_t, k)}{\mathcal{L}(\mathbf{S}_t, k)} \leq \frac{\mathcal{J}(t) - B(t)}{B(t)}.$$

$\mathcal{J}(t)$: current loss; $\mathcal{L}(S\_t,k)$: intrinsic loss; $B(t)$: bound of intrinsic loss

Ziwei Zhang, et al. TIMERS: Error-Bounded SVD Restart on Dynamic Networks. ***AAAI***, 2018.

A Lower Bound of SVD Intrinsic Loss

- Idea: use matrix perturbation

Theorem 1 (A Lower Bound of SVD Intrinsic Loss). *If* $\mathbf{S}$ *and* $\Delta\mathbf{S}$ *are symmetric matrices, then:*

$$\mathcal{L}(\mathbf{S}+\Delta\mathbf{S},k) \geq \mathcal{L}(\mathbf{S},k) + \Delta tr^2(\mathbf{S}+\Delta\mathbf{S},\mathbf{S}) - \sum_{l=1}^{k}\lambda_l, \quad (9)$$

where $\lambda_1 \geq \lambda_2 ... \geq \lambda_k$ *are the top-*k *eigenvalues of* $\nabla_{S^2} = \mathbf{S}\cdot\Delta\mathbf{S} + \Delta\mathbf{S}\cdot\mathbf{S} + \Delta\mathbf{S}\cdot\Delta\mathbf{S}$, *and*

$$\Delta tr^2(\mathbf{S}+\Delta\mathbf{S},\mathbf{S}) = tr\left((\mathbf{S}+\Delta\mathbf{S})\cdot(\mathbf{S}+\Delta\mathbf{S})\right) - tr(\mathbf{S}\cdot\mathbf{S}).$$

- Intuition: treat changes as a perturbation to the original network

Ziwei Zhang, et al. TIMERS: Error-Bounded SVD Restart on Dynamic Networks. ***AAAI***, 2018.

Time Complexity Analysis

Theorem 2. *The time complexity of calculating* $B(t)$ *in Eqn* (13) *is* $O(M_S + M_L k + N_L k^2)$*, where* M_S *is the number of the non-zero elements in* $\Delta\mathbf{S}$*, and* N_L, M_L *are the number of the non-zero rows and elements in* ∇_{S^2} *respectively.*

- If every node has a equal probability of adding new edges, we have: $M_L \approx 2d_{avg}M_S$, where d_{avg} is the average degree of the network .
- For Barabasi Albert model (Barabási and Albert 1999), a typical example of preferential attachment networks, we have: $M_L \approx \frac{12}{\pi^2}\left[log(d_{max}) + \gamma\right] M_S$, where d_{max} is the maximum degree of the network and $\gamma \approx 0.58$ is a constant.

- Conclusion: the complexity is only linear to the local dynamic changes

Ziwei Zhang, et al. TIMERS: Error-Bounded SVD Restart on Dynamic Networks. ***AAAI***, 2018.

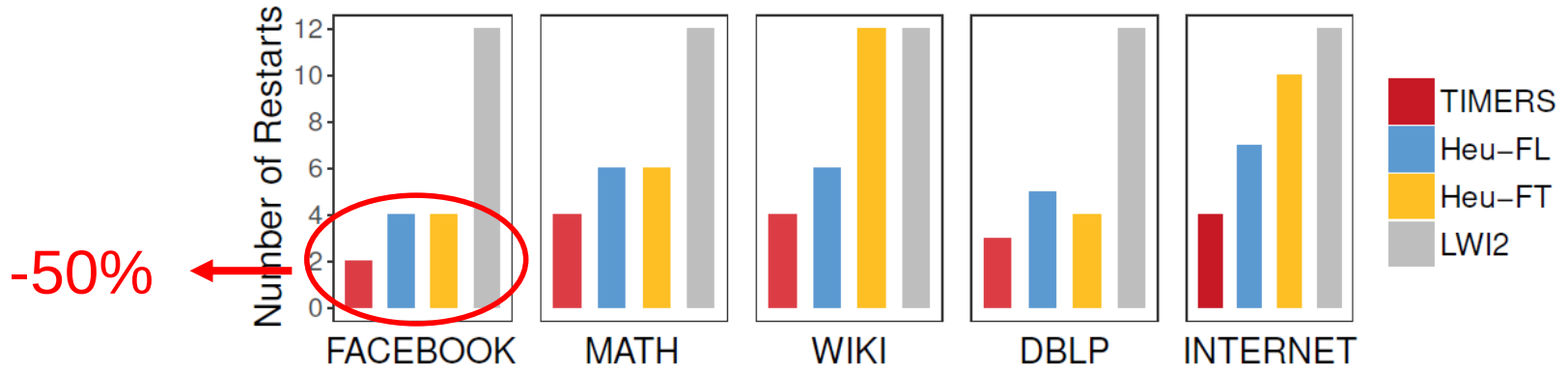
Experimental Results: Approximation Error

- Fixing number of restarts

Dataset	$avg(r)$				$max(r)$			
	TIMERS	LWI2	Heu-FL	Heu-FT	TIMERS	LWI2	Heu-FL	Heu-FT
FACEBOOK	**0.005**	0.020	0.009	0.011	**0.014**	0.038	0.025	0.023
MATH	**0.037**	0.057	0.044	0.051	**0.085**	0.226	0.117	0.179
WIKI	**0.053**	0.086	0.071	0.281	**0.139**	0.332	0.240	0.825
DBLP	**0.042**	0.110	0.053	0.064	**0.121**	0.386	0.198	0.238
INTERNET	**0.152**	0.218	0.196	0.961	**0.385**	0.806	0.647	1.897

27%~42% Improvement

- Fixing maximum error



Ziwei Zhang, et al. TIMERS: Error-Bounded SVD Restart on Dynamic Networks. ***AAAI***, 2018.

Section Summary

- I : Out-of-sample nodes
  - DepthLGP = Non-parametric GP + DNN
- II : Incremental edges
  - DHPE: Generalized Eigen Perturbation
- III: Aggregated error
  - TIMERS: A theoretically guaranteed SVD restart strategy
- IV: Scalable optimization
  - D-SGD: A iteration-wise weighted SGD for highly dynamic data

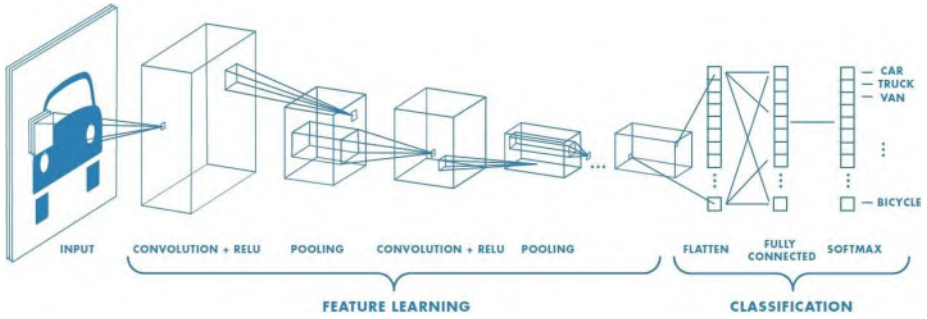
Recap: Network Embedding

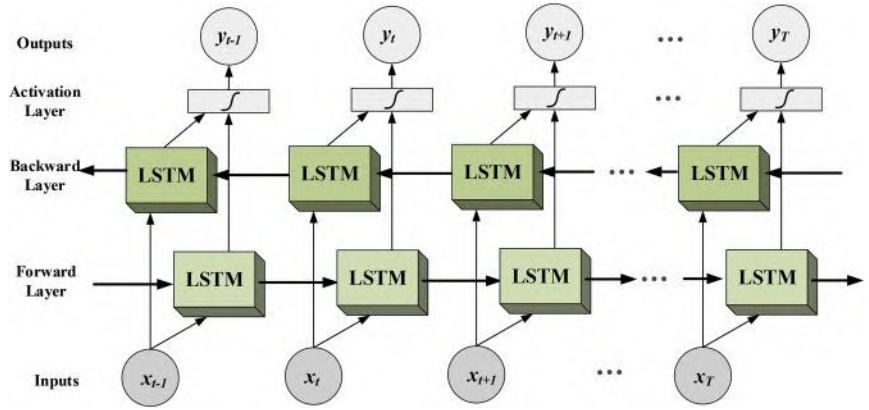
- Structure-preserved network embedding
- Property-preserved network embedding
- Dynamic network embedding

Learning from networks

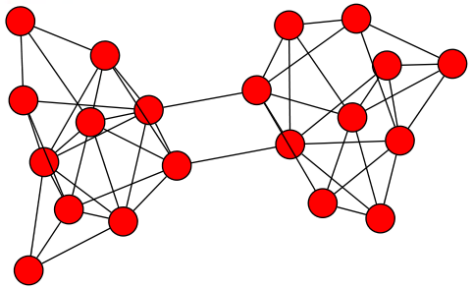
GNN

Graph Neural Networks





$G = (V, E)$



?

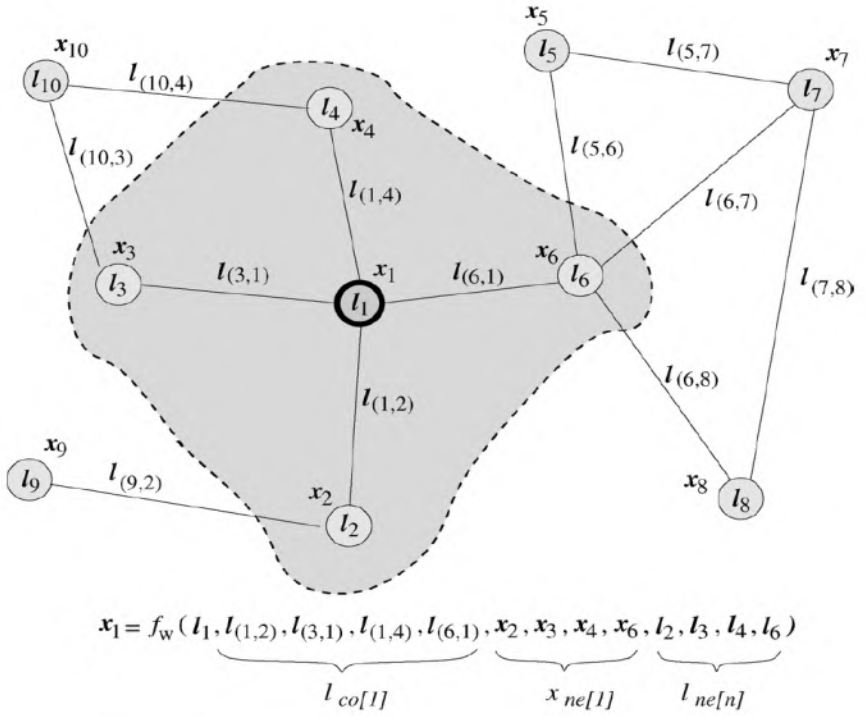
Can we design a learning mechanism to directly work on graphs?

The First Graph Neural Network

- Basic idea: a recursive definition of states

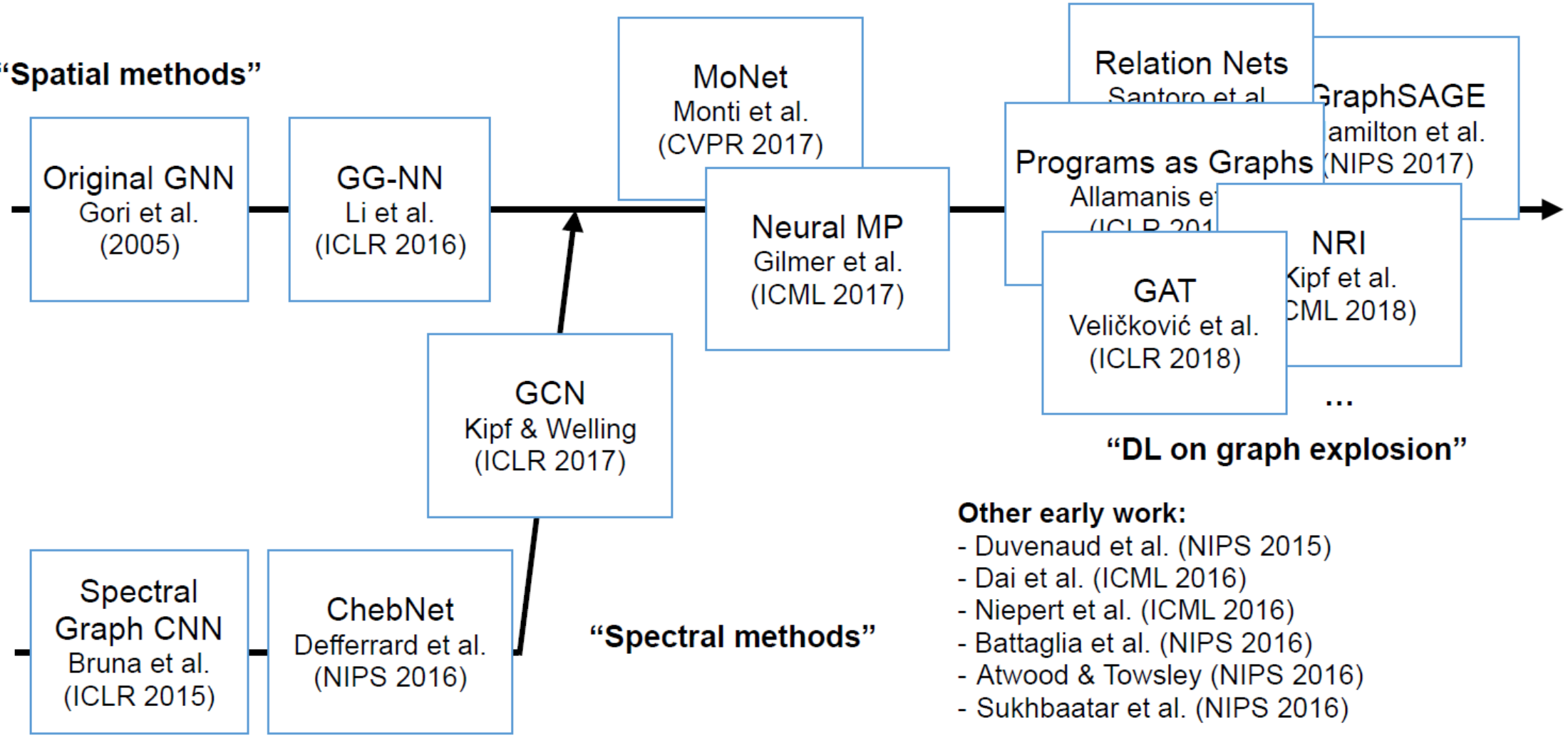
$$\mathbf{s}_i = \sum_{j \in \mathcal{N}(i)} \mathcal{F}\left(\mathbf{s}_i, \mathbf{s}_j, \mathbf{F}_i^V, \mathbf{F}_j^V, \mathbf{F}_{i,j}^E\right)$$

- A simple example: PageRank



F. Scarselli, et al. The graph neural network model. *IEEE TNN, 2009.*

Many GNNs have emerged since then



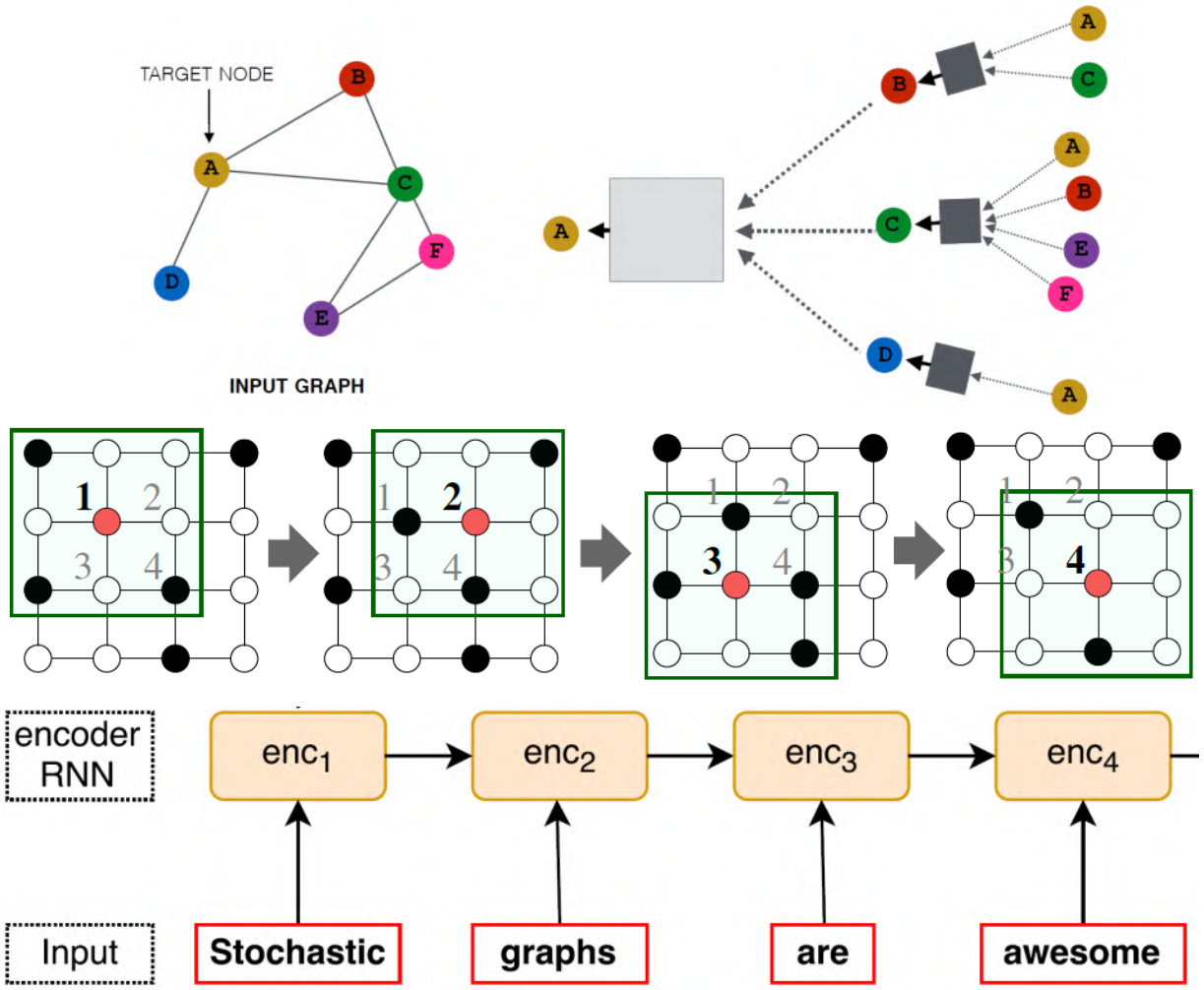
Other early work:

- Duvenaud et al. (NIPS 2015)
- Dai et al. (ICML 2016)
- Niepert et al. (ICML 2016)
- Battaglia et al. (NIPS 2016)
- Atwood & Towsley (NIPS 2016)
- Sukhbaatar et al. (NIPS 2016)

(slide inspired by Alexander Gaunt's talk on GNNs)

Picture credit to Thomas Kipf

How are GNNs compared with other NNs?



- Capture information from graph neighborhoods
- Capture information from nearby grids (i.e., a 2-D graph)
- Capture information from contexts (i.e., a 1-D graph)

We need to exchange information within neighborhoods

Picture credit to GraphSAGE (NIPS 17), PATCHY-SAN (ICML 16)

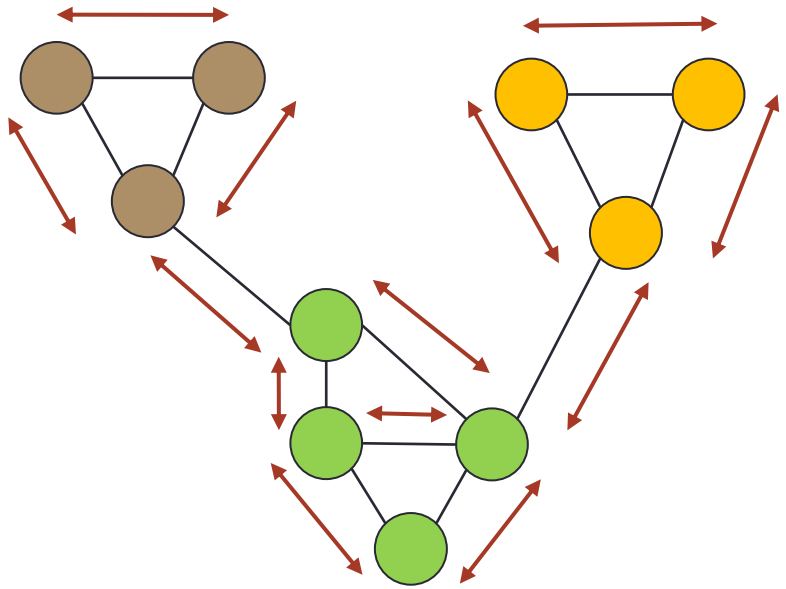
Message-passing Framework

- Formulation:

$$\mathbf{m}_i^{(l)} = \text{AGG}(\{\mathbf{h}_j^{(l)}, \forall j \in \tilde{\mathcal{N}}_i\})$$

$$\mathbf{h}_i^{(l+1)} = \text{UPDATE}([\mathbf{h}_i^{(l)}, \mathbf{m}_i^{(l)}])$$

  - $\mathrm{h}_i^{(l)}$: representation of node v_i in the l^{th} layer
  - $\mathrm{m}_i^{(l)}$: messages for node v_i in the l^{th} layer by aggregating neighbor representations



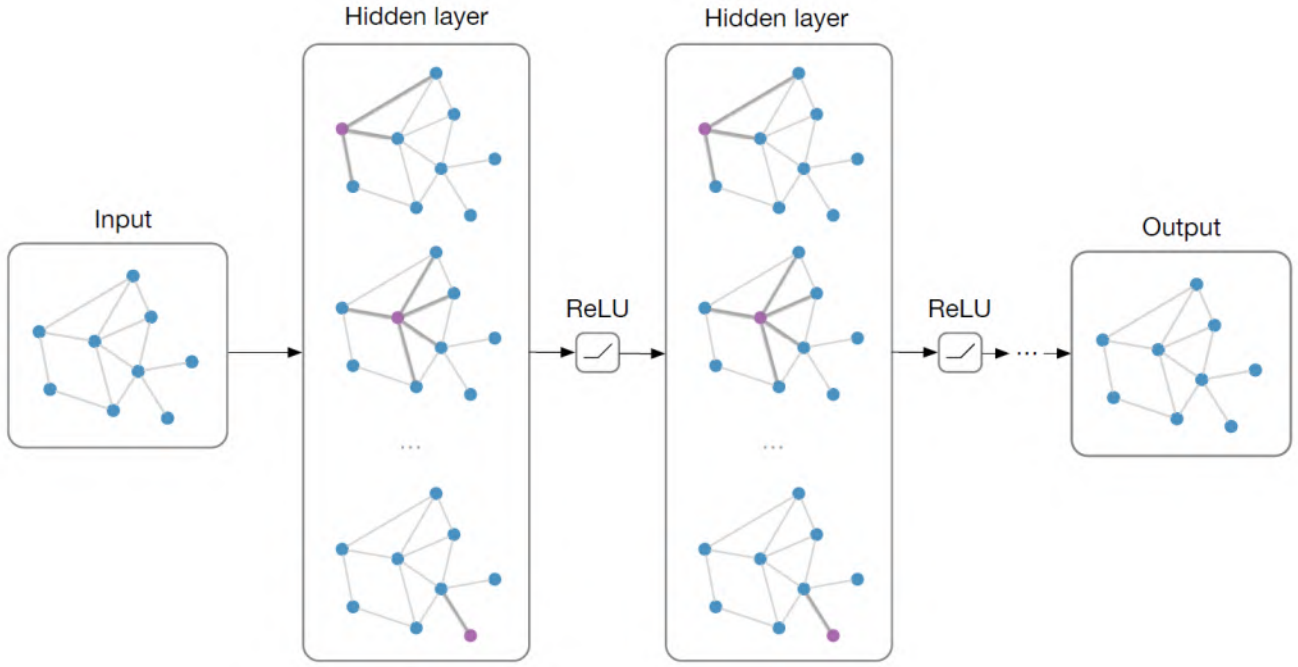
J. Gilmer, et al. Neural message passing for quantum chemistry. *ICML, 2017.*
W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. *NIPS, 2017.*

Graph Convolutional Networks (GCN)

- Main idea: averaging messages from direct neighborhoods

$$\mathbf{H}^{l+1} = \rho\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{l}\mathbf{\Theta}^{l}\right)$$

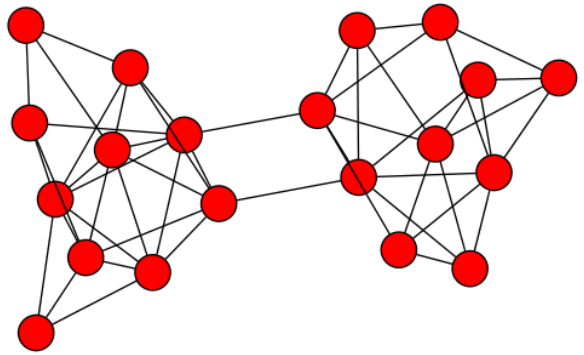
- Stacking multiple layers like standard CNNs:
  - State-of-the-art results on node classification

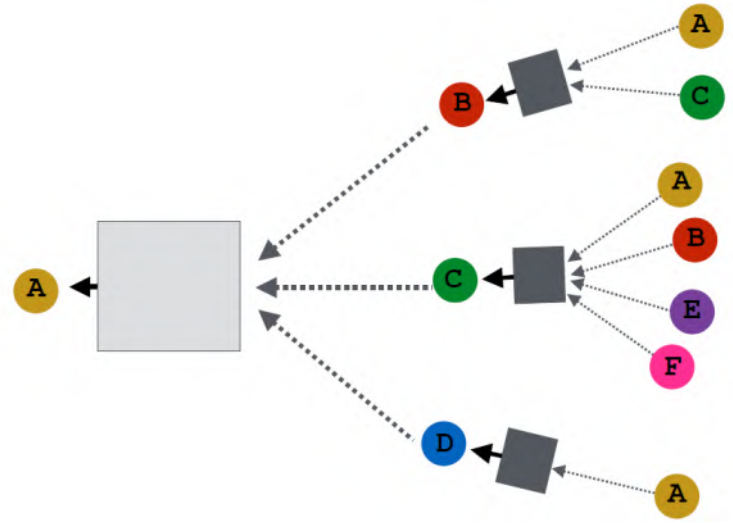


T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *ICLR, 2017.*

Expected Properties of GNNs

$G = (V, E)$





- Some expected properties of GNNs:
  - Trained end-to-end for downstream tasks
    - Vs. network embedding: unsupervised representation learning to handle various tasks
  - Utilize node features and graph structures simultaneously
  - A deep learning model
  - Can handle real applications with data represented as graphs

Are existing GNNs good enough?

Outline

- Does GNN fuse *feature* and *topology* optimally?
- Is GNN really a *deep* model?
- Technical challenges in real applications: robustness, explainability and applicability

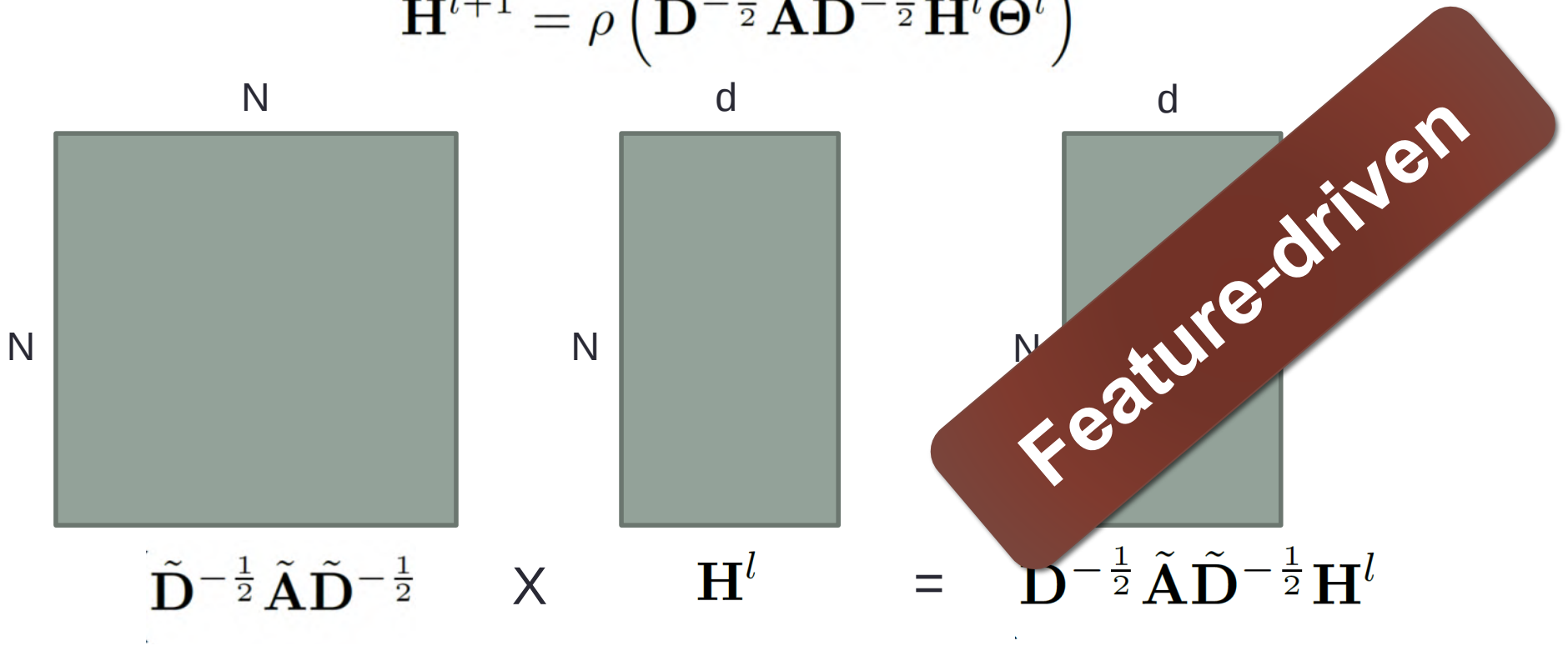
Outline

- Does GNN fuse *feature* and *topology* optimally?
- Is GNN really a *deep* model?
- Technical challenges in real applications: robustness, explainability and applicability

The intrinsic problem GCN is solving

Fusing topology and features in the way of smoothing features with the assistance of topology with a deep model.

$$\mathbf{H}^{l+1} = \rho\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}^{l}\mathbf{\Theta}^{l}\right)$$



Revisiting Graph Neural Networks All We Have is Low-Pass Filters, *arXiv 1905.09550*
Understanding the Representation Power of Graph Neural Networks in Learning Graph Topology, *NeurIPS 2019*

Can GNNs Fully Preserve Graph Structures?

- When feature plays the key role, GNN performs good
- How about the contrary?
- Synthesis data: stochastic block model + random features
  - DeepWalk greatly outperforms all the GCNs
  - Recall the message-passing framework

$$\mathbf{m}_i^{(l)} = \text{AGG}(\{\mathbf{h}_j^{(l)}, \forall j \in \tilde{\mathcal{N}}_i\})$$

$$\mathbf{h}_i^{(l+1)} = \text{UPDATE}([\mathbf{h}_i^{(l)}, \mathbf{m}_i^{(l)}])$$

Initialized as node features

Graph structures only provide neighborhoods in aggregation

- Initial node features provide important inductive bias!

Method	Results
Random	10.0
GCN-1	29.7
GCN-2	48.4
GCN-3	56.2
GCN-5	53.3
DeepWalk	99.9

GCN-X: X number of layers

Can GNNs Fully Preserve Graph Structures?

- Theoretical analysis: node features as "true signal", GNNs as a low-pass filtering
  - Simplified GCN[1]: removing all non-linearity

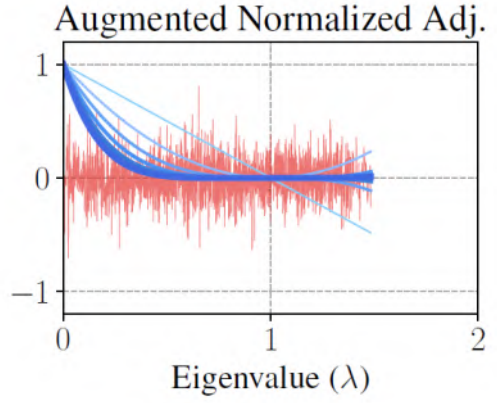
$$H^{(l)} = S^l H^{(0)}$$

$$S = \tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2} = I - \tilde{L}_{Sym}$$

  - From graph signal processing, $f' = S^l f$ corresponds to a spectral filter [2]

$$g(\lambda) = \left(1 - \hat{\lambda}\right)^l$$

  - Thus GNNs are inevitably feature-centric



- More recent results on learning graph moments [3]:

Proposition 1. *A graph convolutional network with n layers, and no bias terms, in general, can learn $f(A)_i = \sum_j (A^n)_{ij}$ only if $n = p$ or $n > p$ if the bias is allowed.*

  - Graph moment: similar to high-order proximities in network embedding

1. Simplifying Graph Convolutional Networks, *ICML 2019*
2. Revisiting Graph Neural Networks All We Have is Low-Pass Filters, *arXiv 1905.09550*
3. Understanding the Representation Power of Graph Neural Networks in Learning Graph Topology, *NeurIPS 2019*

A New Perspective to Understand GNNs

- How can we empower GNNs to preserve graph structures well?
- A new perspective: treating GNNs as a type of (non-linear) dimensionality reduction
  - A slightly modified framework:

$$\mathbf{H}^{(l+1)} = \sigma\left(\mathcal{F}(\mathbf{A})\,\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right)$$

Non-linear mapping

Graph structures

Previous bases

Linear transform

Method	$\mathcal{F}(\mathbf{A})$
GCN, SGC	$\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$
DCNN	$\left(\mathbf{D}^{-1}\mathbf{A}\right)^{K}$
DGCN	$\mathbf{D}_P^{-\frac{1}{2}}\mathbf{A}_P\mathbf{D}_P^{-\frac{1}{2}}$
PPNP	$\alpha\left(\mathbf{I}_N-(1-\alpha)\,\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\right)^{-1}$

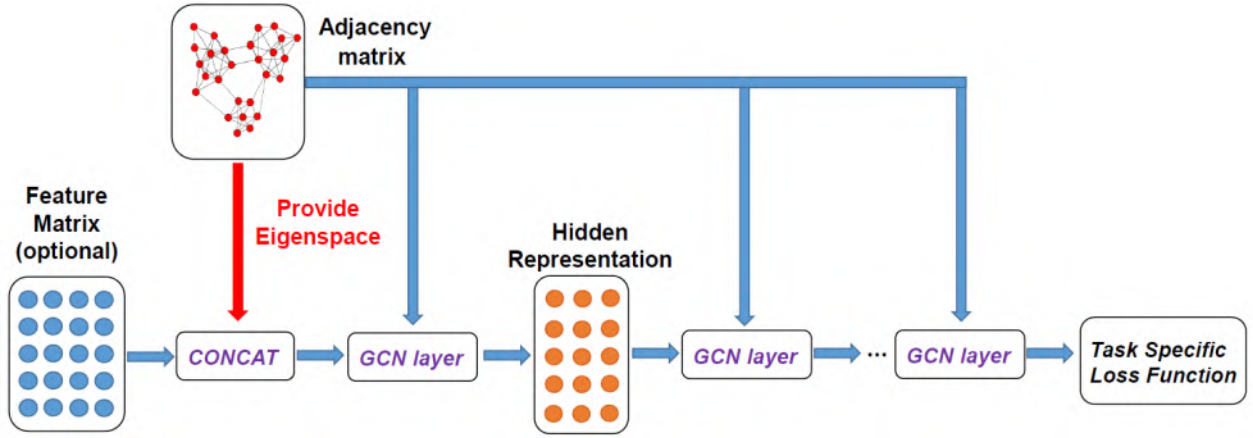
  - Three-steps
    (1) Projecting graph structures into a subspace spanned by node representations in the last step
    (2) The projected representations are linearly transformed followed by a non-linear mapping
    (3) Repeat the process by using the new node representations as bases
- Why are the existing GNNs feature-centric?

→ The initial space is solely determined by node features!

Ziwei Zhang, Peng Cui, Jian Pei, Xin Wang, Wenwu Zhu. Eigen-GNN: A Graph Structure Preserving Plug-in for GNNs. *arXiv, 2006.04330*.

Eigen-GNN: A Graph Structure Preserving Plug-in

- Framework



$$\mathbf{H}^{(0)} = [\mathbf{X}, f(\mathbf{Q})]$$

X: node features; Q: top-d eigenvector of a graph structure matrix; $f(\cdot)$: a simple function such as normalization

- Experimental results:

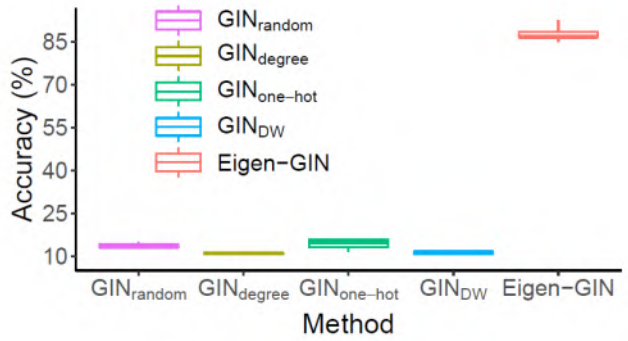
Node classification

Data	Method	Harvard	Columbia	Stanford
A,Y	GCN_{random}	74.6±0.5	63.6±1.6	68.2±1.5
	GCN_{degree}	74.4±2.0	63.8±2.3	67.8±1.6
	GCN_{DW}	82.5±1.0	**76.0 ± 1.3**	76.6±1.3
	Eigen-GCN_{struc}	**82.7 ± 1.2**	**76.0 ± 1.9**	**78.9 ± 1.3**
A,X,Y	GCN_{feat}	70.6±1.3	74.8±1.7	71.3±1.6
	$GCN_{feat+DW}$	83.1±0.7	77.6±1.3	78.3±1.4
	Eigen-$GCN_{feat+struc}$	**84.6 ± 1.4**	**78.6 ± 1.1**	**79.7 ± 1.2**

Link Prediction

Dataset	C.elegans	E.coli	Power
SEAL	77.6±0.9	91.5±0.8	69.9±1.6
$SEAL_{DW}$	77.1±1.5	92.1±0.7	69.8±1.5
Eigen-SEAL	**79.5 ± 0.8***	**92.5 ± 0.6***	**73.2 ± 2.4***
Gain†	+1.9	+0.4	+3.3

Graph Isomorphism Test

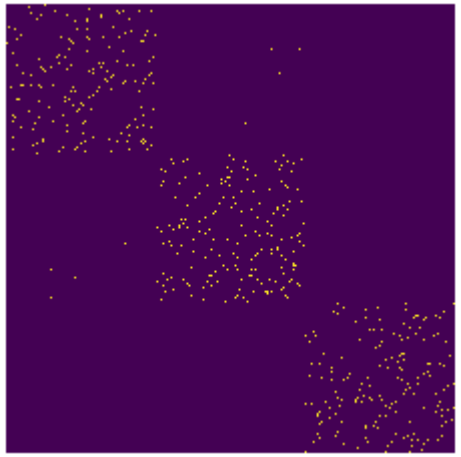


Ziwei Zhang, Peng Cui, Jian Pei, Xin Wang, Wenwu Zhu. Eigen-GNN: A Graph Structure Preserving Plug-in for GNNs. *arXiv, 2006.04330*.

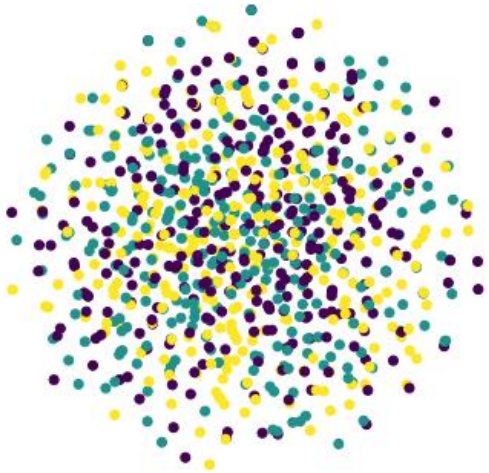
Can GNNs Fully Preserve Node Features?

- Though GNNs have a feature-driven mechanism, can they preserve node features well?

Case 1



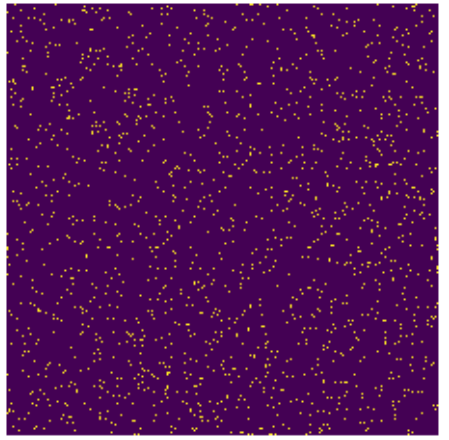
Correlated Topology



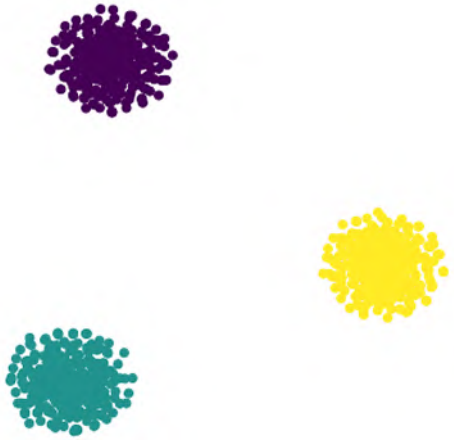
Random Features

DeepWalk > GCN

Case 2



Random topology



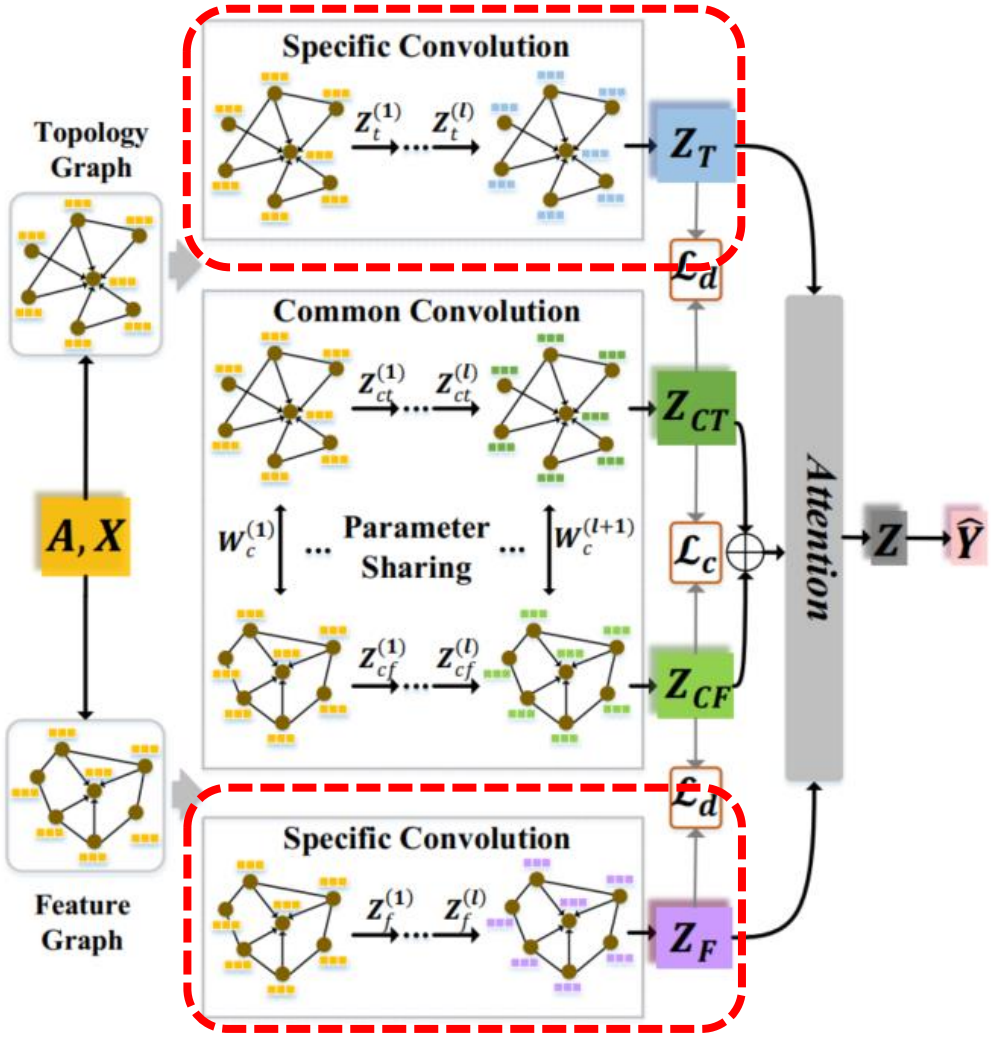
Correlated Features

MLP> GCN

- Reason: the neighborhood only depends on structures (i.e., not depend on features)
- How to find the most suitable neighborhoods adaptively?

GNNs also fail in preserving node features!

Adaptive Multi-channel GCN



Specific Convolution Module

- **Topology Graph** $G_t = (\mathbf{A}, \mathbf{X})$

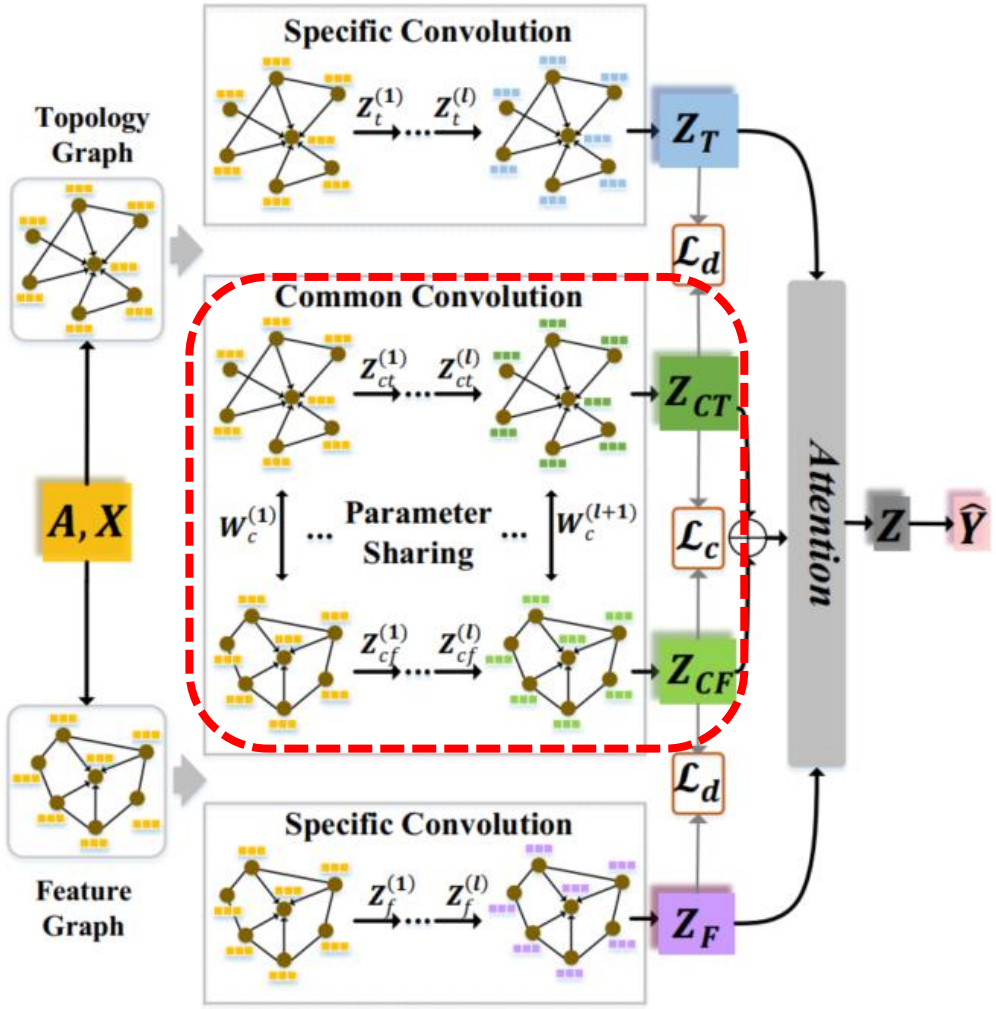
$$\mathbf{Z}_t^{(l)} = ReLU(\tilde{\mathbf{D}}_t^{-\frac{1}{2}} \tilde{\mathbf{A}}_t \tilde{\mathbf{D}}_t^{-\frac{1}{2}} \mathbf{Z}_t^{(l-1)} \mathbf{W}_t^{(l)}),$$

- **Feature Graph** $G_f = (\mathbf{A}_f, \mathbf{X})$

$$\mathbf{Z}_f^{(l)} = ReLU(\tilde{\mathbf{D}}_f^{-\frac{1}{2}} \tilde{\mathbf{A}}_f \tilde{\mathbf{D}}_f^{-\frac{1}{2}} \mathbf{Z}_f^{(l-1)} \mathbf{W}_f^{(l)}),$$

Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, Jian Pei. AM-GCN: Adaptive Multi-channel Graph Convolutional Networks. *KDD, 2020.*

Adaptive Multi-channel GCN



Common Convolution Module

- Topology Graph $G_t = (\mathbf{A}, \mathbf{X})$

$$\mathbf{Z}_{ct}^{(l)} = ReLU(\tilde{\mathbf{D}}_t^{-\frac{1}{2}} \tilde{\mathbf{A}}_t \tilde{\mathbf{D}}_t^{-\frac{1}{2}} \mathbf{Z}_{ct}^{(l-1)} \mathbf{W}_c^{(l)}),$$

- Feature Graph $G_f = (\mathbf{A}_f, \mathbf{X})$

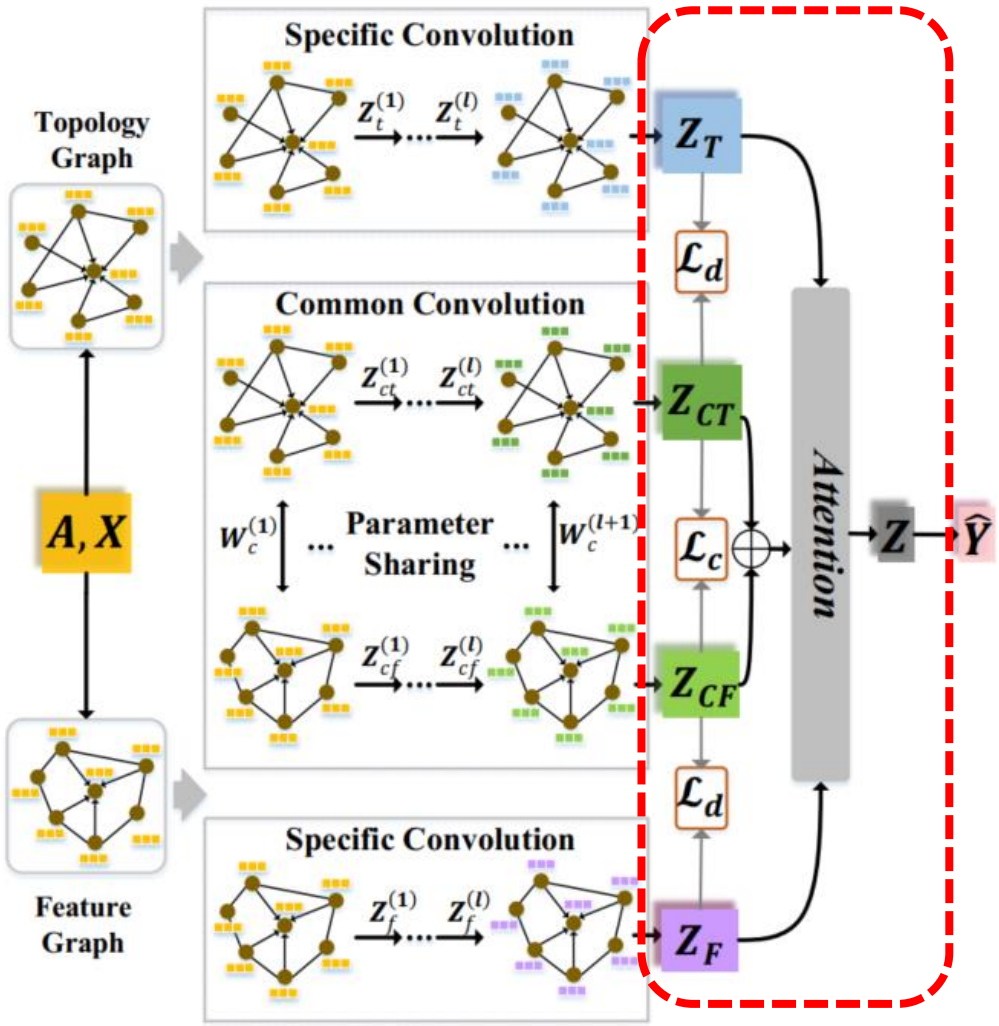
$$\mathbf{Z}_{cf}^{(l)} = ReLU(\tilde{\mathbf{D}}_f^{-\frac{1}{2}} \tilde{\mathbf{A}}_f \tilde{\mathbf{D}}_f^{-\frac{1}{2}} \mathbf{Z}_{cf}^{(l-1)} \mathbf{W}_c^{(l)}),$$

- Parameter Sharing
- Common Embedding

$$\mathbf{Z}_C = (\mathbf{Z}_{CT} + \mathbf{Z}_{CF})/2.$$

Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, Jian Pei. AM-GCN: Adaptive Multi-channel Graph Convolutional Networks. *KDD, 2020.*

Adaptive Multi-channel GCN



Attention Mechanism

$$(\boldsymbol{\alpha}_t, \boldsymbol{\alpha}_c, \boldsymbol{\alpha}_f) = att(\mathbf{Z}_T, \mathbf{Z}_C, \mathbf{Z}_F),$$

$$\boldsymbol{\alpha}_T = diag(\boldsymbol{\alpha}_t)\ \boldsymbol{\alpha}_C = diag(\boldsymbol{\alpha}_c)\ \boldsymbol{\alpha}_F = diag(\boldsymbol{\alpha}_f).$$

$$\mathbf{Z} = \boldsymbol{\alpha}_T \cdot \mathbf{Z}_T + \boldsymbol{\alpha}_C \cdot \mathbf{Z}_C + \boldsymbol{\alpha}_F \cdot \mathbf{Z}_F.$$

How to Calculate $\boldsymbol{\alpha}_t$

$$\omega_T^i = \mathbf{q}^T \cdot tanh(\mathbf{W}_T \cdot (\mathbf{z}_T^i)^T + \mathbf{b}_T).$$

$$\alpha_T^i = softmax(\omega_T^i) = \frac{exp(\omega_T^i)}{exp(\omega_T^i) + exp(\omega_C^i) + exp(\omega_F^i)}.$$

$$\boldsymbol{\alpha}_t = [\alpha_T^i]$$

Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, Jian Pei. AM-GCN: Adaptive Multi-channel Graph Convolutional Networks. *KDD, 2020.*

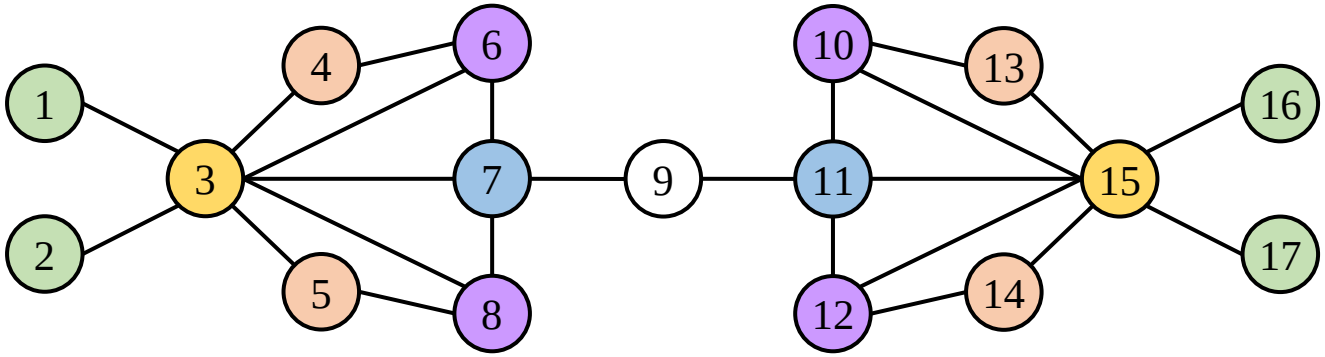
Experimental Results

Datasets	Metrics	L/C	DeepWalk	LINE	Chebyshev	GCN	*k*NN-GCN	GAT	DEMO-Net	MixHop	AM-GCN
Citeseer	ACC	20	43.47	32.71	69.80	70.30	61.35	72.50	69.50	71.40	**73.30**
		40	45.15	33.32	71.64	73.10	61.54	73.04	70.44	71.48	**74.70**
		60	48.86	35.39	73.26	74.48	62.38	74.76	71.86	72.16	**75.56**
	F1	20	38.09	31.75	65.92	67.50	58.86	68.14	67.84	66.96	**69.22**
		40	43.18	32.42	68.31	69.70	59.33	69.58	66.97	67.40	**69.81**
		60	48.01	34.37	70.31	71.24	60.07	**71.60**	68.22	69.31	70.92
UAI2010	ACC	20	42.02	43.47	50.02	49.88	66.06	56.92	23.45	61.56	**70.46**
		40	51.26	45.37	58.18	51.80	68.74	63.74	30.29	65.05	**73.14**
		60	54.37	51.05	59.82	54.40	71.64	68.44	34.11	67.66	**74.40**
	F1	20	32.93	37.01	33.65	32.86	52.43	39.61	16.82	49.19	**55.61**
		40	46.01	39.62	38.80	33.80	54.45	45.08	26.36	53.86	**64.88**
		60	44.43	43.76	40.60	34.12	54.78	48.97	29.05	56.31	**65.99**
ACM	ACC	20	62.69	41.28	75.24	87.80	78.52	87.36	84.48	81.08	**90.70**
		40	63.00	45.83	81.64	89.06	81.66	88.60	85.70	82.34	**90.76**
		60	67.03	50.41	85.43	90.54	82.00	90.40	86.55	83.09	**91.42**
	F1	20	62.11	40.12	74.86	87.82	78.14	87.44	84.16	81.40	**90.63**
		40	61.88	45.79	81.26	89.00	81.53	88.55	84.83	81.13	**90.66**
		60	66.99	49.92	85.26	90.49	81.95	90.39	84.05	82.24	**91.36**

Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, Jian Pei. AM-GCN: Adaptive Multi-channel Graph Convolutional Networks. *KDD, 2020.*

Permutation-equivariance of GNNs

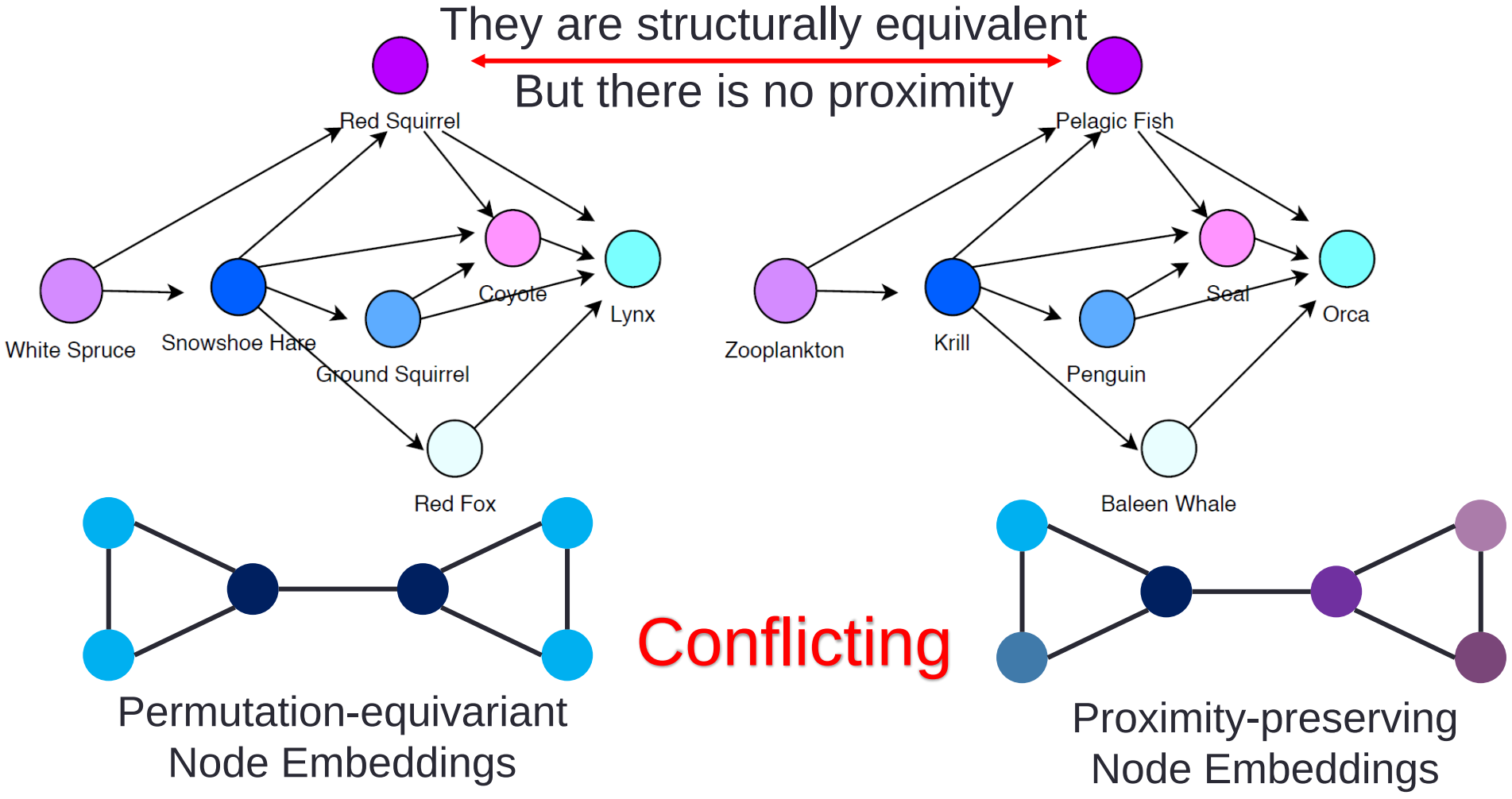
- Permutation-equivariance property
  - If we randomly permute the IDs of nodes while maintaining the graph structure, the representations of nodes in GNNs should be permuted accordingly
- Pros:
  - Guarantees that the embeddings of automorphic nodes are identical
  - Automatically generalize to all the $O(N!)$ permutations when training with only one permutation
- Most of the existing message-passing GNNs satisfy permutation-equivariance



Permutation-equivariant Node Embeddings

Permutation-equivariance vs. Proximity-aware

- However, permutation-equivariance and proximity-aware are conflicting

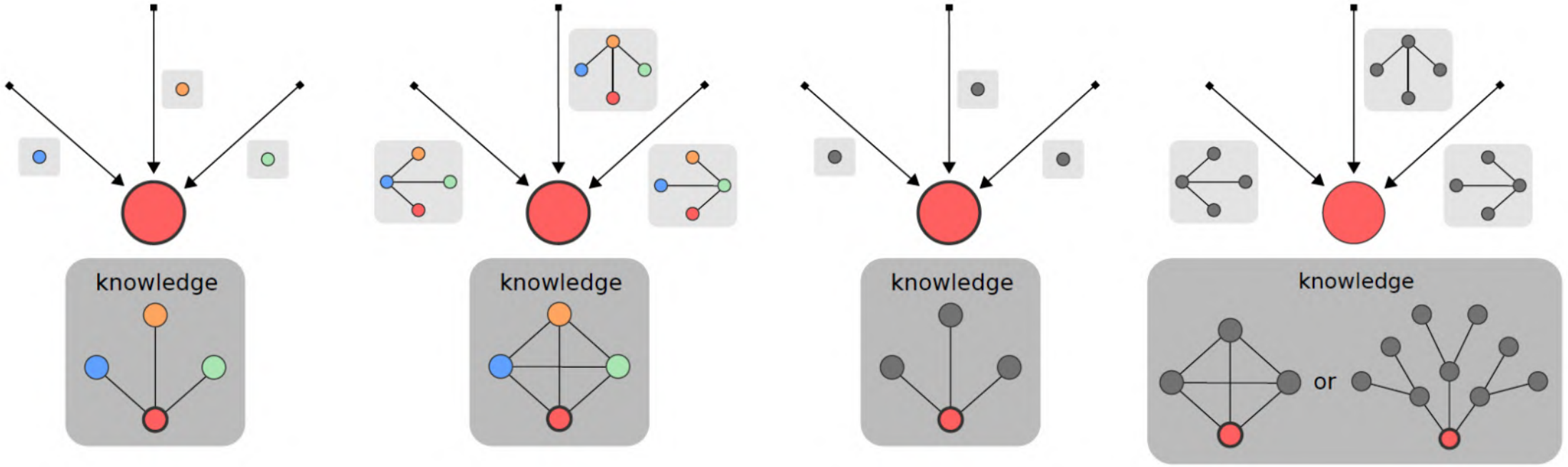


Position-aware graph neural networks. *ICML 2019.*

On the Equivalence between Positional Node Embeddings and Structural Graph Representations. *ICLR 2020.*

Unique Node Identifiers

- The key problem is that nodes cannot differentiate each other

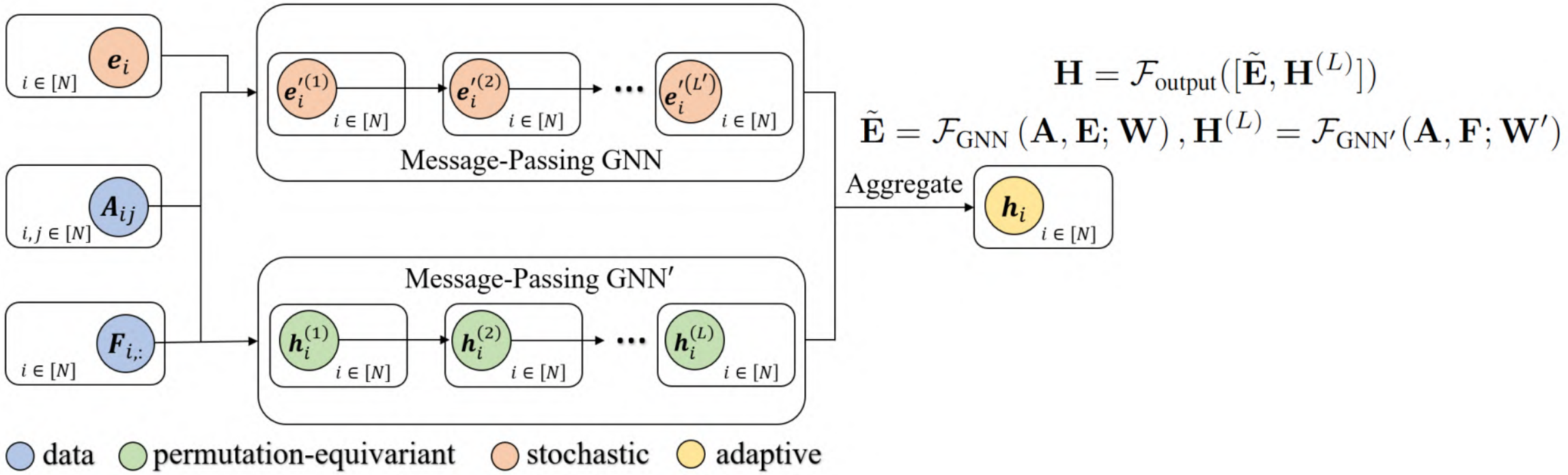


- Theoretical analysis: unique node identifiers are one necessary condition for GNNs to be universal approximation

What graph neural networks cannot learn depth vs width, *ICLR 2020*

Stochastic Message Passing (SMP)

- Assign stochastic features as node identifiers
  - Gaussian features: associate with random projection literature
- A dual GNN architecture:



Ziwei Zhang, Chenhao Niu, Peng Cui, et al. A Simple and General Graph Neural Network with Stochastic Message Passing. *arXiv, 2009.02562.*

Theoretical Guarantee

- SMP can preserve node proximities

Theorem 2. *An SMP in Eq. (9) with the message-passing matrix $\tilde{\mathbf{A}}$ and the number of propagation steps K can preserve the walk-based proximity $\tilde{\mathbf{A}}^K(\tilde{\mathbf{A}}^K)^T$ with high probability if the dimensionality of the stochastic matrix d is sufficiently large, where the superscript T denotes matrix transpose. The theorem is regardless of whether $\mathbf{E}$ are fixed or resampled.*

- SMP can recover the existing permutation-equivariant GNNs

Corollary 2. *For any task, Eq. (8) with the aforementioned linear $\mathcal{F}_{output}(\cdot)$ is at least as powerful as the permutation-equivariant $\mathcal{F}_{GNN'}(\mathbf{A}, \mathbf{F}; \mathbf{W}')$, i.e., the minimum training loss of using $\mathbf{H}$ in Eq. (8) is equal to or smaller than using $\mathbf{H}^{(L)} = \mathcal{F}_{GNN'}(\mathbf{A}, \mathbf{F}; \mathbf{W}')$.*

- An adaptive GNN that maintains both proximity-awareness and permutation-equivariance

Ziwei Zhang, Chenhao Niu, Peng Cui, et al. A Simple and General Graph Neural Network with Stochastic Message Passing. *arXiv, 2009.02562.*

Experimental Results

Table 2: The results of link prediction tasks measured in AUC (%). The best results and the second-best results for each dataset, respectively, are in bold and underlined.

Model	Grid	Communities	Email	CS	Physics	PPI
SGC	57.6±3.8	51.9±1.6	68.5±7.0	96.5±0.1	**96.6±0.1**	80.5±0.4
GCN	61.8±3.6	50.3±2.5	67.4±6.9	93.4±0.3	93.8±0.2	78.0±0.4
GAT	61.0±5.5	51.1±1.6	53.5±6.3	93.7±0.9	94.1±0.4	79.3±0.5
PGNN[5]	73.4±6.0	97.8±0.6	70.9±6.4	82.2±0.5	Out of memory	80.8±0.4
SMP-Identity	55.1±4.8	**98.0±0.7**	72.9±5.1	96.5±0.1	96.5±0.1	81.0±0.2
SMP-Linear	**73.6±6.2**	97.7±0.5	**75.7±5.0**	**96.7±0.1**	96.1±0.1	**81.9±0.3**

Superior performance when the task is proximity-related

Table 3: The results of node classification tasks measured by accuracy (%). The best results and the second-best results for each dataset, respectively, are in bold and underlined.

Model	Communities	CS	Physics	Cora	CiteSeer	PubMed
SGC	7.1±2.1	67.2±12.8	92.3±1.6	76.9±0.2	63.6±0.0	74.2±0.1
GCN	7.5±1.2	91.1±0.7	93.1±0.8	81.4±0.5	**71.3±0.5**	**79.3±0.4**
GAT	5.0±0.0	90.5±0.5	**93.1±0.4**	**82.9±0.5**	71.2±0.6	77.9±0.5
PGNN	5.2±0.5	77.6±7.6	Out of memory	59.2±1.5	55.7±0.9	Out of memory
SMP-Linear	**99.9±0.3**	**91.5±0.8**	93.1±0.8	80.9±0.8	68.2±1.0	76.5±0.8

Comparable performance when permutation-equivariant is helpful

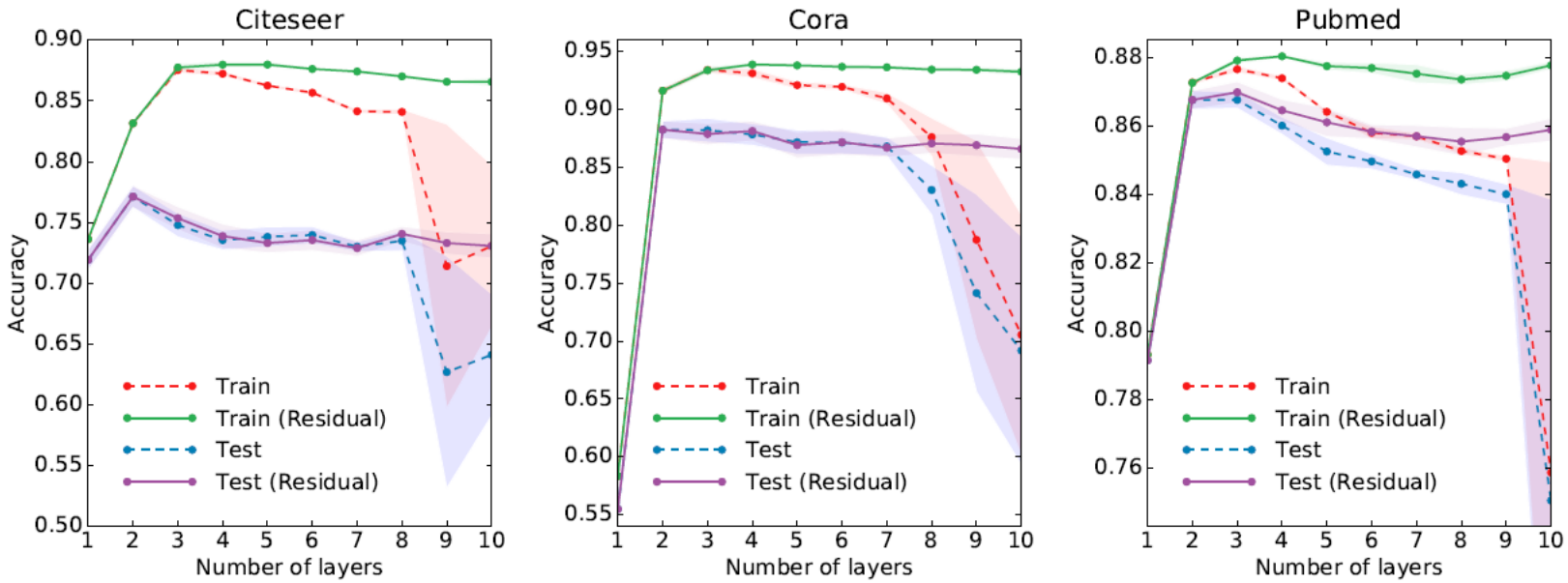
Ziwei Zhang, Chenhao Niu, Peng Cui, et al. A Simple and General Graph Neural Network with Stochastic Message Passing. *arXiv, 2009.02562.*

Outline

- Does GNN fuse *feature* and *topology* optimally?
- Is GNN really a *deep* model?
- Technical challenges in real applications: robustness, explainability and applicability

Is GNN really a deep model?

- Though GNNs are motivated as “deep learning”, most models only adopt 2-3 layers

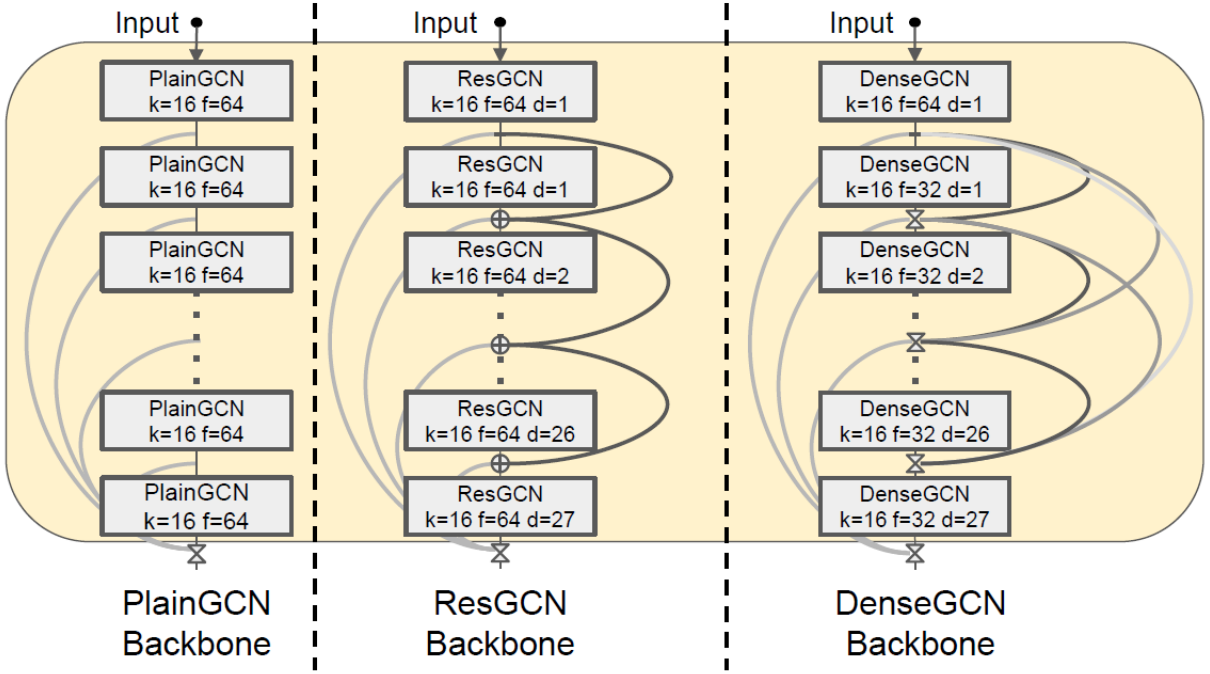


- Possible reasons:
  - Difficulties in training deep networks, e.g., over-fitting, vanishing gradients
  - Over-smoothing, i.e. all nodes have similar representations in deeper layers

Semi-Supervised Classification with Graph Convolutional Networks, *ICLR 2017.*
Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning, *AAAI 2018.*
Graph Neural Networks Exponentially Lose Expressive Power for Node Classification, *ICLR 2020.*

Is GNN really a deep model?

- State-of-the-art CNNs can have hundreds of layers, e.g., ResNet, DenseNet
  - Can we mimic them and use similar ideas to develop GNNs?
- Residual connections, dense connections, and dilated aggregation



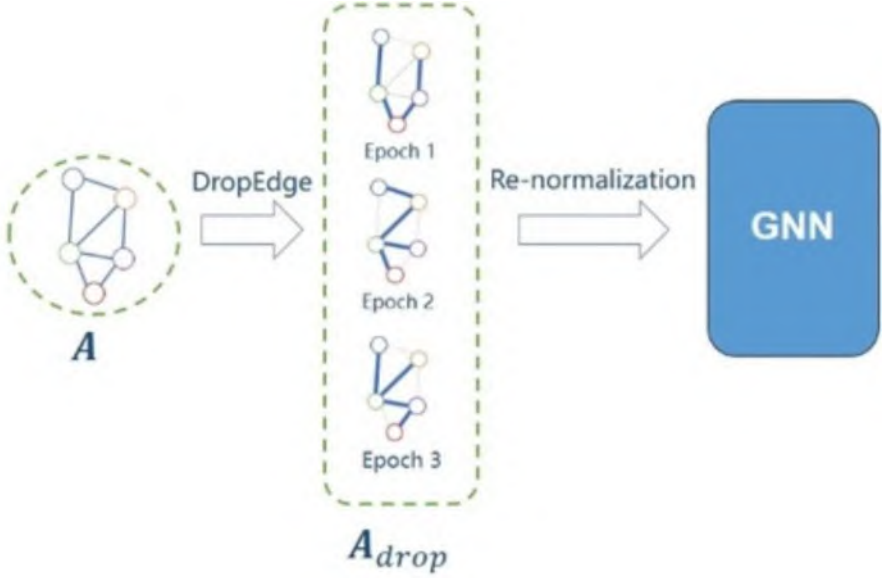
Method	OA	mIOU
PointNet [27]	78.5	47.6
MS+CU [8]	79.2	47.8
G+RCU [8]	81.1	49.7
PointNet++ [29]	-	53.2
3DRNN+CF [48]	**86.9**	56.3
DGCNN [42]	84.1	56.1
ResGCN-28 (*Ours*)	85.9	**60.0**

DeepGCNs: Can GCNs Go as Deep as CNNs? *ICCV 2019*

More Recent Approaches

- Design specific dropout to graph data

→ Randomly drop edges in each epoch

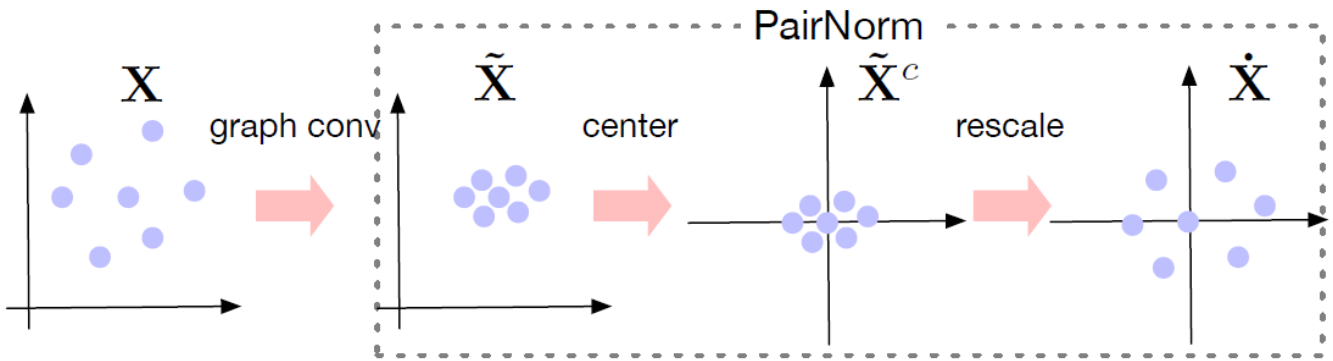


- Theoretical insights:
  - A data augmentation technique
  - A message passing reducer

DropEdge: Towards Deep Graph Convolutional Networks on Node Classification, *ICLR 2020*

- How to prevent node embedding from being similar?

→ Make total pairwise feature distances remained a constant across layers



$$\tilde{\mathbf{x}}_i^c = \tilde{\mathbf{x}}_i - \frac{1}{n}\sum_{i=1}^{n}\tilde{\mathbf{x}}_i \qquad \text{(Center)}$$

$$\dot{\mathbf{x}}_i = s \cdot \frac{\tilde{\mathbf{x}}_i^c}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}\|\tilde{\mathbf{x}}_i^c\|_2^2}} = s\sqrt{n} \cdot \frac{\tilde{\mathbf{x}}_i^c}{\sqrt{\|\tilde{\mathbf{X}}^c\|_F^2}} \qquad \text{(Scale)}$$

PairNorm: Tackling Oversmoothing in GNNs, *ICLR 2020*

A Hot Research Topic

- How to train real deep GNNs is still an on-going research topic
  - DeepGCNs, DropEdge, PairNorm
  - Many more:
    - Measuring and Relieving the Over-smoothing Problem for Graph Neural Networks from the Topological View, *AAAI 2020*
    - Towards Deeper Graph Neural Networks, *KDD 2020*
    - What graph neural networks cannot learn: depth vs width, *ICLR 2020*
    - Simple and deep GNN, *ICML 2020*
    - Bayesian Graph Neural Networks with Adaptive Connection Sampling, *ICML 2020*
    - An Anatomy of Graph Neural Networks Going Deep via the Lens of Mutual Information: Exponential Decay vs. Full Preservation, *arXiv 1910.04499*
    - Revisiting Oversmoothing in Deep GCNs, *arXiv 2003.13663*
    - Towards Deeper Graph Neural Networks with Differentiable Group Normalization, *arXiv 2006.06972*
    - DeeperGCN: All You Need to Train Deeper GCNs, *arXiv 2006.07739*
    - Effective Training Strategies for Deep Graph Neural Networks, *arXiv 2006.07107*
    - Graphs, Entities, and Step Mixture, *arXiv 2005.08485*

Cautious: whether and when GNNs need to go deep?

- Cautious: whether and when do we need deep GNNs?
  - A wider neighborhood structure? The extra non-linearity?
- Recall Simplified GCN[1] formulation:

$$H^{(k+1)} = SH^{(k)}W^{(k)},$$

$$\hat{Y} = softmax(S^K H^{(0)} W)$$

High-order proximity

- There are also recent theoretical results[2] :

problem	*bound*	*problem*	*bound*
cycle detection (odd)	$dw = \Omega(n/\log n)$	shortest path	$d\sqrt{w} = \Omega(\sqrt{n}/\log n)$
cycle detection (even)	$dw = \Omega(\sqrt{n}/\log n)$	max. indep. set	$dw = \Omega(n^2/\log^2 n)$ for $w = O(1)$
subgraph verification*	$d\sqrt{w} = \Omega(\sqrt{n}/\log n)$	min. vertex cover	$dw = \Omega(n^2/\log^2 n)$ for $w = O(1)$
min. spanning tree	$d\sqrt{w} = \Omega(\sqrt{n}/\log n)$	perfect coloring	$dw = \Omega(n^2/\log^2 n)$ for $w = O(1)$
min. cut	$d\sqrt{w} = \Omega(\sqrt{n}/\log n)$	girth 2-approx.	$dw = \Omega(\sqrt{n}/\log n)$
diam. computation	$dw = \Omega(n/\log n)$	diam. 3/2-approx.	$dw = \Omega(\sqrt{n}/\log n)$

Simplifying graph convolutional networks, *ICML 2019*
What graph neural networks cannot learn depth vs width, *ICLR 2020*

Outline

- Does GNN fuse *feature* and *topology* optimally?
- Is GNN really a *deep* model?
- Technical challenges in real applications: robustness, explainability and applicability

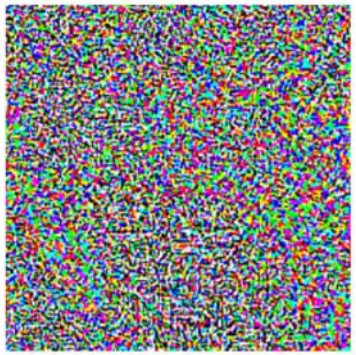
Technical challenges in real applications

Research → **Application**

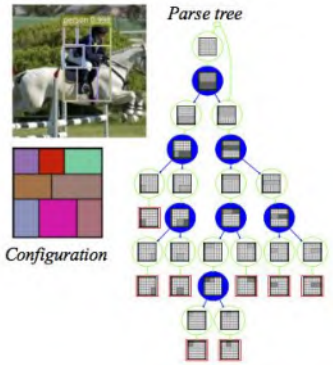
Robustness | Interpretability | Applicability

Hot directions in computer vision:

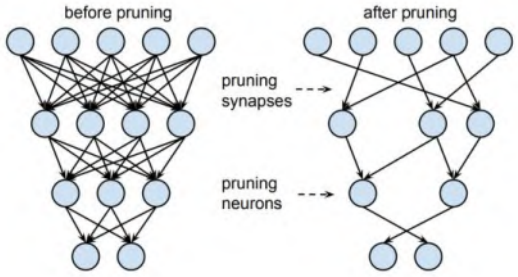
Adversarial



Explainable

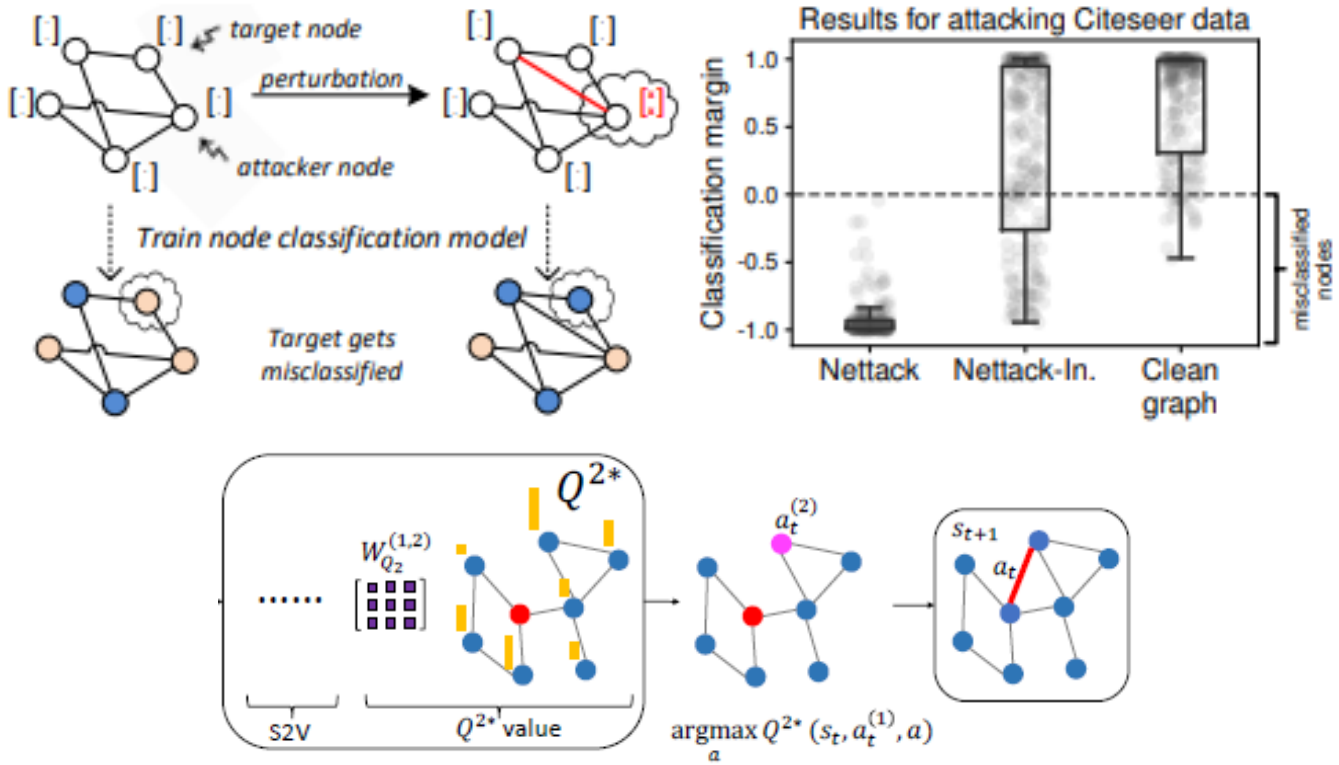


Scalable



Robustness in GNNs

- **Adversarial attacks**
  - Small perturbations in graph structures and node attributes
  - Great challenges for applying GNNs to node classification



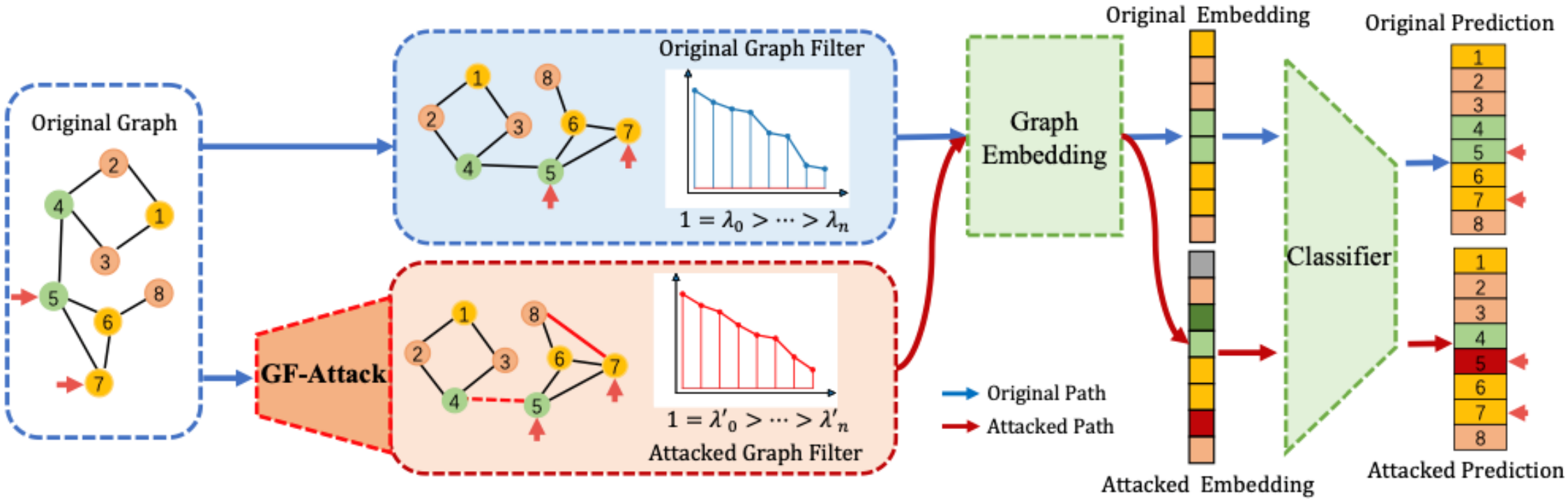
Adversarial Attacks on Neural Networks for Graph Data, *KDD 2018*

Adversarial Attacks on GNNs

- Categories
  - Targeted vs. Non-targeted
    - Targeted: the attacker focus on misclassifying some target nodes
    - Non-targeted: the attacker aims to reduce the overall model performance
  - Direct vs. Influence
    - Direct: the attacker can directly manipulate the edges/features of the target nodes
    - Influence: the attacker can only manipulate other nodes except the targets
  - Attacker knowledge:

Settings	Parameters	Predictions	Labels	Training Input
White-**B**ox **A**ttack (WBA)	✓	✓	✓	✓
Practical **W**hite-box **A**ttack (PWA)		✓	✓	✓
Restrict **B**lack-box **A**ttack (RBA)				✓

GF-Attack: Restrict Black-box Attack for Graphs



- Understand the various graph embedding model from a new general Graph Signal Processing (GSP) perspective.
- An adversarial framework, *GF-Attack*, is proposed accordingly to perform attacks though graph filters in a RBA fashion.

Heng Chang, Yu Rong, Tingyang Xu, Wenbing Huang, Honglei Zhang, Peng Cui, Wenwu Zhu, Junzhou Huang
A Restricted Black-box Adversarial Framework Towards Attacking Graph Embedding Models. *AAAI, 2020.*

GF-Attack: Restrict Black-box Attack for Graphs

- **Core idea**: constructing the attack damage measuring by attacking the graph filter $\mathcal{H}$
  - Measuring the quality of output embedding Z as the T-rank approximation problem:

$$\mathscr{L}(A', Z) = \|h(S')X - h(S')_T X\|_F^2,$$

  - Equivalent to optimize:

$$\arg\max_{A'} \sum_{i=T+1}^{n} {\lambda'_i}^2 \cdot \sum_{i=T+1}^{n} \|\mathbf{u}_i^T X\|_2^2,$$
$$\text{s.t. } \|A' - A\| = 2\beta.$$

λ_i and $\boldsymbol{u}_i$ are an eigen-pair of graph filter $\mathcal{H}$.

- **GCN/SGC as Example**: rewrite the GCN/SGC with $S = 2I_n - L^{sym}$:

$$\tilde{X} = (2I_n - L^{sym})^K X, \quad X' = \sigma(\tilde{X}\Theta), L^{sym} = I_n - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} = I_n - \hat{A}$$

- The corresponding adversarial attack loss for K_{th} order GCN/SGC is constructed as:

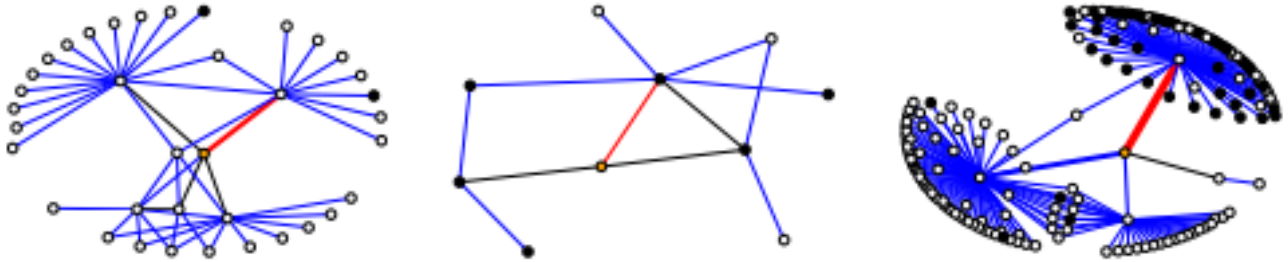
$$\arg\max_{A'} \sum_{i=T+1}^{n} (\lambda'_{\hat{A}',i} + 1)^{2K} \cdot \sum_{i=T+1}^{n} \|\mathbf{u}_{\hat{A}',i}^T X\|_2^2$$

- A linear time approximation via eigenvalue perturbation theory is used

Heng Chang, Yu Rong, Tingyang Xu, Wenbing Huang, Honglei Zhang, Peng Cui, Wenwu Zhu, Junzhou Huang
A Restricted Black-box Adversarial Framework Towards Attacking Graph Embedding Models. *AAAI, 2020.*

Robust Graph Convolutional Networks

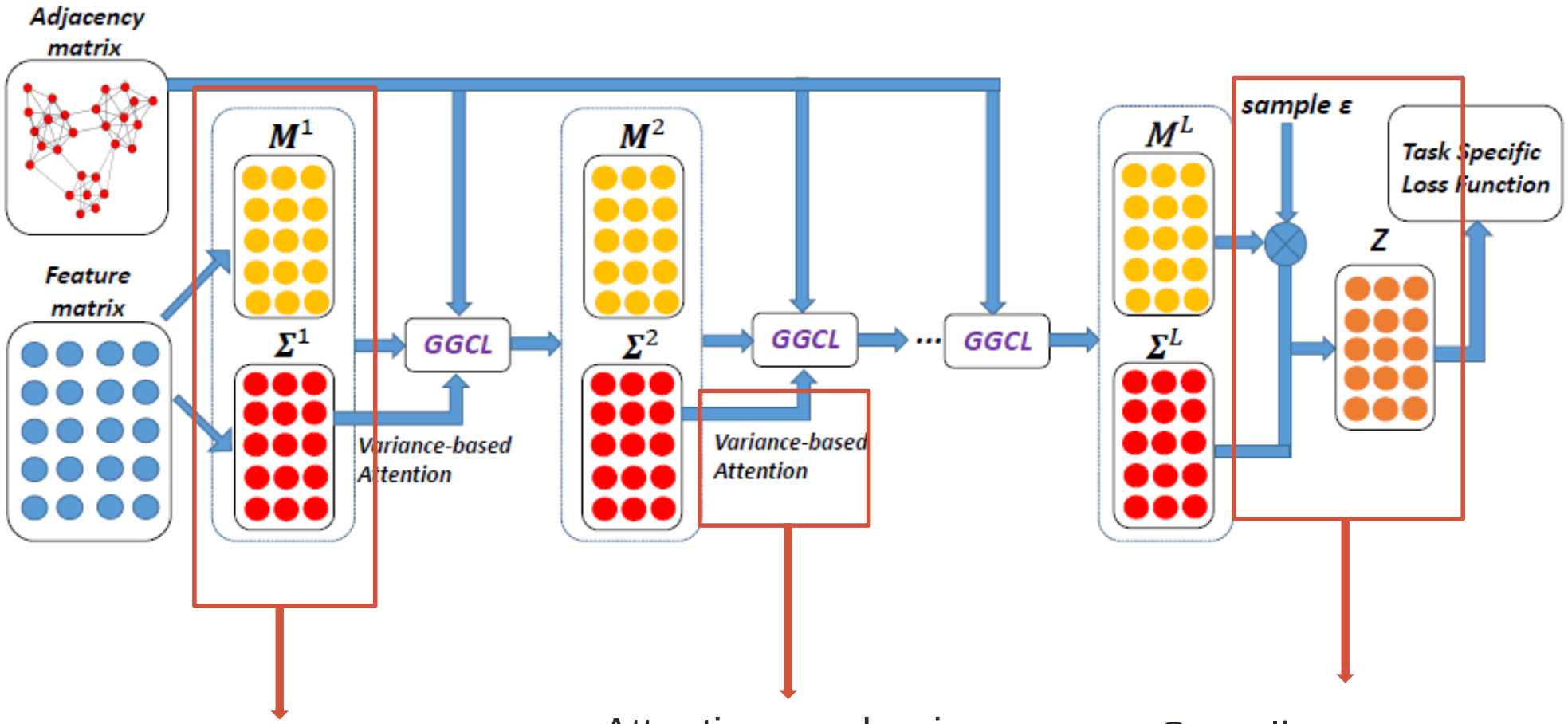
- How to enhance the robustness of GNNs against adversarial attacks?
- Adversarial attacks in node classification
  - Connect nodes from different communities to confuse the classifier



- Distribution vs. plain vectors
  - Plain vectors cannot adapt to such changes
  - Variances can help to absorb the effects of adversarial changes
  - Gaussian distributions → Hidden representations of nodes

Dingyuan Zhu, Ziwei Zhang, Peng Cui, Wenwu Zhu. Robust Graph Convolutional Networks Against Adversarial Attacks. ***KDD***, 2019.

The Framework of RGCN



Gaussian based representations: variance terms absorb the effects of adversarial attacks

Attention mechanism: Remedy the propagation of adversarial attacks

Sampling process: explicitly considers mathematical relevance between means and variances

Dingyuan Zhu, Ziwei Zhang, Peng Cui, Wenwu Zhu. Robust Graph Convolutional Networks Against Adversarial Attacks. ***KDD***, 2019.

Experimental Results

- Node Classification on Clean Datasets

	Cora	Citeseer	Pubmed
GCN	81.5	70.9	79.0
GAT	83.0	**72.5**	79.0
RGCN	**83.1**	71.3	**79.2**

- Against Non-targeted Adversarial Attacks

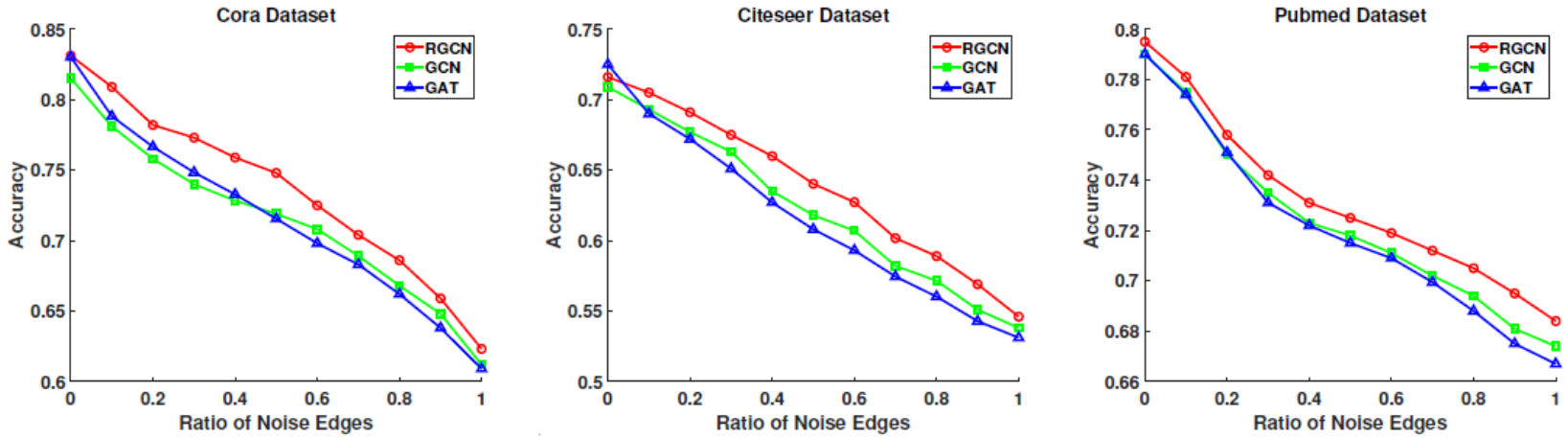
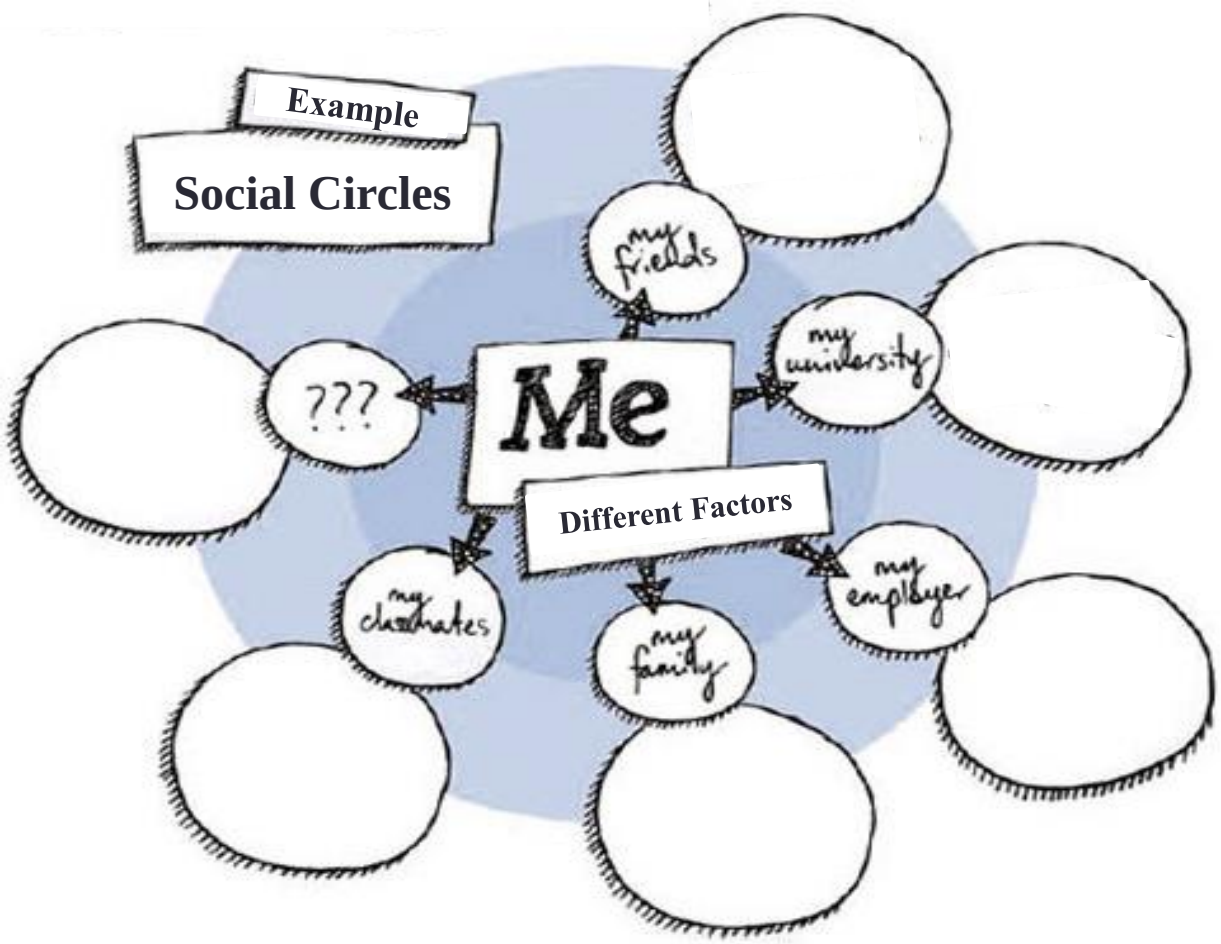


Figure 2: Results of different methods when adopting Random Attack as the attack method.

Dingyuan Zhu, Ziwei Zhang, Peng Cui, Wenwu Zhu. Robust Graph Convolutional Networks Against Adversarial Attacks. ***KDD***, 2019.

Interpretability of GNNs

- A real-world graph is typically formed due to *many* latent factors.



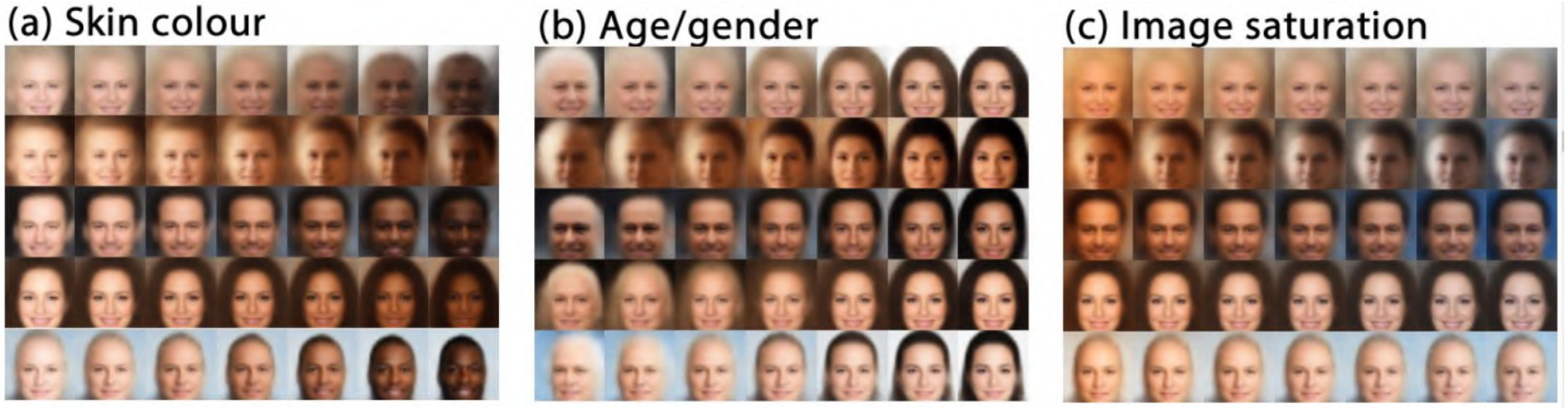
- Existing GNNs/GCNs:
  - A holistic approach, that takes in the *whole* neighborhood to produce a *single* node representation

- We suggest:
  - To disentangle the latent factors

  (By segmenting the heterogeneous parts, and learning multiple factor-specific representations for a node)
  - Robustness (e.g., not overreact to an irrelevant factor) & Interpretability

Disentangled Representation Learning

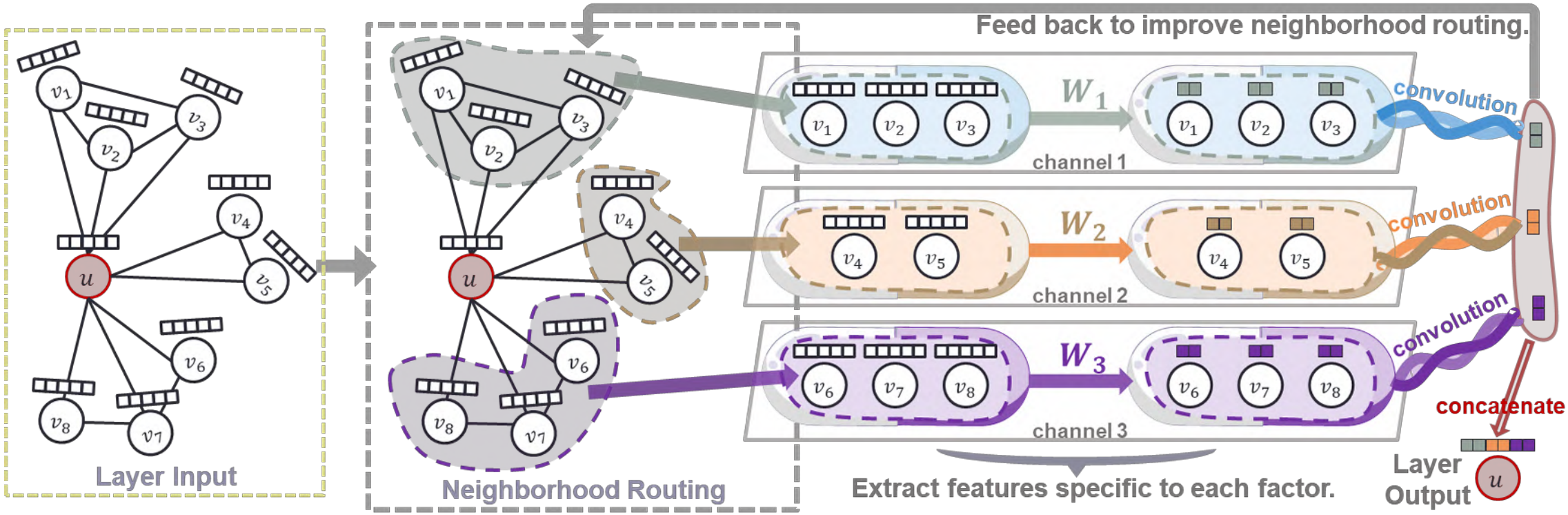
- That is, we aim to learn disentangled node representations
  - A representation that contains independent components, that describes different aspects (caused by different latent factors) of the observation
- The topic is well studied in the field of computer vision
  - But largely unexplored in the literature of GNNs.



Example: Three dimensions that are related skin color, age/gender, and saturation, respectively.

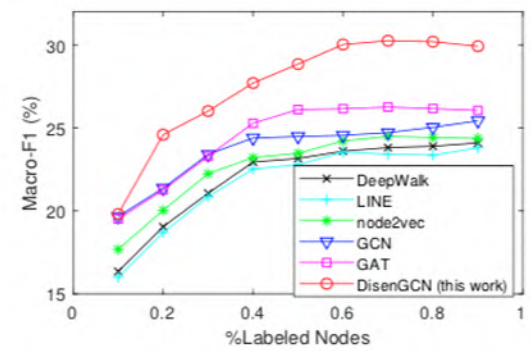
Method Overview

- We present DisenGCN, the *disentangled* graph convolutional network
  - DisenConv, a disentangled multichannel convolutional layer (figure below)
  - Each channel convolutes features related with a single latent factor

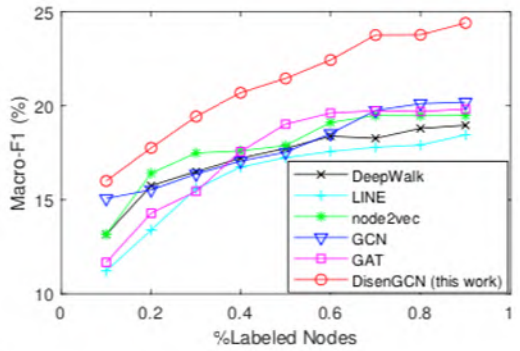


Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, Wenwu Zhu. Disentangled Graph Convolutional Networks. ***ICML***, 2019.

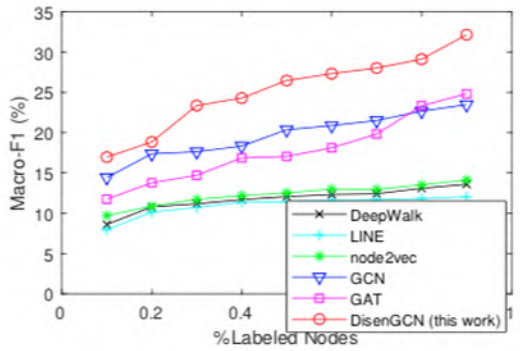
Results: Multi-label Classification



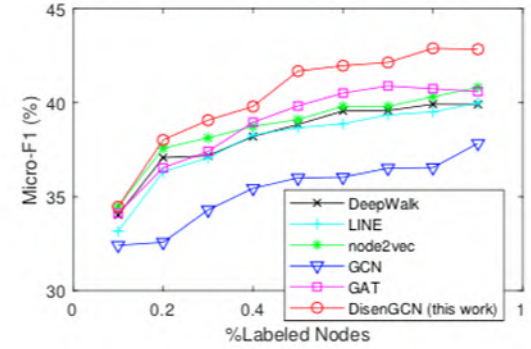
(a) Macro-F1(%), BlogCatalog.



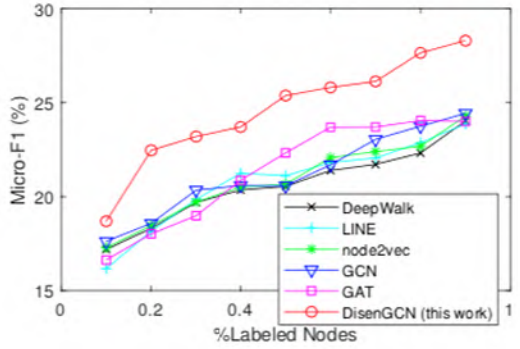
(c) Macro-F1(%), PPI.



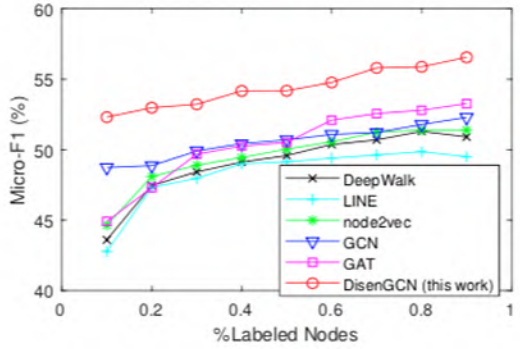
(e) Macro-F1(%), POS.



(b) Micro-F1(%), BlogCatalog.



(d) Micro-F1(%), PPI.



(f) Micro-F1(%), POS.

Figure 2. Macro-F1 and Micro-F1 scores on the multi-label classification tasks. Our approach consistently outperforms the best performing baselines by a large margin, reaching 10% to 20% relative improvement in most cases.

Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, Wenwu Zhu. Disentangled Graph Convolutional Networks. ***ICML***, 2019.

Results: On Synthetic Graphs

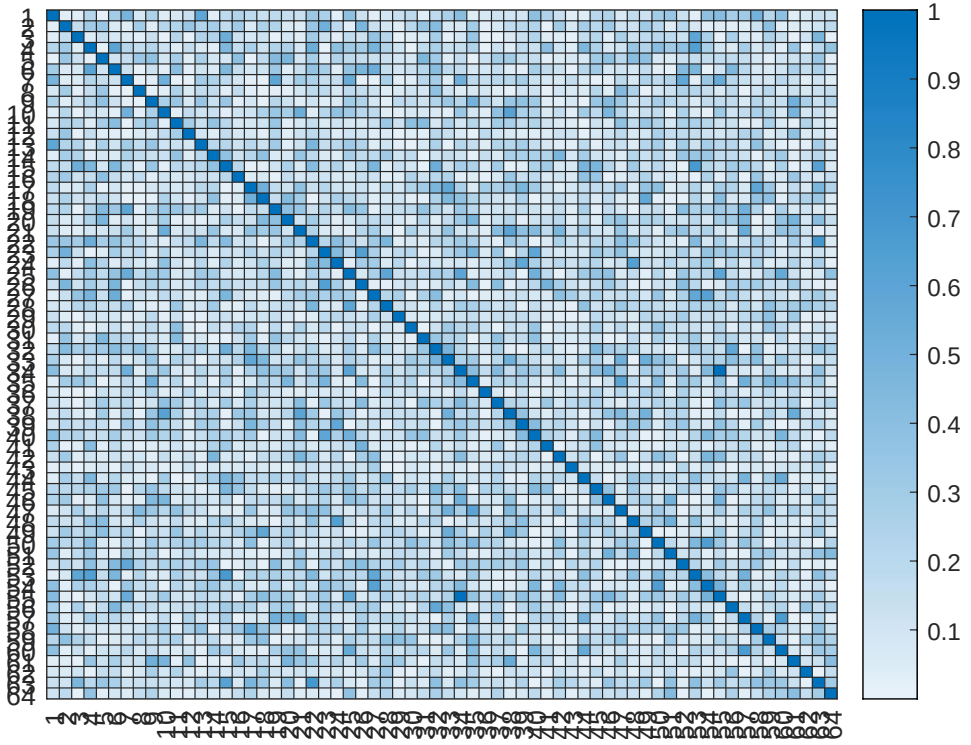
Table 3. Micro-F1 scores on synthetic graphs generated with different numbers of latent factors.

	Number of latent factors						
Method	4	6	8	10	12	14	16
GCN	78.78 ± 1.52	65.73 ± 1.94	46.55 ± 1.55	37.37 ± 1.52	24.49 ± 1.03	18.14 ± 1.50	16.43 ± 0.92
GAT	83.77 ± 2.32	60.89 ± 3.75	45.88 ± 3.79	36.72 ± 3.58	24.77 ± 3.47	20.89 ± 3.57	19.53 ± 3.97
DisenGCN (this work)	**93.84** ± 1.12	**74.68** ± 1.92	**54.57** ± 1.79	**43.96** ± 1.45	**28.17** ± 1.22	**23.57** ± 1.28	**21.99** ± 1.34
Relative improvement	+12.02%	+13.62%	+17.23%	+17.63%	+13.73%	+12.83%	+12.6%

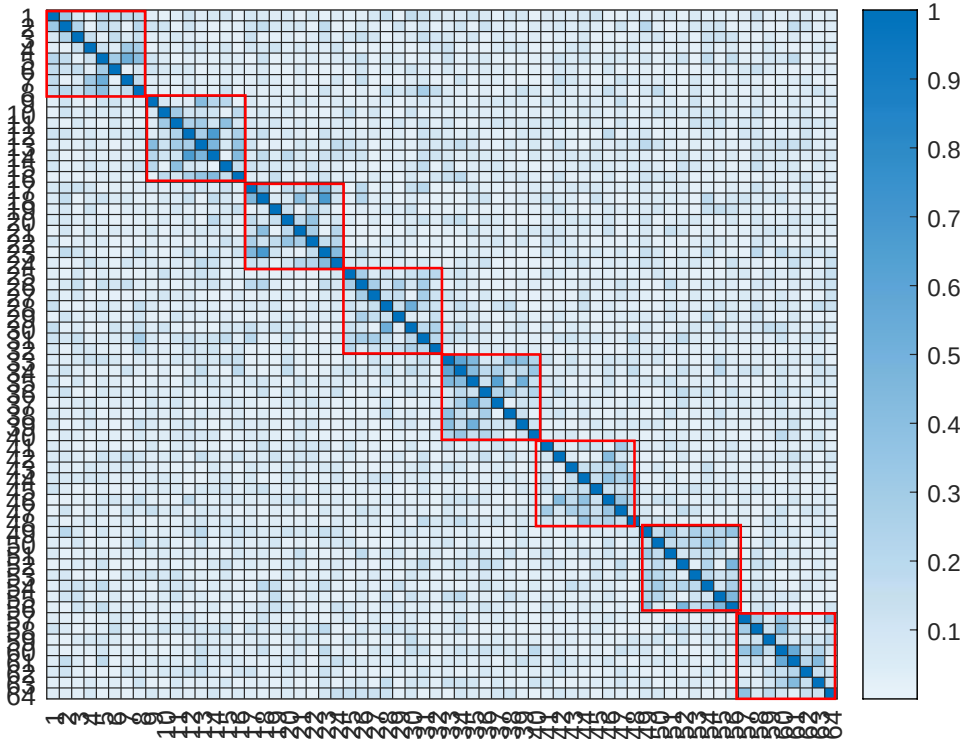
- Improvement is larger when #factors is relatively large (around 8)

Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, Wenwu Zhu. Disentangled Graph Convolutional Networks. ***ICML***, 2019.

Results: Correlations between the Neurons



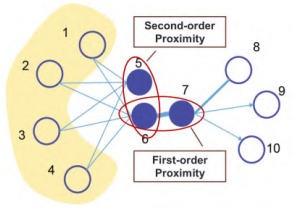
(a) GCN.



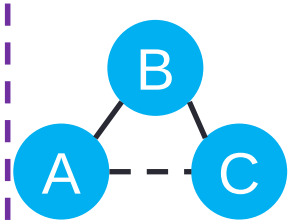
(b) DisenGCN (this work).

Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, Wenwu Zhu. Disentangled Graph Convolutional Networks. ***ICML***, 2019.

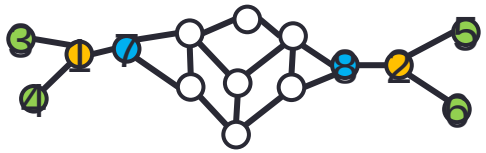
Applicability of GNNs/Network embedding



High-order



Transitivity



Global position

- Link Prediction
- Community Detection
- Node Classification
- Network Distance
- Node Importance
- …

Various network properties

Various applications

- Leading to a large number of hyper-parameters
- Must be carefully tuned

→ **AutoML**

AutoML

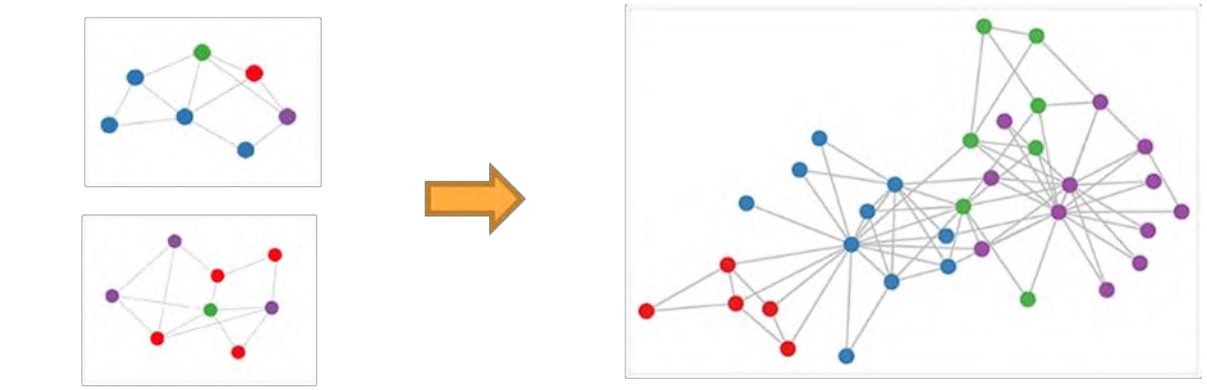
- Ease the adoption of machine learning and reduce the reliance on human experts
  - E.g., hyper-parameter optimization, neural architecture search
- Largely unexplored on graph/network data

- Large-scale issue:
  - Complexity of GNNs/Network Embedding is usually at least $O(E)$
    - E is the number of edges (can be 10 billion)
  - Total complexity: $O(ET)$, T is the times searching for optimal hyperparameters

How to incorporate AutoML into massive GNNs efficiently?

AutoML for GNN/Network embedding

- A straightforward way: configuration selection on sampled sub-networks



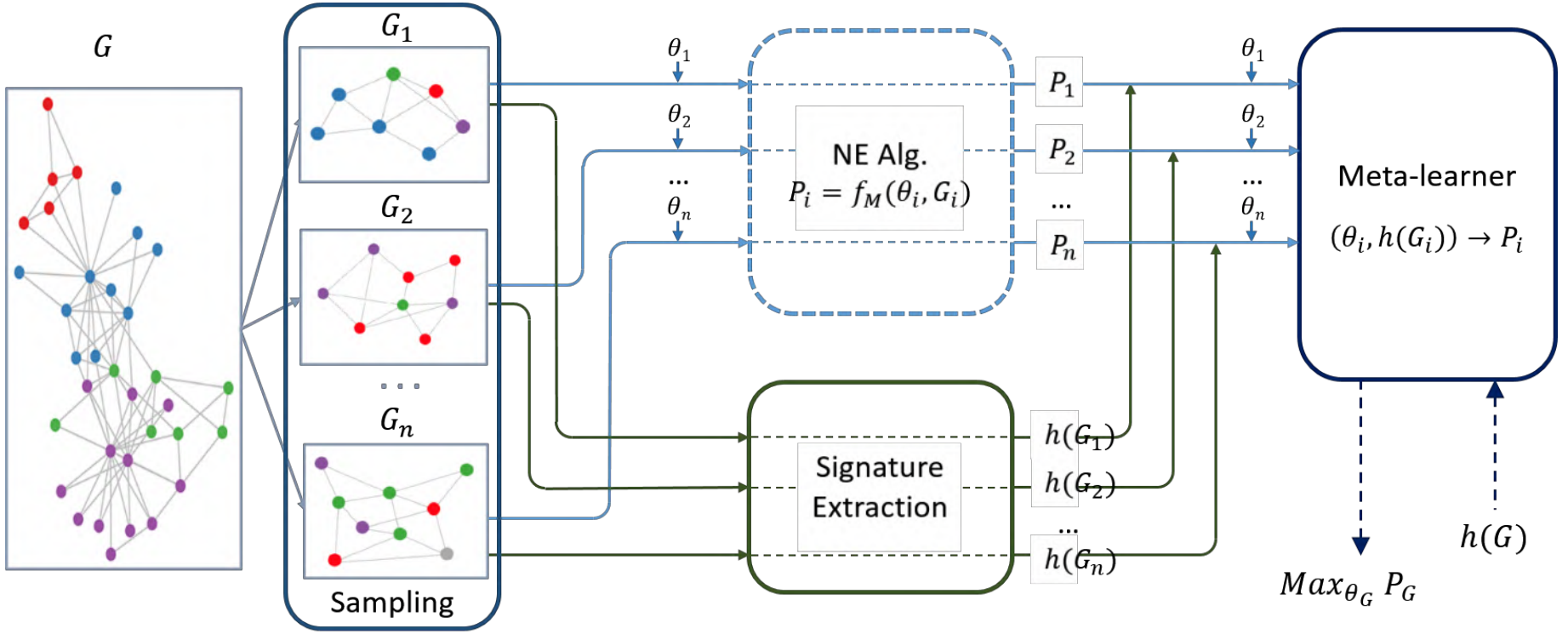
Sampled sub-network
Optimal configuration θ

Origin massive network

- Transferability
  - $\theta \neq$ optimal configuration on the origin network
- Heterogeneity
  - Several highly heterogeneous components → needs carefully designed sampling

Ke Tu, Jianxin Ma, Peng Cui, Jian Pei, Wenwu Zhu. AutoNE: Hyperparameter Optimization for Massive Network Embedding. ***KDD***, 2019.

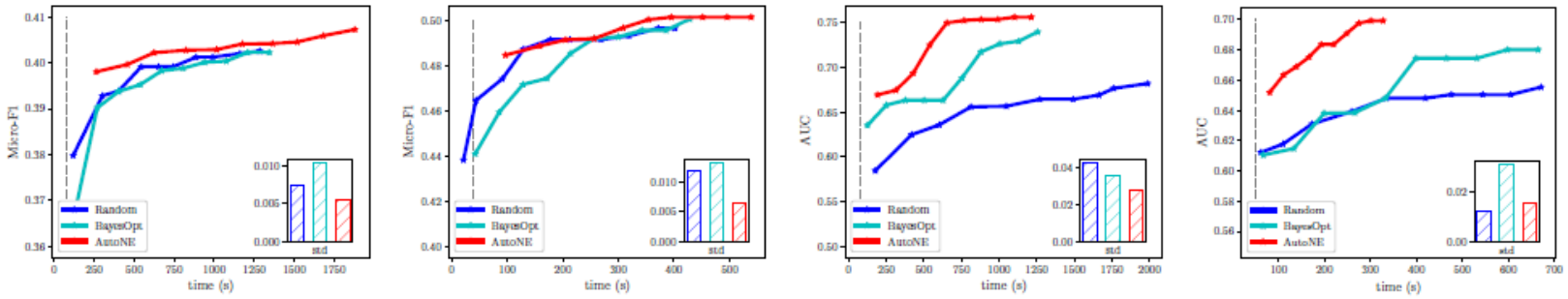
AutoNE



Transfer the knowledge about optimal hyperparameters from the sub-networks to the original massive network

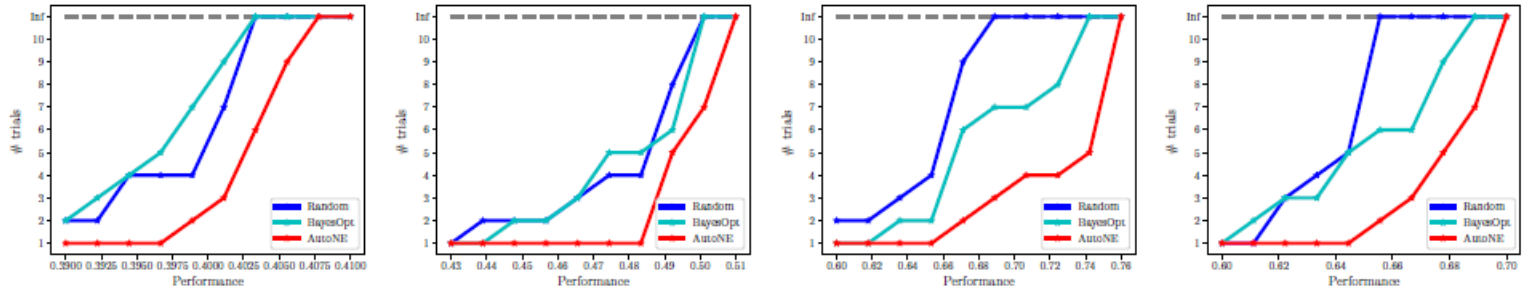
Ke Tu, Jianxin Ma, Peng Cui, Jian Pei, Wenwu Zhu. AutoNE: Hyperparameter Optimization for Massive Network Embedding. ***KDD***, 2019.

Experiments: Sampling-Based NE



(a) Classification on BlogCatalog (b) Classification on Wikipedia (c) Link prediction on BlogCatalog (d) Link prediction on Wikipedia

The performance achieved within various time thresholds.



(a) Classification on BlogCatalog (b) Classification on Wikipedia (c) Link prediction on BlogCatalog (d) Link prediction on Wikipedia

The number of trials to reach a certain performance threshold

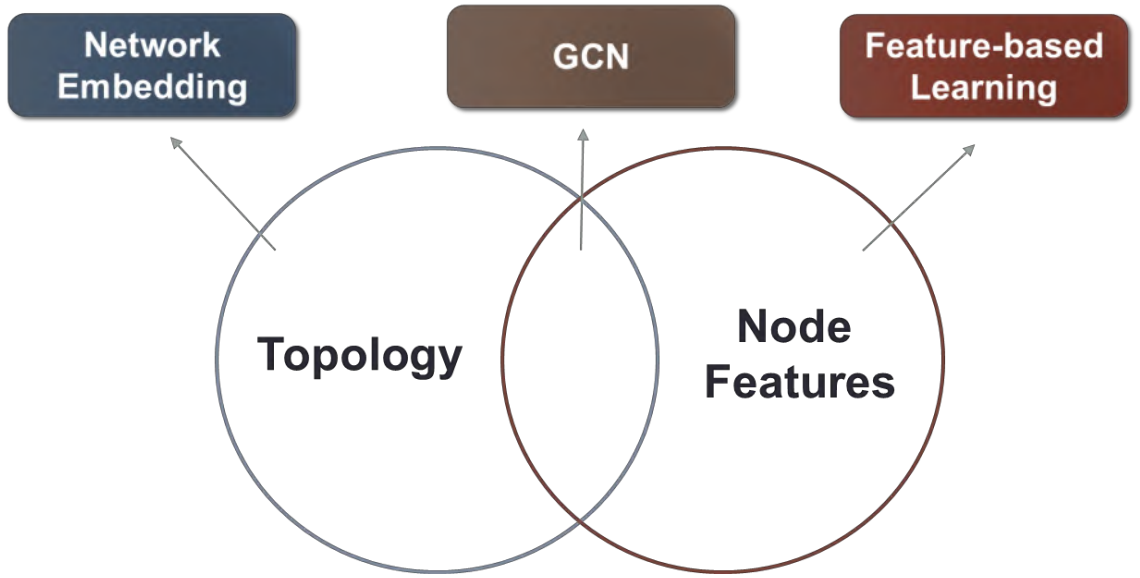
Ke Tu, Jianxin Ma, Peng Cui, Jian Pei, Wenwu Zhu. AutoNE: Hyperparameter Optimization for Massive Network Embedding. ***KDD***, 2019.

Recap: Graph Neural Networks

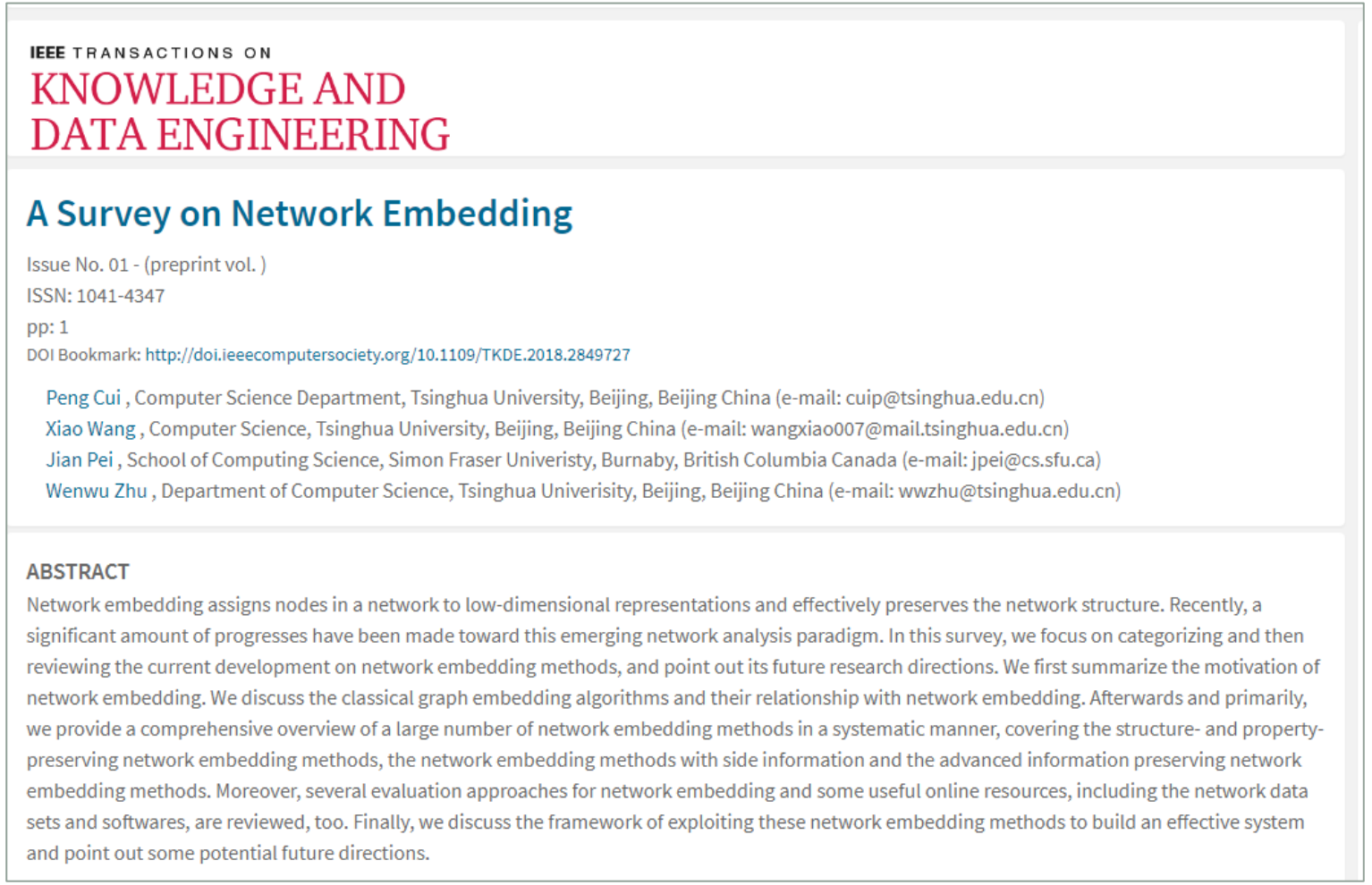
- Message-passing framework of GNNs
- Frontiers:
  - Does GNN fuse *feature* and *topology* optimally?
  - Is GNN really a *deep* model?
  - Technical challenges in real applications: robustness, explainability and applicability

Summaries and Conclusions

- Unsupervised vs. (Semi-)Supervised
- Learning for Networks vs. Learning via Graphs
- Topology-driven vs. Feature-driven
- Both GNN and NE need to treat the counterpart as the baselines



A Survey on Network Embedding



IEEE TRANSACTIONS ON
KNOWLEDGE AND
DATA ENGINEERING

A Survey on Network Embedding

Issue No. 01 - (preprint vol.)
ISSN: 1041-4347
pp: 1
DOI Bookmark: http://doi.ieeecomputersociety.org/10.1109/TKDE.2018.2849727

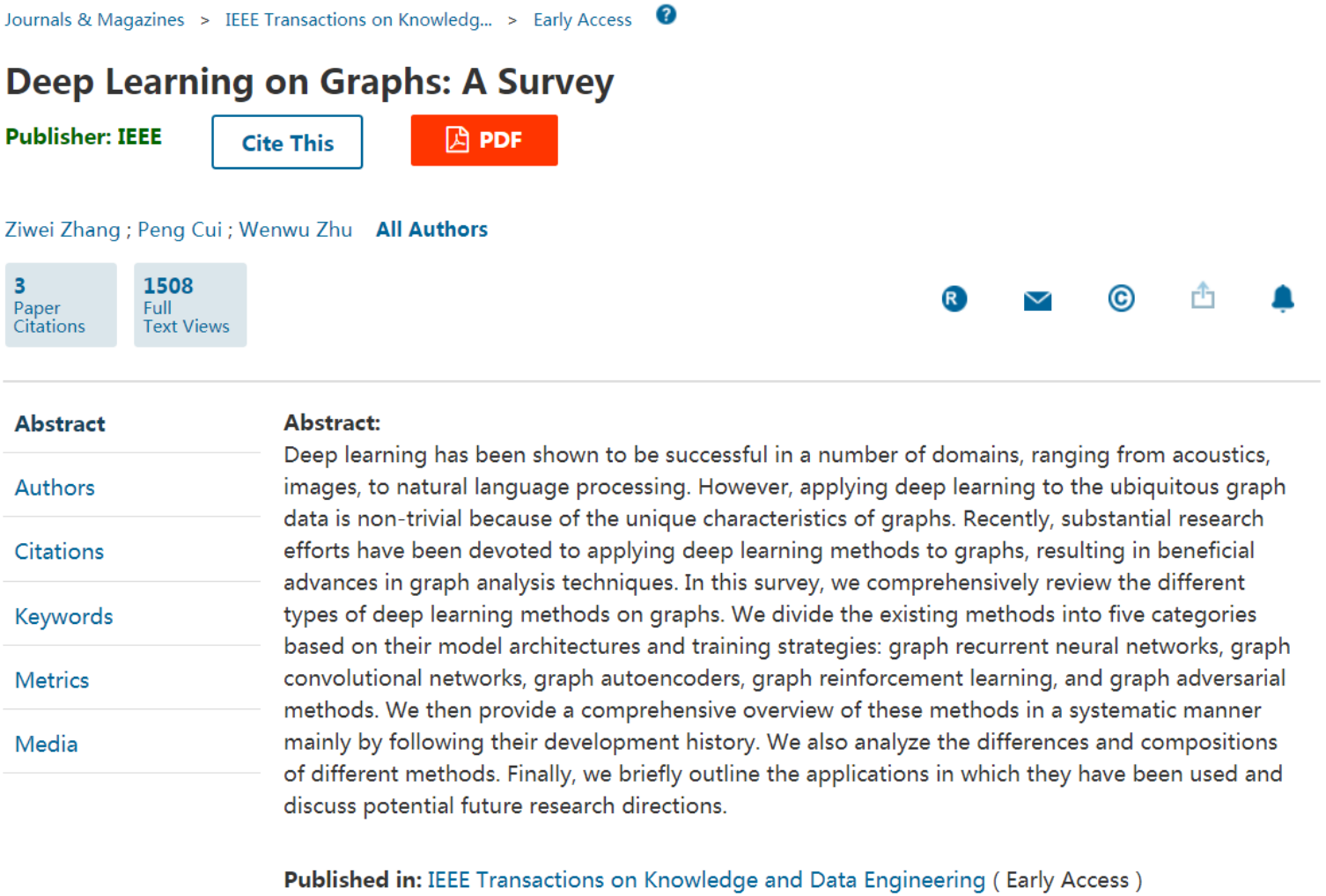
Peng Cui , Computer Science Department, Tsinghua University, Beijing, Beijing China (e-mail: cuip@tsinghua.edu.cn)
Xiao Wang , Computer Science, Tsinghua University, Beijing, Beijing China (e-mail: wangxiao007@mail.tsinghua.edu.cn)
Jian Pei , School of Computing Science, Simon Fraser Univeristy, Burnaby, British Columbia Canada (e-mail: jpei@cs.sfu.ca)
Wenwu Zhu , Department of Computer Science, Tsinghua Univerisity, Beijing, Beijing China (e-mail: wwzhu@tsinghua.edu.cn)

ABSTRACT

Network embedding assigns nodes in a network to low-dimensional representations and effectively preserves the network structure. Recently, a significant amount of progresses have been made toward this emerging network analysis paradigm. In this survey, we focus on categorizing and then reviewing the current development on network embedding methods, and point out its future research directions. We first summarize the motivation of network embedding. We discuss the classical graph embedding algorithms and their relationship with network embedding. Afterwards and primarily, we provide a comprehensive overview of a large number of network embedding methods in a systematic manner, covering the structure- and property-preserving network embedding methods, the network embedding methods with side information and the advanced information preserving network embedding methods. Moreover, several evaluation approaches for network embedding and some useful online resources, including the network data sets and softwares, are reviewed, too. Finally, we discuss the framework of exploiting these network embedding methods to build an effective system and point out some potential future directions.

Peng Cui, Xiao Wang, Jian Pei, Wenwu Zhu. **A Survey on Network Embedding**. *IEEE TKDE, 2018*.

Deep Learning on Graphs: A Survey



Ziwei Zhang, Peng Cui, Wenwu Zhu. **Deep Learning on Graphs: A Survey**. *IEEE TKDE, 2020.*

Thanks!

Peng Cui

cuip@tsinghua.edu.cn

http://pengcui.thumedialab.com