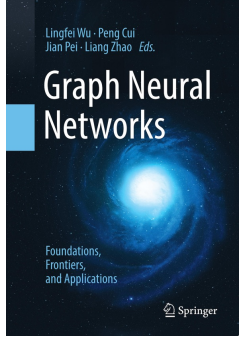
Pinterest




# Graph Neural Networks: Foundations, Frontiers, Applications

**Lingfei Wu**

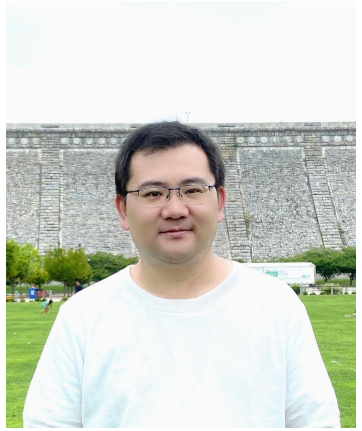
Joint Work with GNNs Team

(Prof. Peng Cui, Prof. Jian Pei and Prof. Liang Zhao)

Discord ML Distinguished Speaker Series

2022-10-06

# About Me



**Lingfei Wu**

**2010**
M.S in Automation
USTC

**2014**
Intern in LBNL

**2016**
Ph.D. in CS at College of William & Mary

**2016-2021**
Lead Researcher & Master Inventor at IBM T. J. Watson Res Center

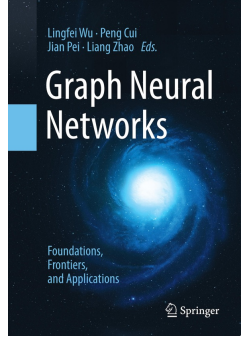
- Graph Neural Networks
- Natural Language Processing
- 90+ Papers
- 40+ Patents (4 high-value)
- 4 Best Papers Awards
- 4 Outstanding Technical Achievement Awards

**2021-2022**
Principal Scientist at JD.com Silicon Valley Res Center

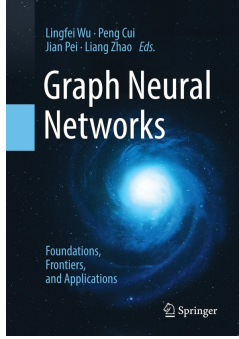
- Led and delivered 5 innovative products using ML/NLP/CV
- 1 book in GNNs
- 2 "IAAI Innovative AI Application Award" in AAAI'22
- Developed 3 "Best Star Employees" in JD.com

**2022-Present**
EM of Knowledge Signal Team

- To be written in Future ☺



# Outline




**GNNs Foundations**

- Time History of GNNs
- GNNs: Foundations and Models
- GNNs: Theory, Scalability, Interpretability

**GNNs Frontiers**

- Graph Structure Learning
- Other Frontiers in GNNs

**GNNs Applications**

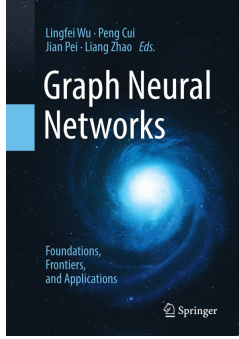
- GNNs in Recommendation System
- Other GNNs-based Applications

**GNN book website：**
https://graph-neural-networks.github.io/index.html

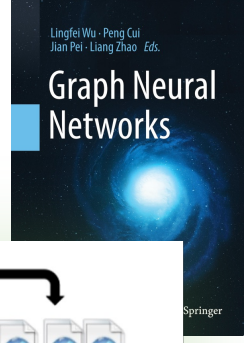
**GNN Springer：**
https://link.springer.com/book/10.1007/978-981-16-6054-2

**Amazon：**
https://www.amazon.com/Graph-Neural-Networks-Foundations-Applications/dp/9811660530

**JD.com (京东商城)：**
https://item.jd.com/10043589466641.html



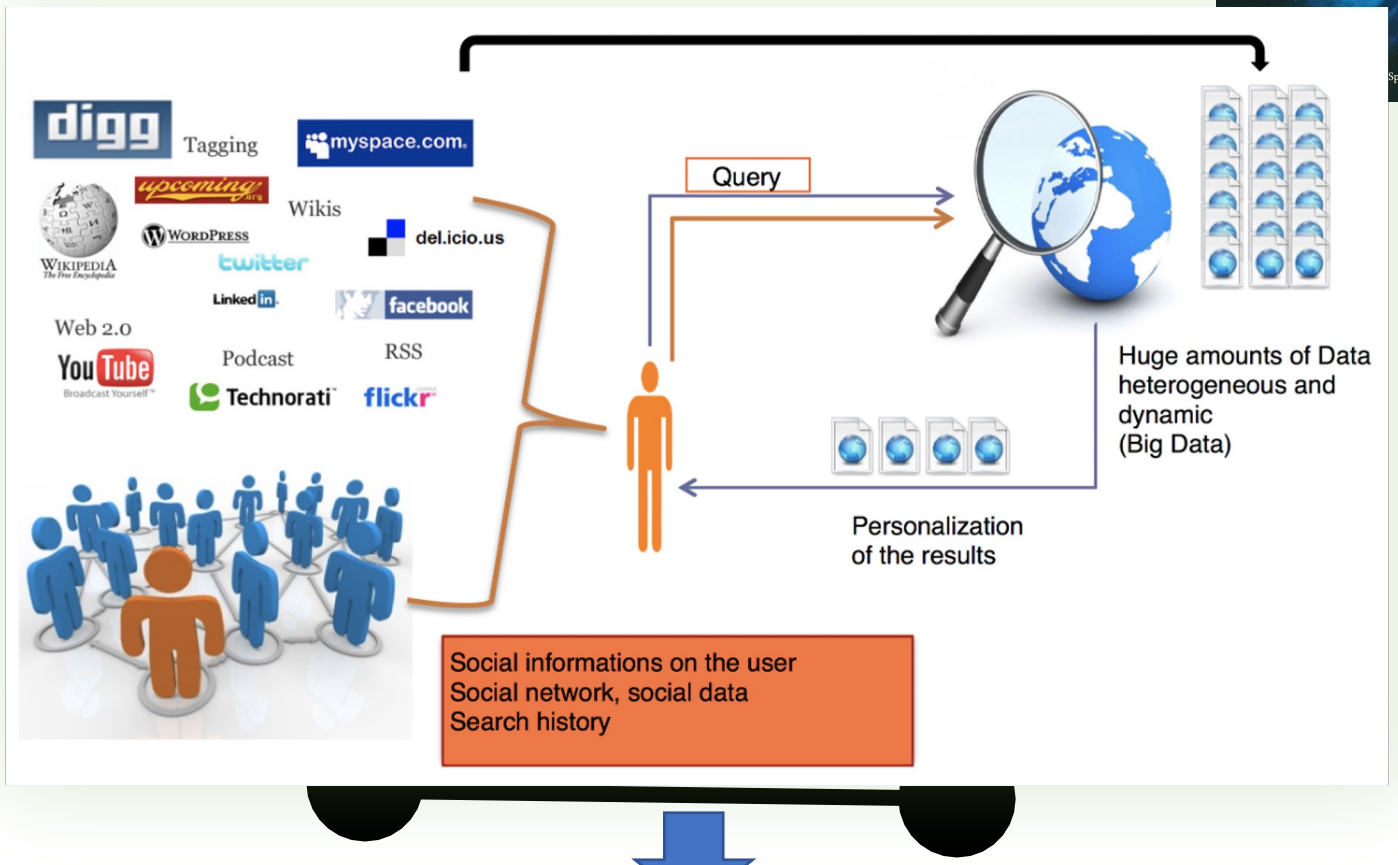

# GNNs
# Introduction




# Why graphs?

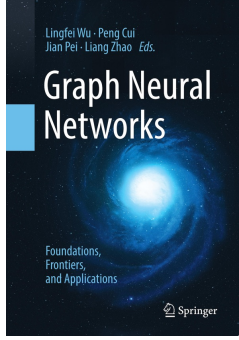
- Graphs are a general language for describing and modeling complex systems


Source from Laboratoire Hubert Curien - Université Jean Monnet

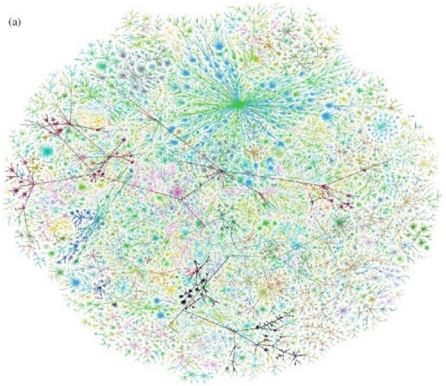



**Graph Topology + Node Attributes + Node Types**

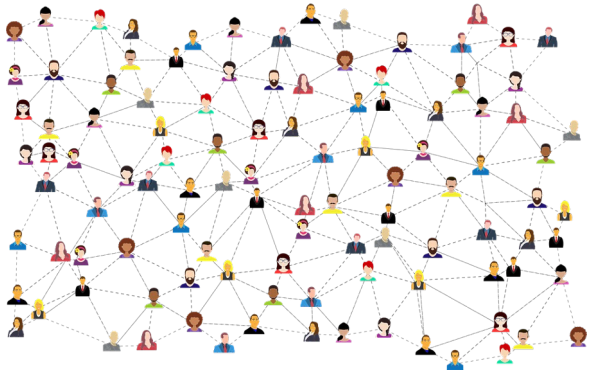
**Graph!**




# Graph-structured data are ubiquitous

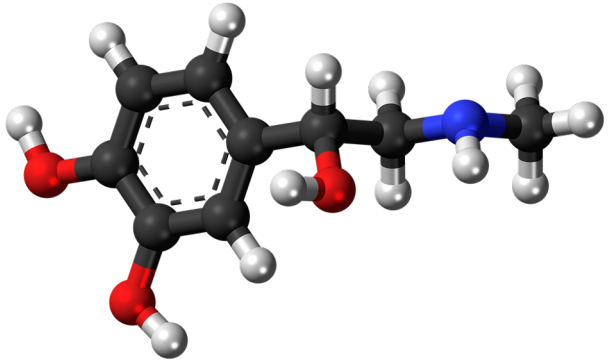


Internet

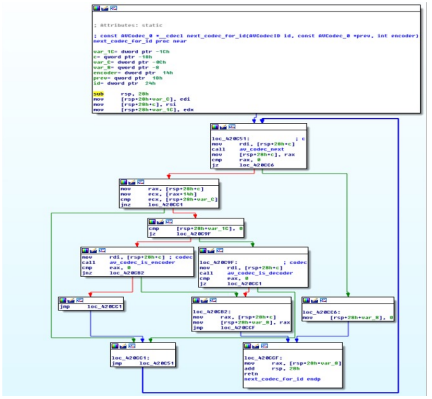


Social networks

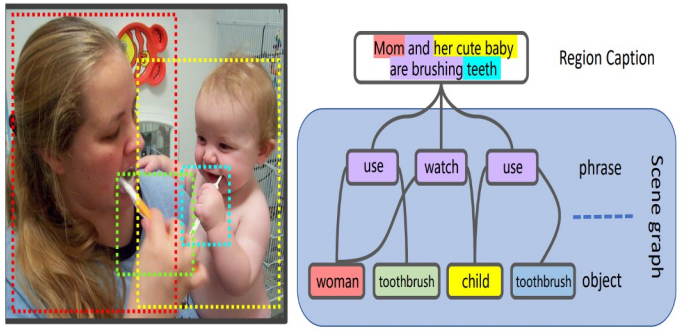
Information retrieval



Biomedical graphs

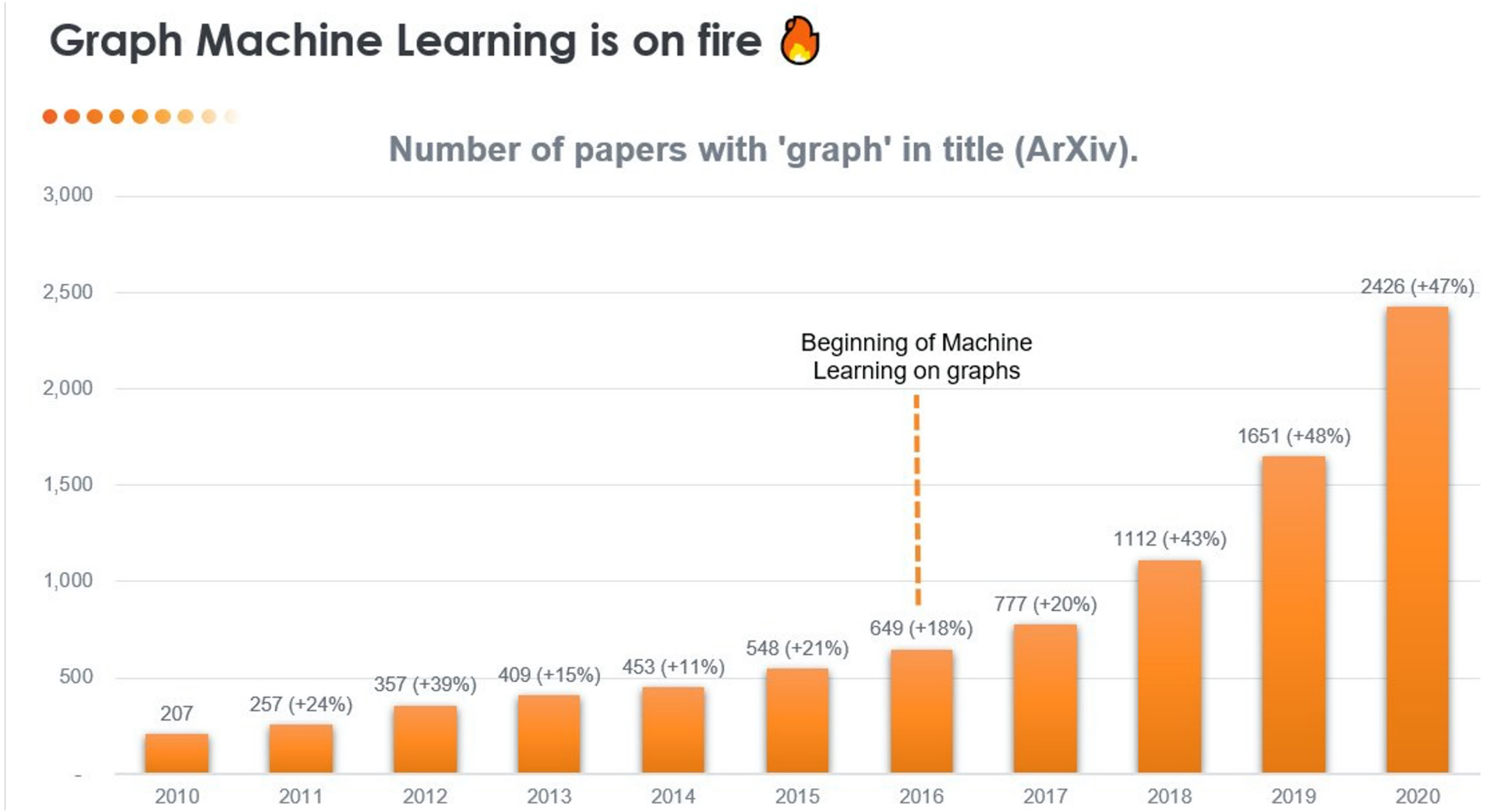


Program graphs

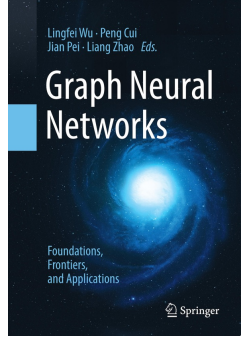



Scene graphs

# Graph Machine Learning: Recent Trending

# GNNs: A Brief History




**Year 2009**

First GNNs Paper (Scarselli et al., 2009)

**Year 2016, 2017**

First GRU-based Modern GNNs paper: GGNN

First graph convolution-based paper GCN, Start a new era of GNNs history

**Year 2017-2021**

A series of Graph Convolution (GCN), Message Passage (MPNN, GraphSage, GIN), Attention-based (GAT), Unsupervised GNNs (Graph-Autoencoder, graph-infomax)

Many new GNNs fast developed!

**Year 2019-2021**

GNNs–based applications, such as search, Recommendation, drug discovery, NLP, intelligent Transport…

Many open-source Githubs like DGL, Pytorch Geometric, Graph4NLP, TorchDrug…

**Year 2021**

3 GNN books released simultaneously:

1) Prof. Liu (Tsinghua) et al.: "Introduction to Graph Neural Networks"

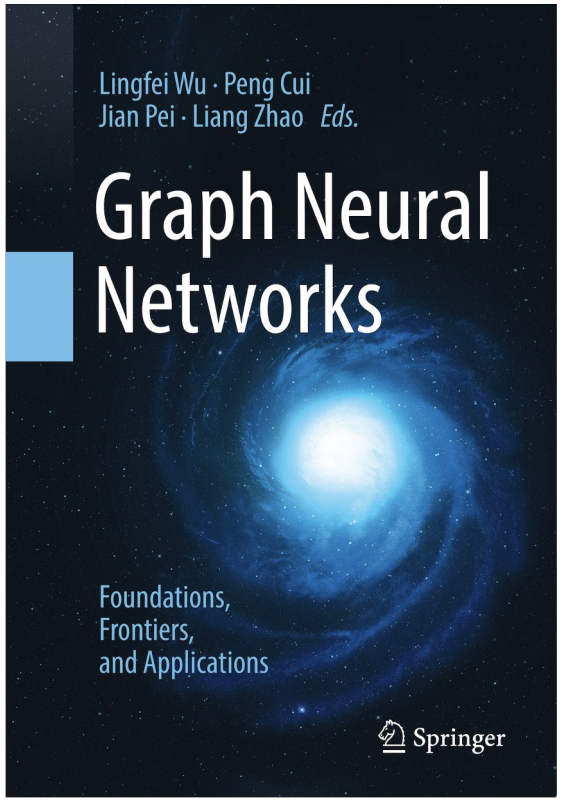
2) Prof. Tang (MSU) et al.: "Deep learning on graphs"

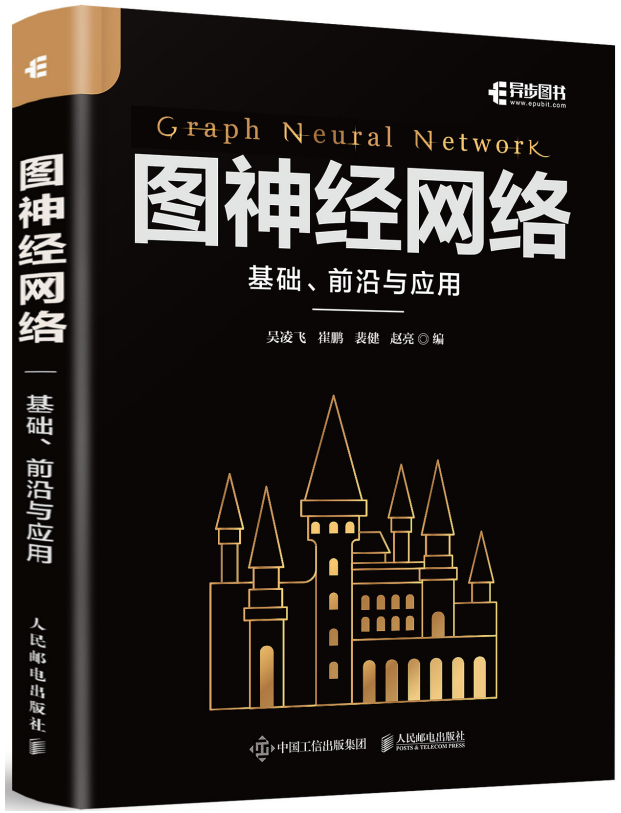
3) Prof. Hamilton (McGill): "Graph representation learning"

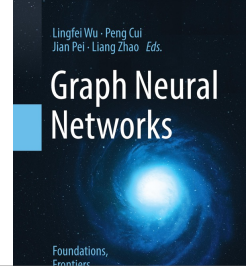
**Year 2022**

Most comprehensive GNN book: "Graph Neural Networks: Foundations, Frontiers, Applications" by Springer

# Graph Neural Networks Book @Springer2022



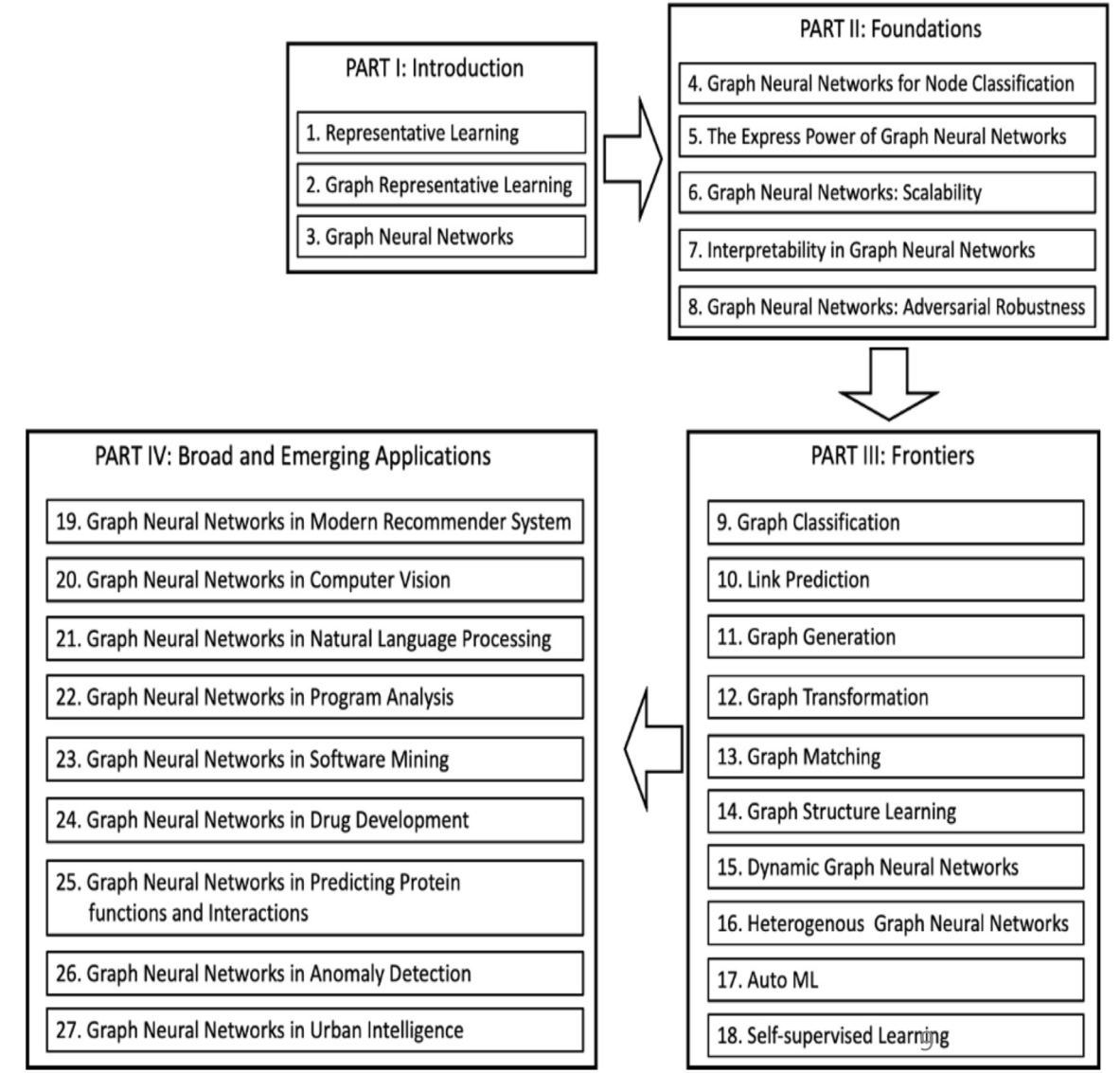



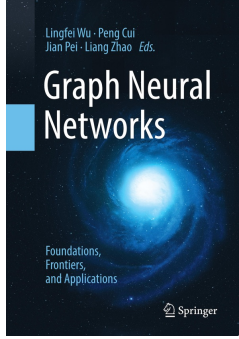




**Free download: https://graph-neural-networks.github.io/index.html**

The English version of the book is available for on Springer, Amazon and JD.COM !

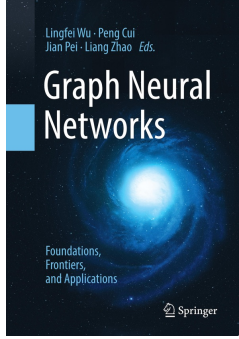
The Chinese version (Posts & Telecom Press, 人民邮电出版社) will be officially published in the end of 2022!!






# GNNs
# Foundations

# Machine(Deep) Learning with Graphs

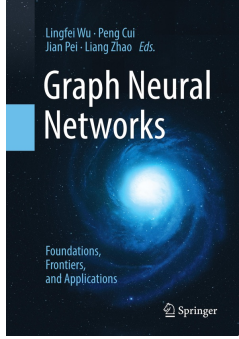



Classical ML tasks on graphs:

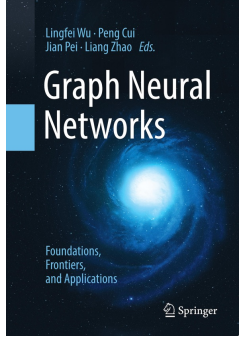
- Node classification
  - Predict a type of a given node
- Link prediction
  - Predict whether two nodes are linked
- Community detection
  - Identify densely linked clusters of nodes
- Graph similarity
  - How similar are two (sub)graphs

Recent ML/DL tasks on graphs:

- Expressive Power of GNNs
  - Theoretical understanding
- Scalability of GNNs
  - Sampling paradigms for scaling up
- Interpretability of GNNs
  - Interpretable modeling and explain
- Adversarial Robustness of GNNs
  - Adversarial attacks and provable robustness

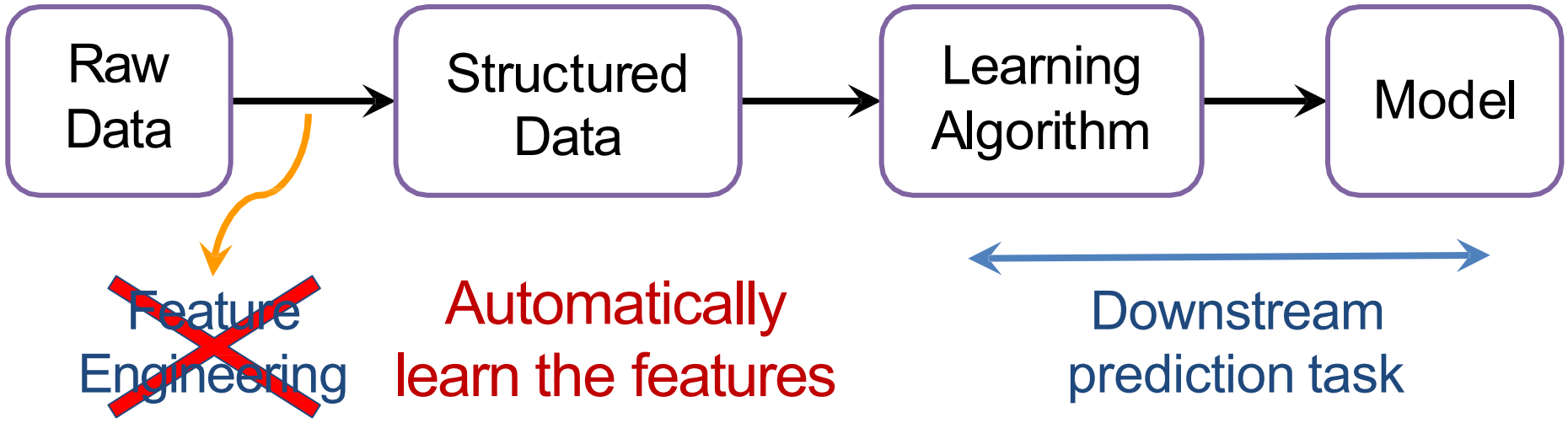


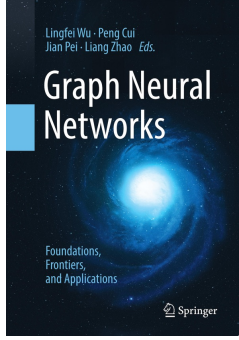

# GNNs for Node Classification (Chapter 4)




# Machine Learning Lifecycle

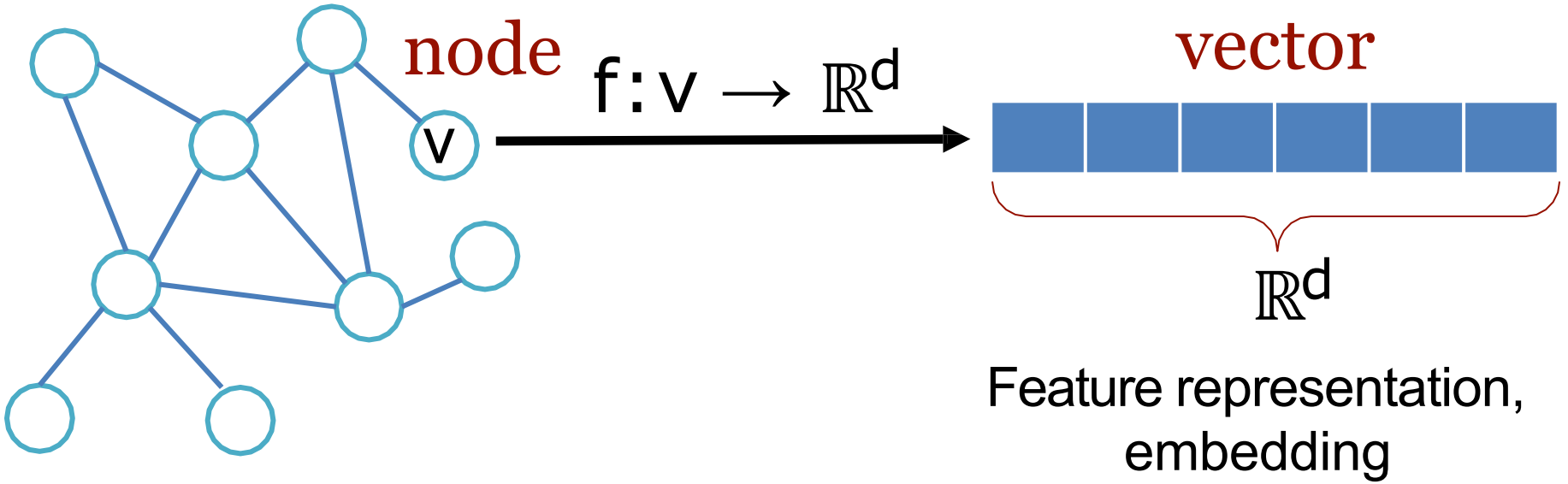
- (Supervised) Machine Learning Lifecycle: feature learning is the key

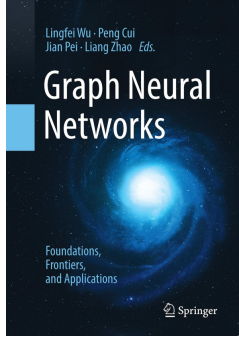





# Feature Learning in Graphs

- Our Goal: Design efficient task-independent/ task-dependent feature learning for machine learning in graphs!






# Graph Neural Networks: Foundations

- Learning node embeddings:

A graph filter  adjacency matrix

$$\mathbf{h}_i^{(l)} = f_{\text{filter}}(A, \mathbf{H}^{(l-1)})$$

Updated node embeddings  Input node embeddings

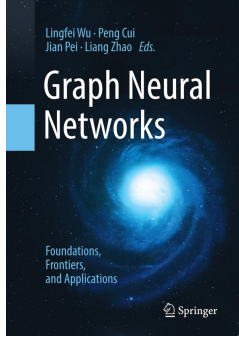
$f_{\text{filter}}(\cdot,\cdot)$
- Spectral-based
- Spatial-based
- Attention-based
- Recurrent-based

- Learning graph-level embeddings:

$$A', \mathbf{H}' = f_{\text{pool}}(A, \mathbf{H})$$

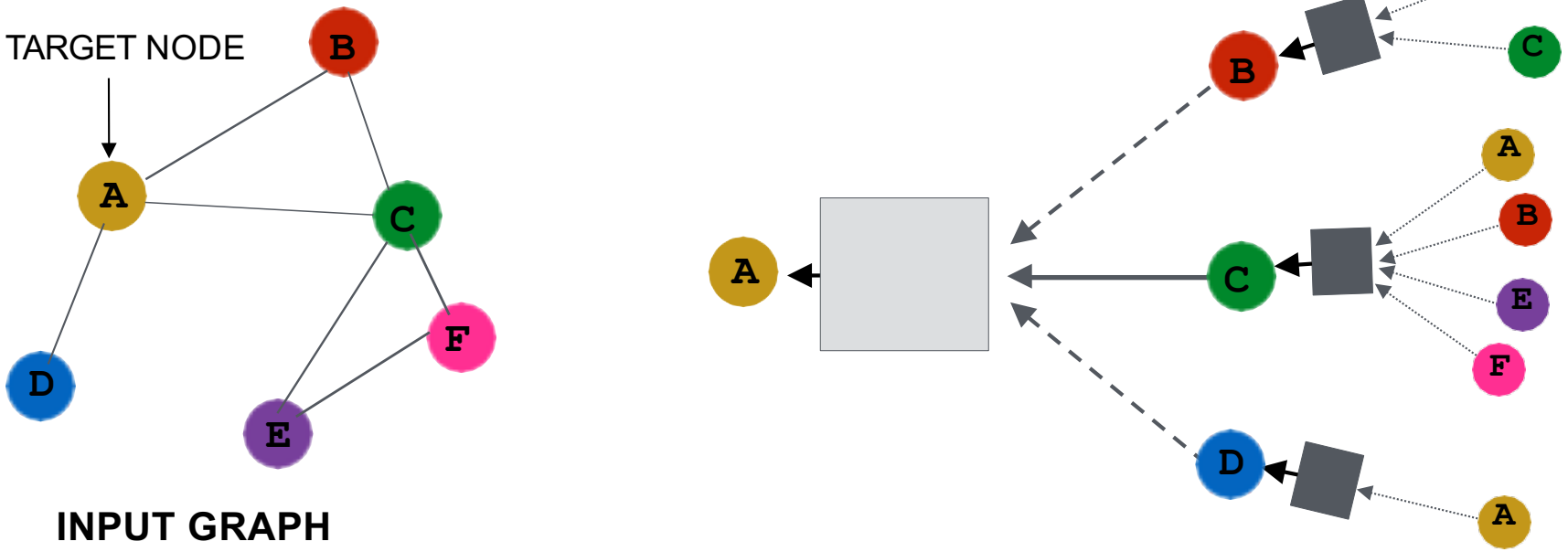
A small graph w/ fewer nodes  New node embeddings  Input graph  Input node embeddings

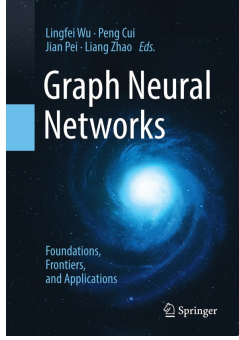
$f_{\text{pool}}(\cdot,\cdot)$
- Flat Graph Pooling (i.e. Max, Ave, Min)
- Hierarchical Graph Pooling (i.e. Diffpool)




# Graph Neural Networks: Basic Model

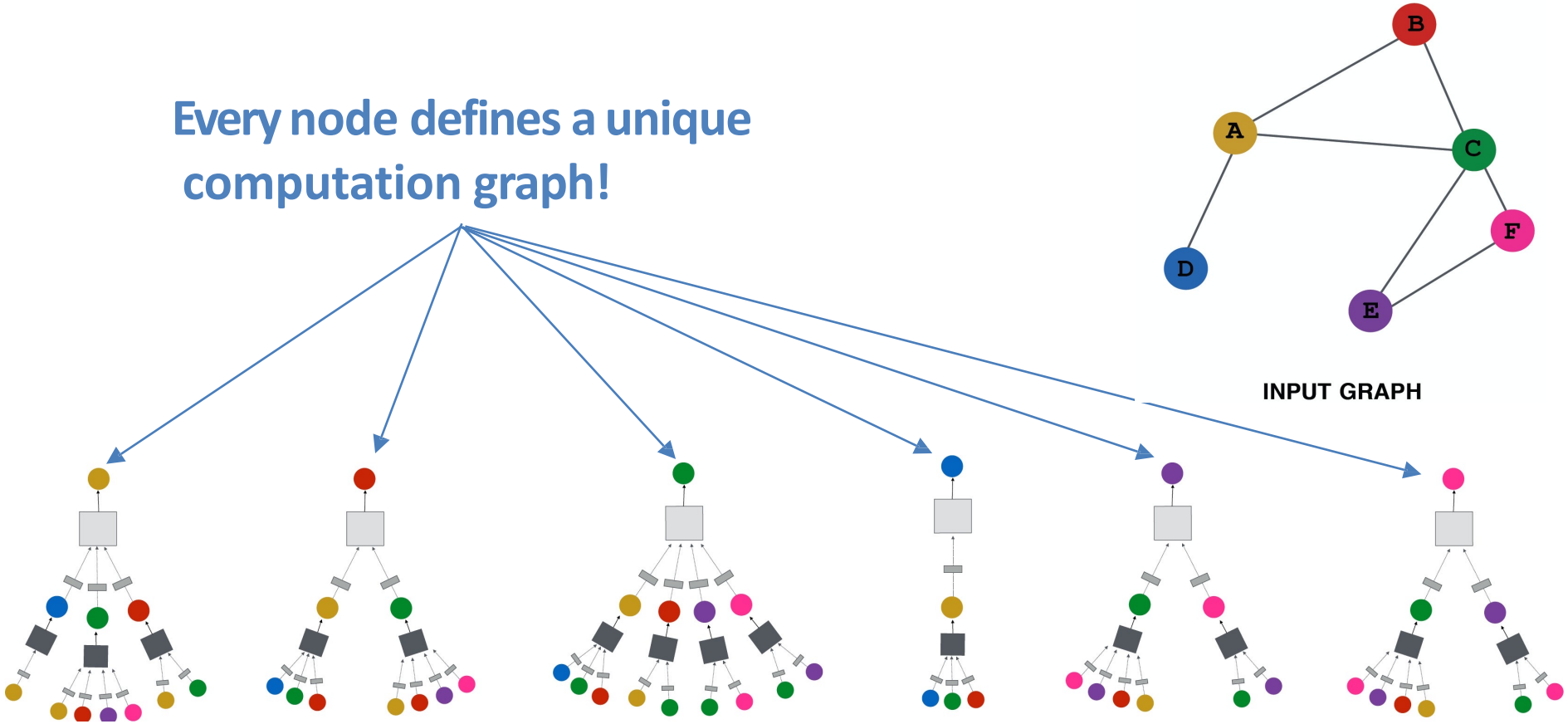
- Key idea: Generate node embeddings based on local neighborhoods.

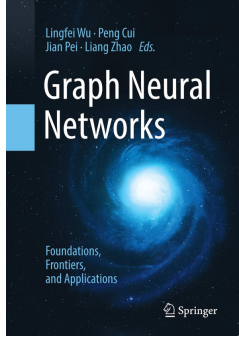





# Neighborhood Aggregation

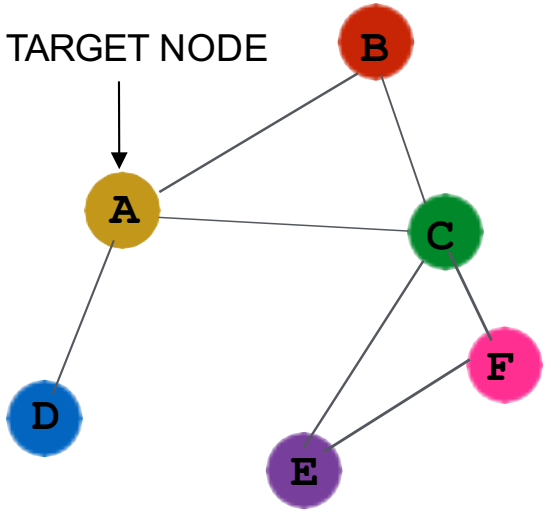
- Intuition: Network neighborhood defines a computation graph

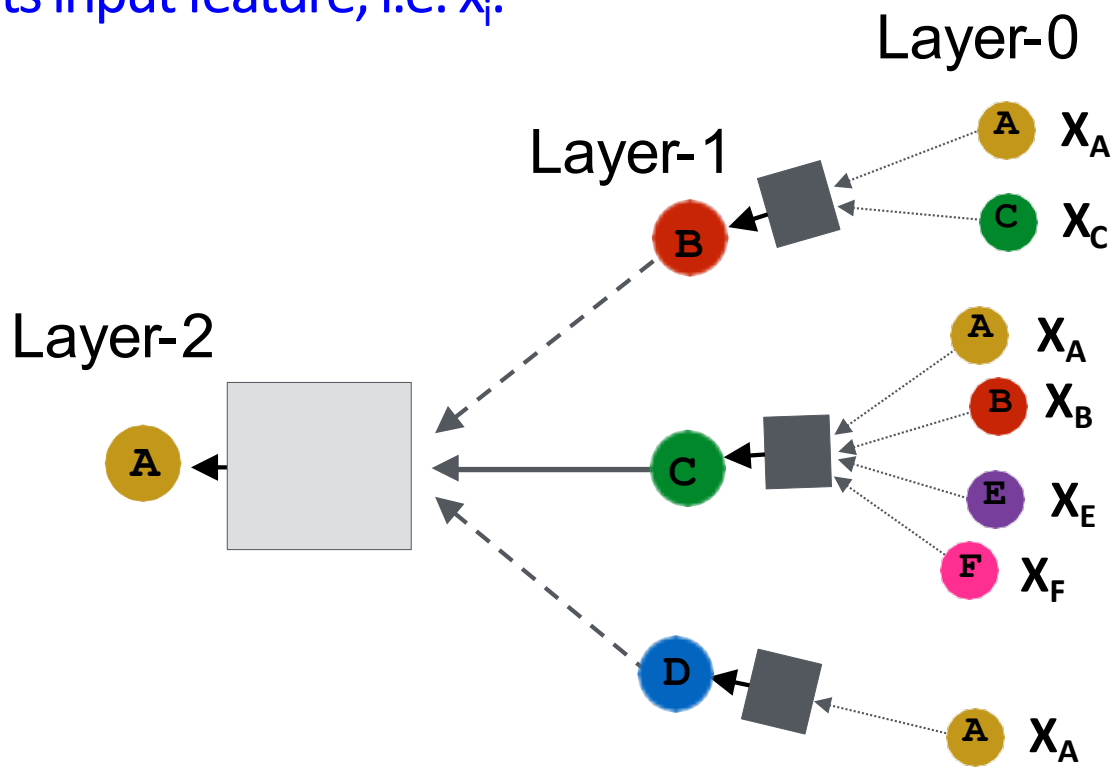




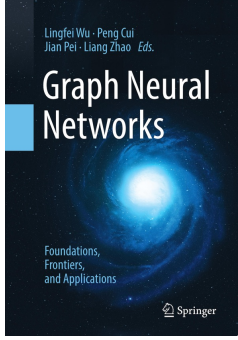

# Neighborhood Aggregation

- Nodes have embeddings at each layer.
- Model can be arbitrary depth.
- "layer-0" embedding of node i is its input feature, i.e. $x_i$.

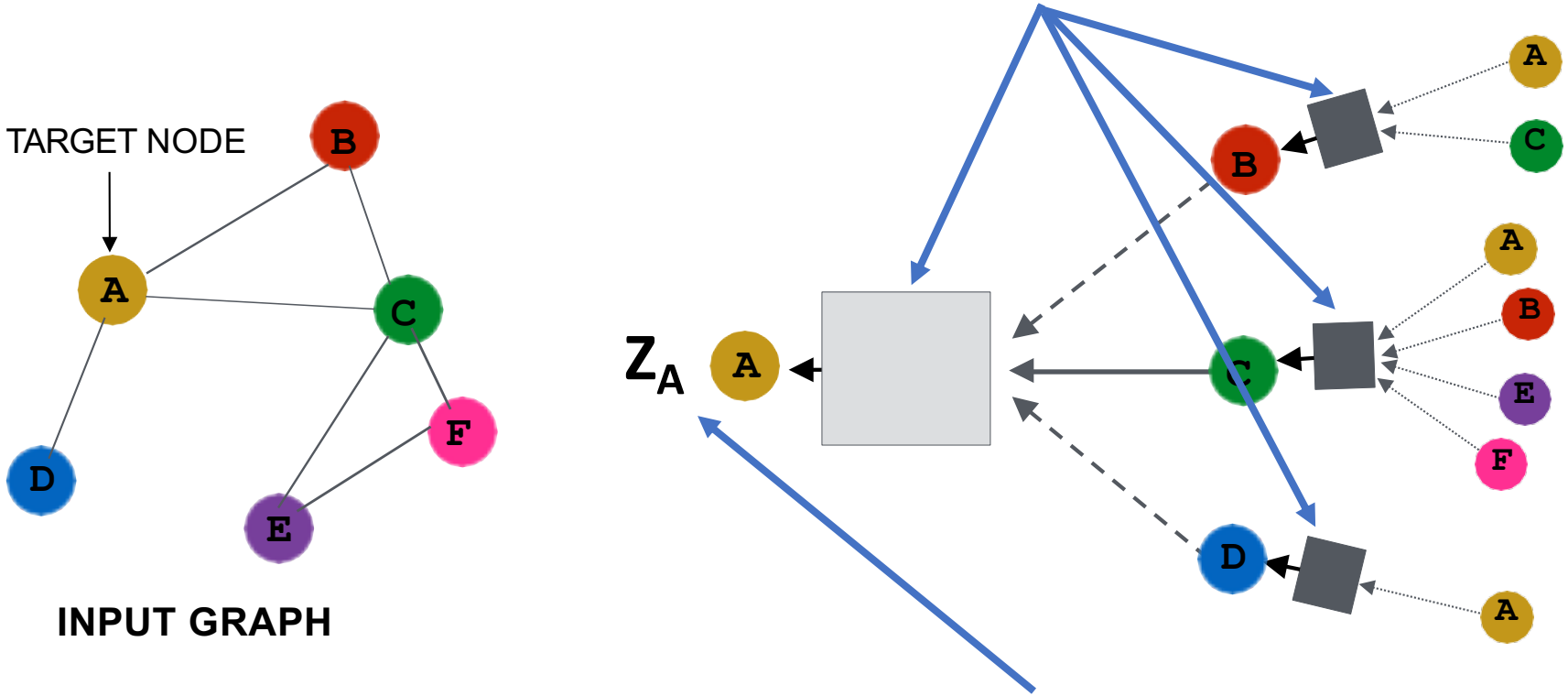


# Overview of GNN Model

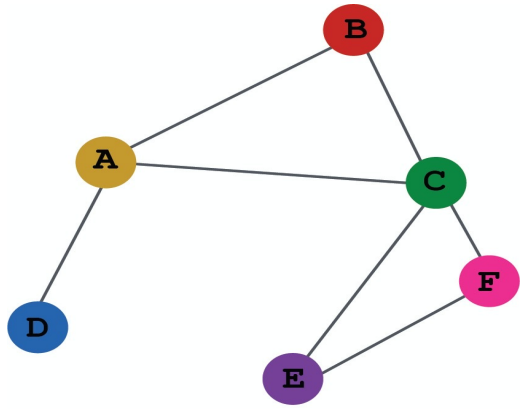



1) Define a neighborhood aggregation function



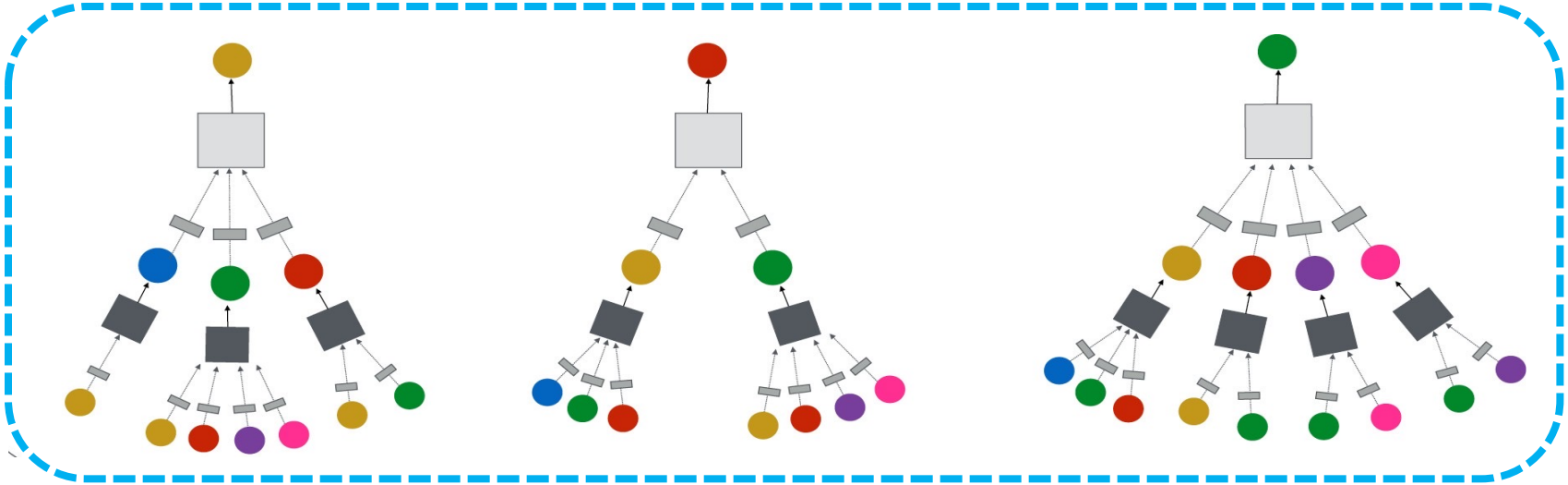

2) Define a loss function on the embeddings, $\mathbf{L(z_v)}$

# Overview of GNN Model

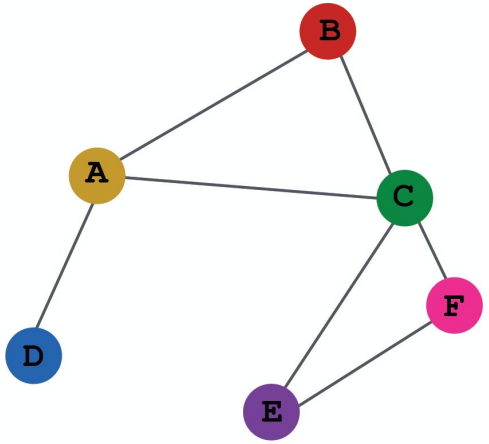



3) Train on a set of nodes, i.e., a batch of computation graphs

INPUT GRAPH



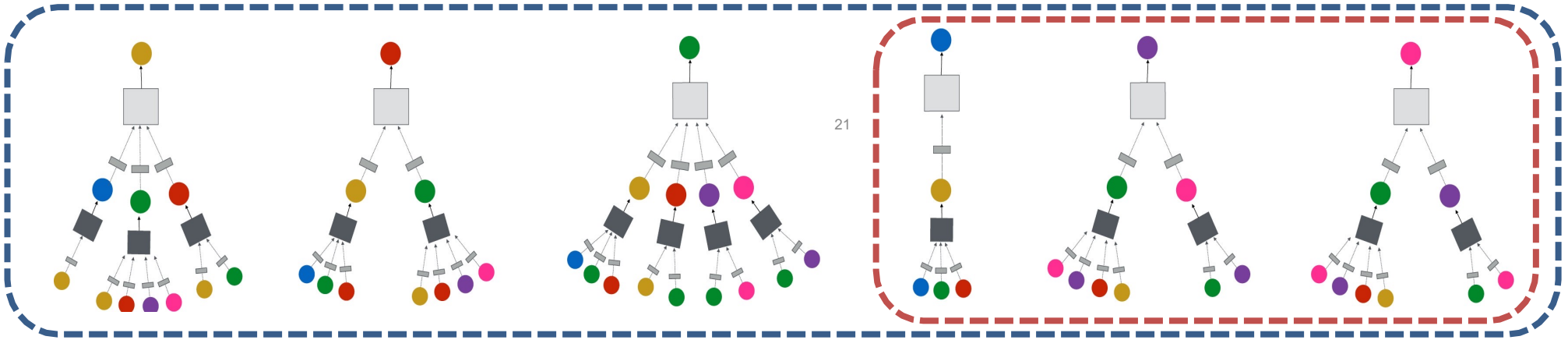
# Overview of GNN Model

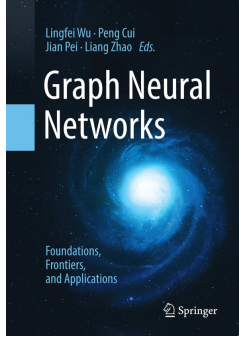



INPUT GRAPH

4) Generate embeddings for nodes as needed

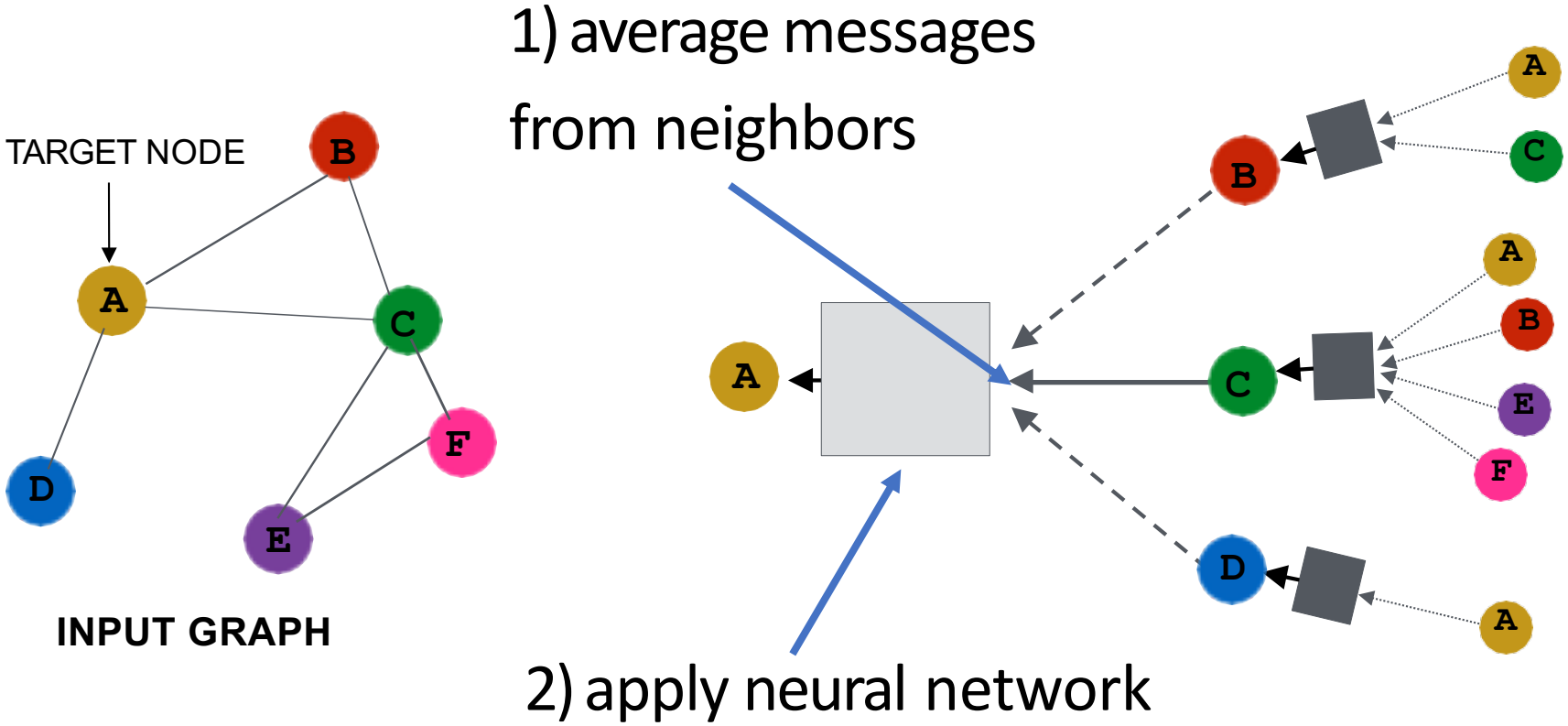
Even for nodes we never trained on!

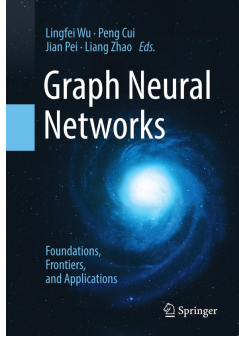





# GNN Model: A Case Study

- Basic approach: Average neighbor information and apply a neural network






# GNN Model: A Case Study

- Basic approach: Average neighbor information and apply a neural network.

Initial "layer 0" embeddings are equal to node features

previous layer embedding of $v$

$$\mathbf{h}_v^0 = \mathbf{x}_v$$

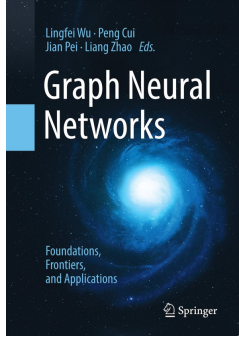
$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right), \quad \forall k > 0$$

kth layer embedding of $v$

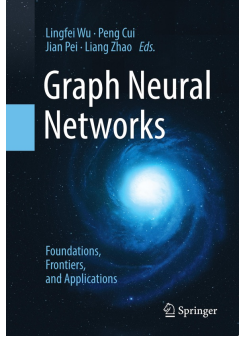
non-linearity (e.g., ReLU or tanh)

average of neighbor's previous layer embeddings

# GNN Model: Quick Summary

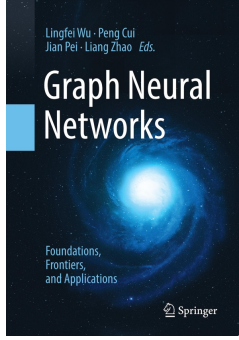



- Key idea: generate node embeddings by aggregating neighborhood information.
  - Allows for parameter sharing in the encoder
  - Allows for inductive learning




# Graph Neural Networks: Popular Models

- Supervised Graph Neural Networks
  - Spectral-based Graph Filters
    - GCN (Kipf & Welling, ICLR 2017), Chebyshev-GNN (Defferrard et al. NIPS 2016)
  - Spatial-based Graph Filters
    - MPNN (Gilmer et al. ICML 2017), GraphSage (Hamilton et al. NIPS 2017)
    - GIN (Xu et al. ICLR 2019)
  - Attention-based Graph Filters
    - GAT (Velickovic et al. ICLR 2018)
  - Recurrent-based Graph Filters
    - GGNN (Li et al. ICLR 2016)
- Unsupervised Graph Neural Networks
  - Variational graph auto-encoders (Kipf and Welling, 2016)
  - Deep graph infomax (Velickovic et al, 2019)




# Graph Convolution Networks (GCN)

Key idea: spectral convolution on graphs

Eigen-decomposition is expensive

$$f_{\text{filter}} * \mathbf{x}_i = \mathbf{U} f(\Lambda) \mathbf{U}^T \mathbf{x}_i$$

Chebyshev polynomials accelerates but still not powerful

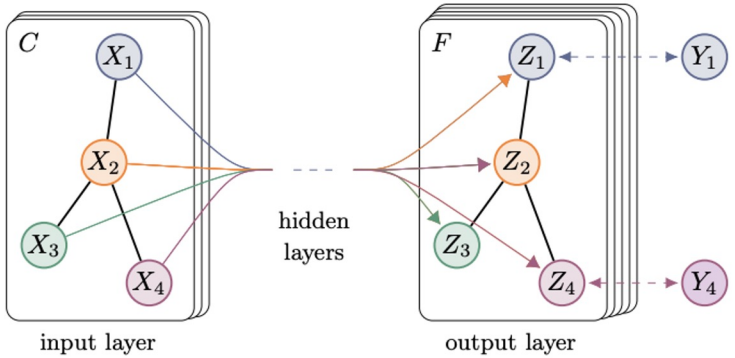
$$f'_{\text{filter}} * \mathbf{x}_i \approx \sum_{p=0}^{P} \theta'_p \mathbf{T}_p(\tilde{\mathbf{L}}) \mathbf{x}_i$$

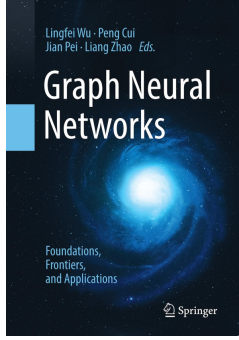
First-order approxima-tion fast and powerful

$$f_{\text{filter}} * \mathbf{h}_i^{(l)} \approx \theta(I_n + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) \mathbf{h}_i^{(l)}$$

Renormalization trick stabilizes the numerical computation

$$\mathbf{H}^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} \mathbf{H}^{(l)} \mathbf{W}^{(l)})$$



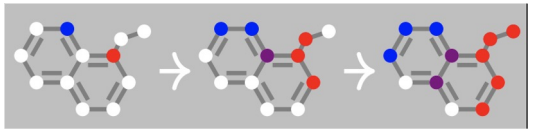



# Message Passing Neural Network (MPNN)

Key idea: graph convolutions as a message passing process

MPNN:

$$\mathbf{h}_i^{(l)} = f_{\text{filter}}(A, \mathbf{H}^{(l-1)}) = f_U(\mathbf{h}_i^{(l-1)}, \sum_{v_j \in N(v_i)} f_M(\mathbf{h}_i^{(l-1)}, \mathbf{h}_j^{(l-1)}, \mathbf{e}_{i,j}))$$



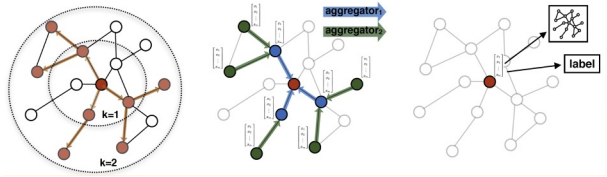
expensive if the number of nodes are large

Update and aggregation functions

Node and edge embeddings

GraphSage:

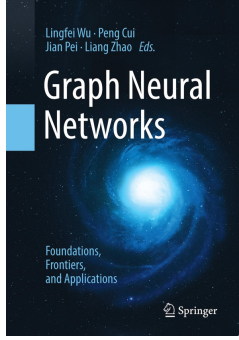
$$f_{\text{filter}}(A, \mathbf{H}^{(l-1)}) = \sigma(\mathbf{W}^{(l)} \cdot f_M(\mathbf{h}_i^{(l-1)}, \{\mathbf{h}_j^{(l-1)}, \forall v_j \in N(v_i)\}))$$




sampling to obtain a fixed number of neighbors

Aggregation functions

Node embeddings

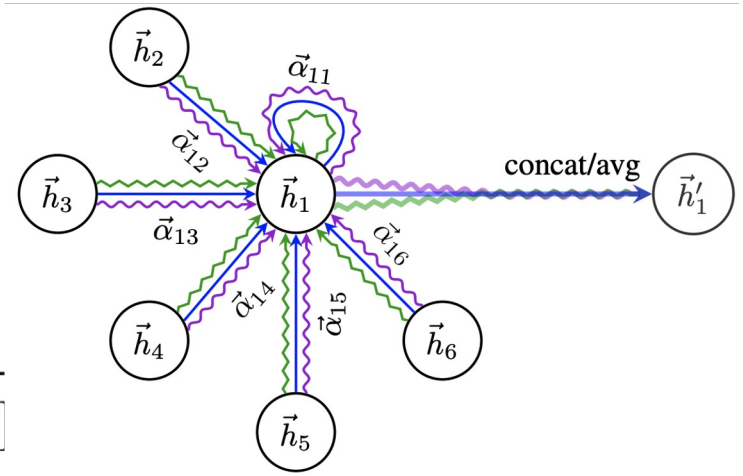



# Graph Attention Network (GAT)

Key idea: dynamically learn the weights (attention scores) on the edges when performing message passing

Weighted sum of node embeddings

$$\mathbf{h}_i^{(l)} = f_{\text{filter}}(A, \mathbf{H}^{(l-1)}) = \sigma\Big(\sum_{v_j \in N(v_i)} \alpha_{ij} \mathbf{W}^{(l)} \mathbf{h}_j^{(l-1)}\Big)$$




Learned local weights with self-attention

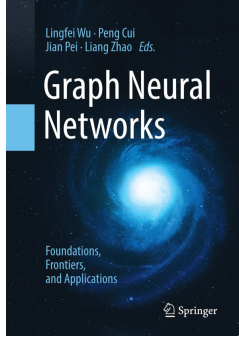
$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(\mathbf{u}^{(l)T}[\mathbf{W}^{(l)}\mathbf{h}_i^{(l-1)} || \mathbf{W}^{(l)}\mathbf{h}_j^{(l-1)}]))}{\sum_{v_k \in N(v_i)} \exp(\text{LeakyReLU}(\mathbf{u}^{(l)T}[\mathbf{W}^{(l)}\mathbf{h}_i^{(l-1)} || \mathbf{W}^{(l)}\mathbf{h}_k^{(l-1)}]))}$$

Intermediate node embeddings

$$f_{\text{filter}}(A, \mathbf{H}^{(l-1)}) = ||_{k=1}^{K} \sigma\Big(\sum_{v_j \in N(v_i)} \alpha_{ij}^k \mathbf{W}_k^{(l)} \mathbf{h}_j^{(l-1)}\Big)$$

Final node embeddings

$$f_{\text{filter}}(A, \mathbf{H}^{(L-1)}) = \sigma\Big(\frac{1}{K}\sum_{k=1}^{K}\sum_{v_j \in N(v_i)} \alpha_{ij}^k \mathbf{W}_k^{(L)} \mathbf{h}_j^{(L-1)}\Big)$$




# Gated Graph Neural Networks (GGNN)

Key idea: the use of Gated Recurrent Units while taking into account edge type and directions

Zero-padding input node embeddings

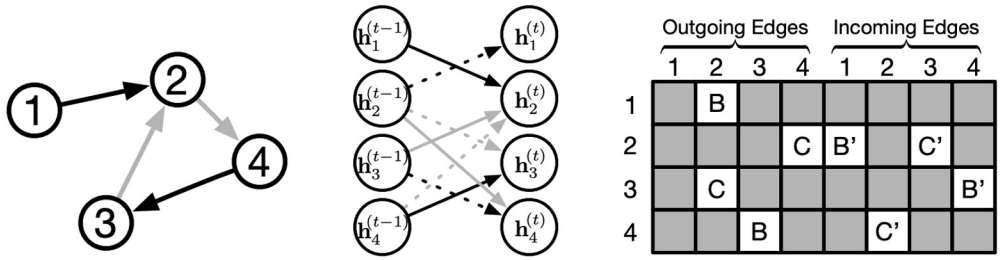
Incoming & outcoming edges for node $v_i$

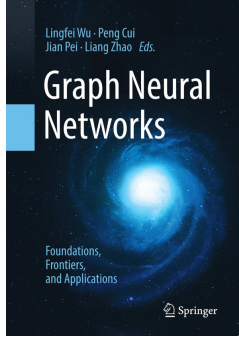
$$\mathbf{h}_i^{(0)} = [\mathbf{x}_i^T, \mathbf{0}]^T$$

$$\mathbf{a}_i^{(l)} = A_{i:}^T [\mathbf{h}_1^{(l-1)} \ldots \mathbf{h}_n^{(l-1)}]^T$$

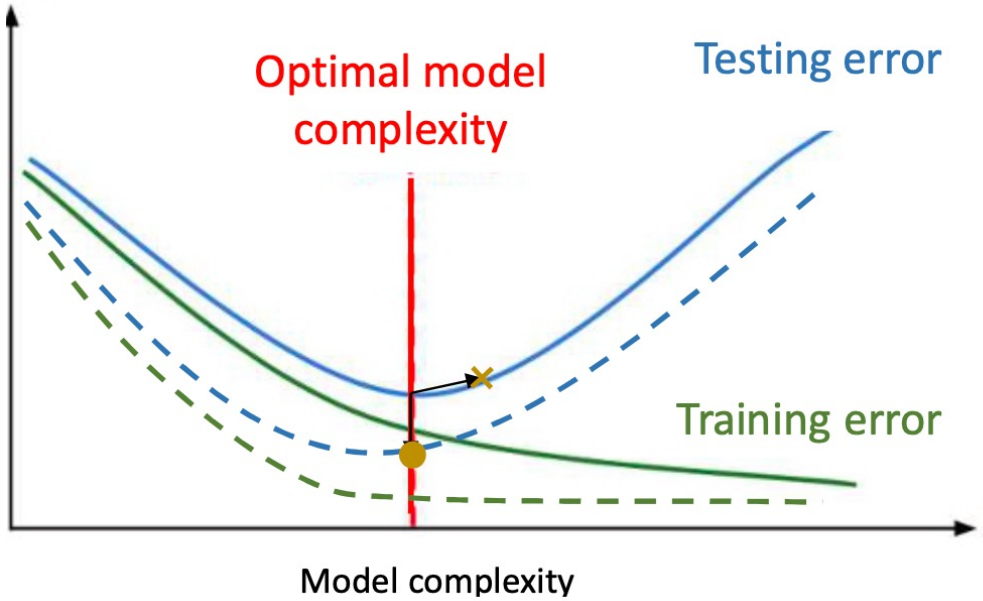
$$\mathbf{h}_i^{(l)} = \mathrm{GRU}(\mathbf{a}_i^{(l)}, \mathbf{h}_i^{(l-1)})$$

GRU for fusing node embeddings





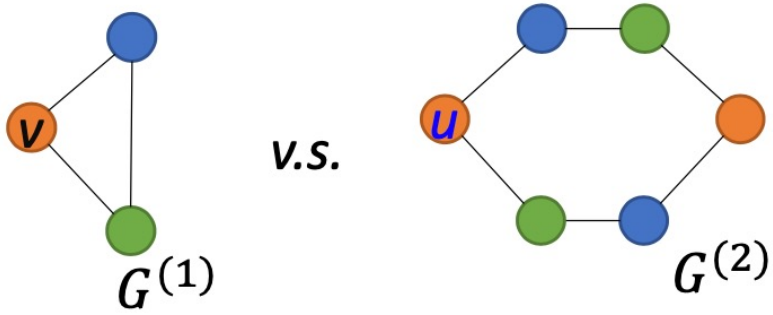

# Expressive Power of GNNs (Chapter 5)



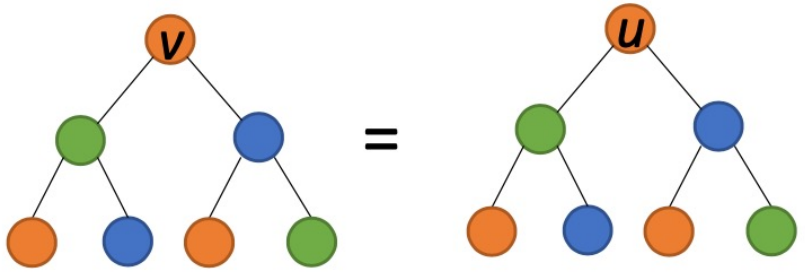

- × Naively improving the expressive power by increasing model complexity
- ● Improving the expressive power by injecting inductive bias into the model while keeping model complexity

—— Without inductive bias

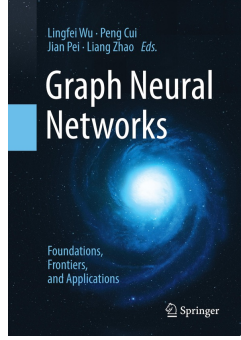
– – With inductive bias

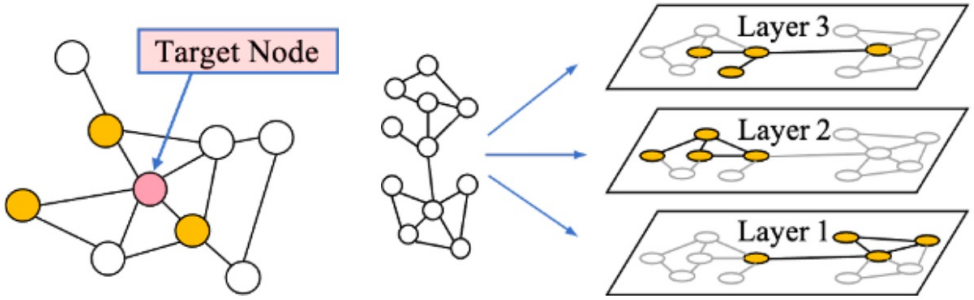


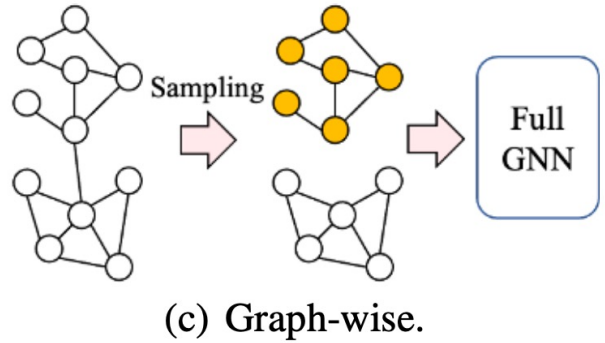

Corresponding subtrees:

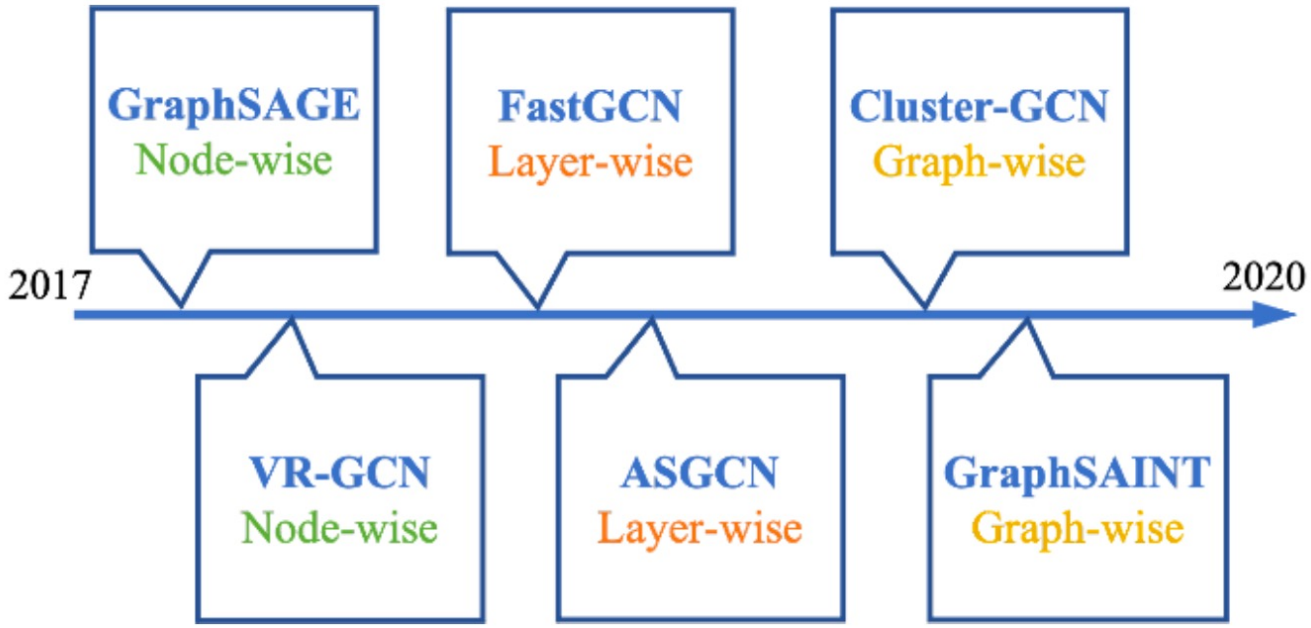


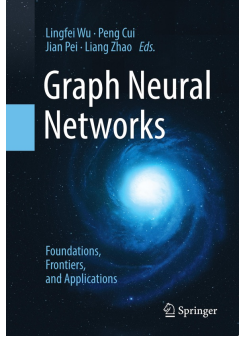
# GNNs: Scalability (Chapter 6)



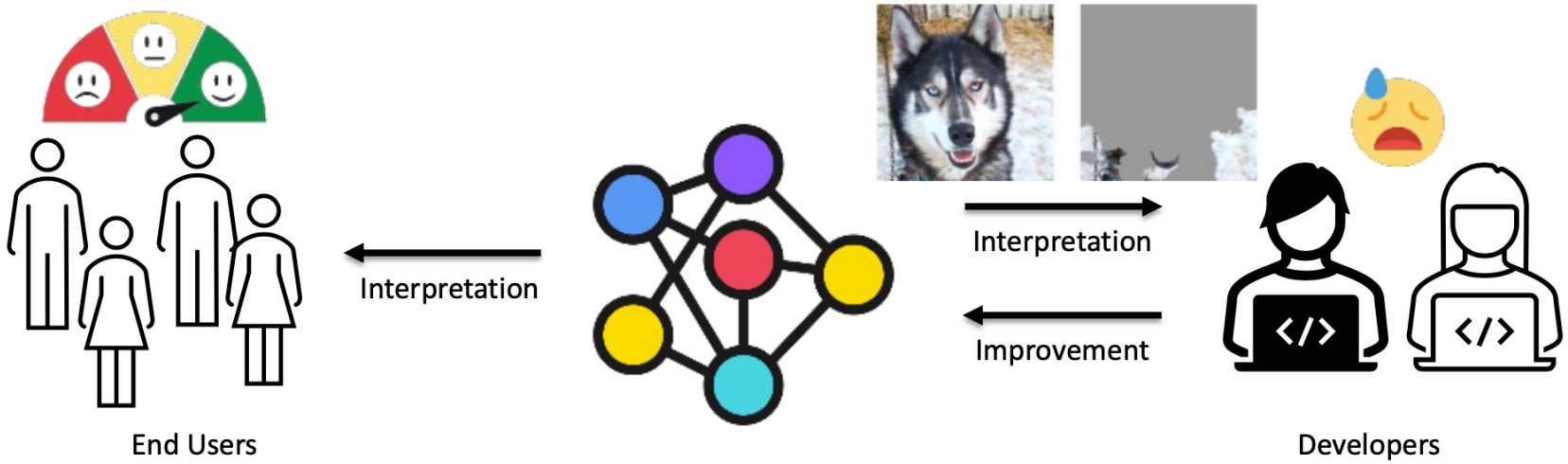



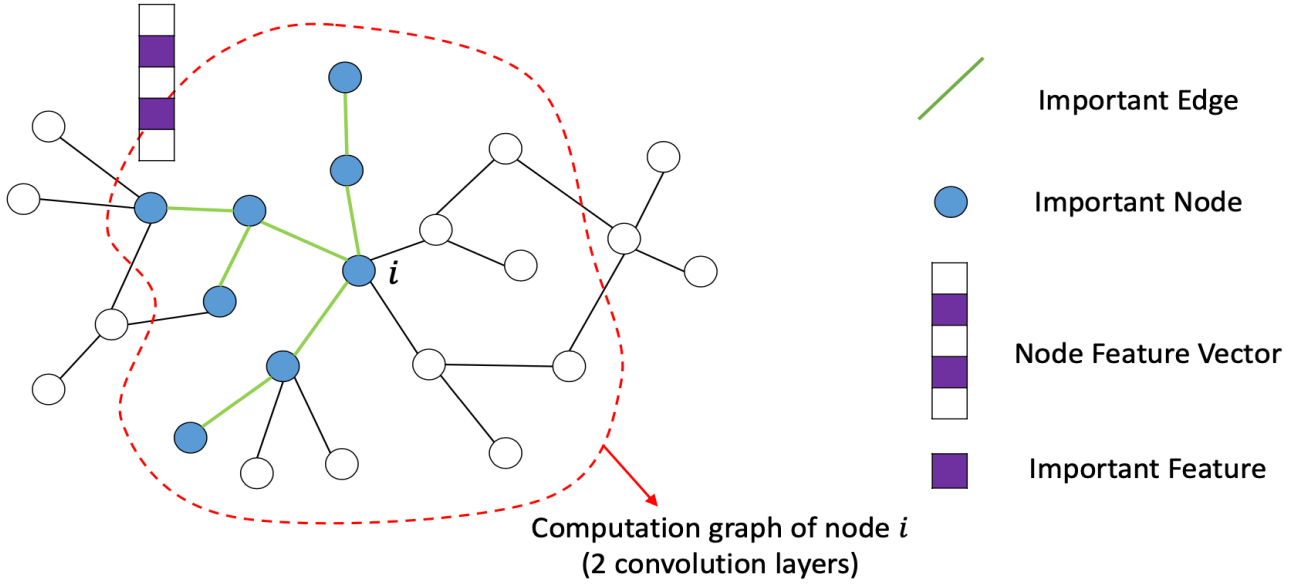



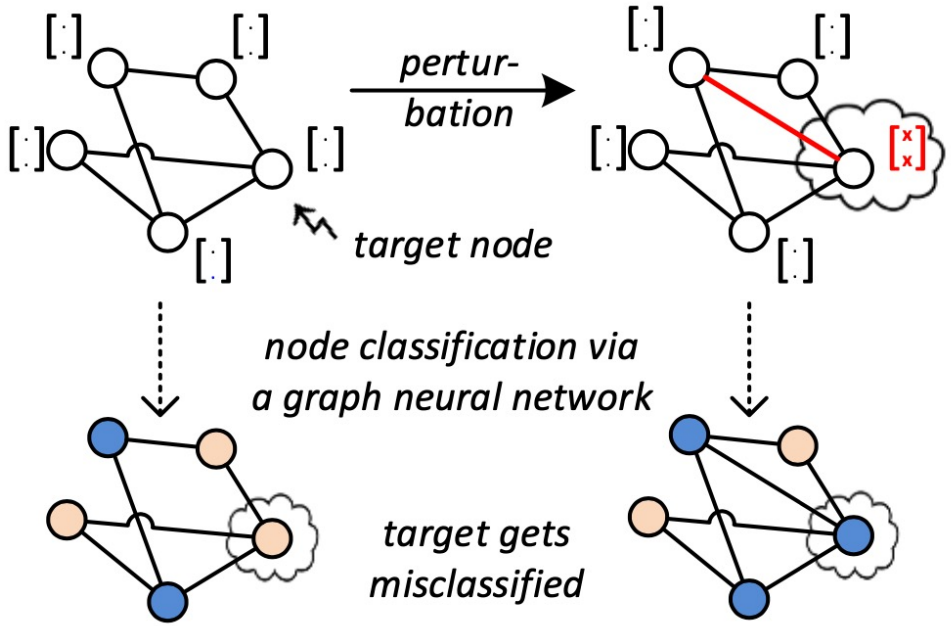


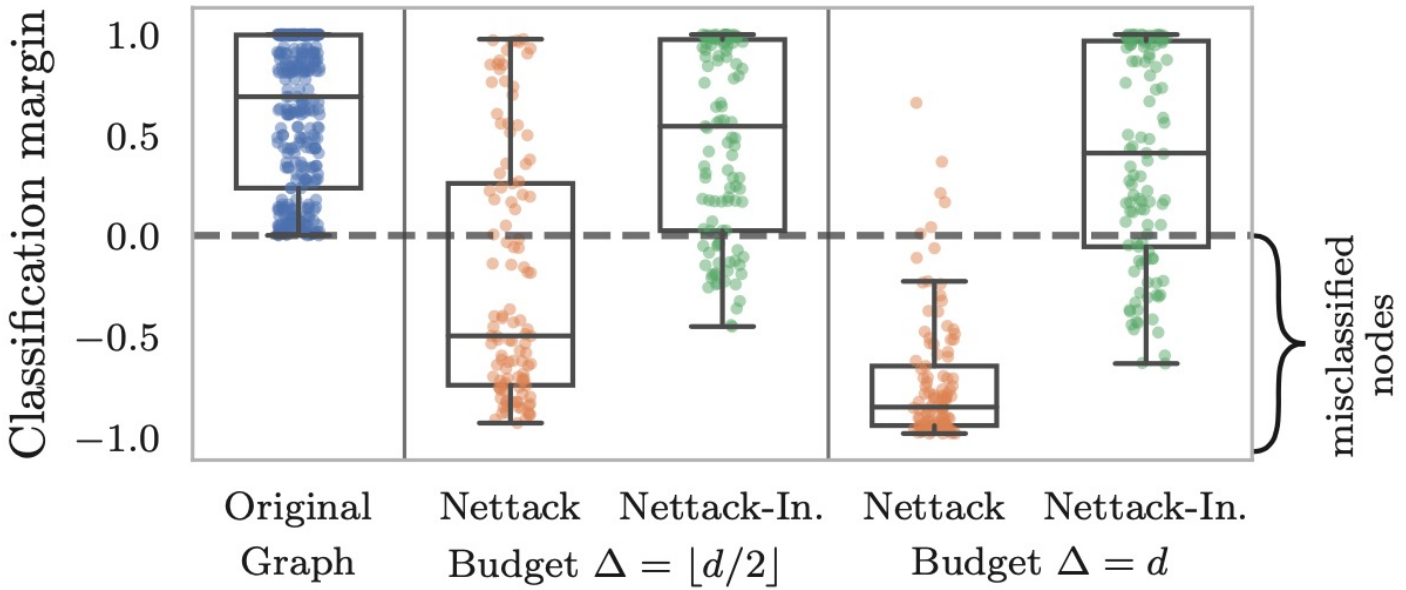

# Interpretability in GNNs (Chapter 7)



# GNNs: Adversarial Robustness (Chapter 8)



# Outline

**GNNs Foundations**

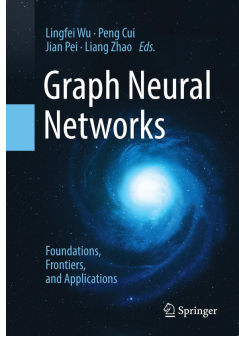
- Time History of GNNs
- GNNs: Foundations and Models
- GNNs: Theory, Scalability, Interpretability

**GNNs Frontiers**

- Graph Structure Learning
- Other Frontiers in GNNs

**GNNs Applications**

- GNNs in Recommendation System
- Other GNNs-based Applications

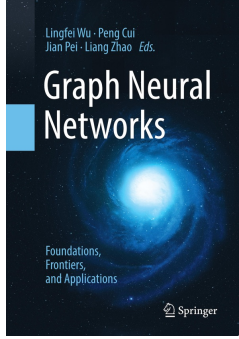



**GNN book website：**
https://graph-neural-networks.github.io/index.html

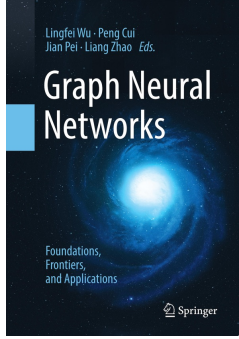
**GNN Springer：**
https://link.springer.com/book/10.1007/978-981-16-6054-2

**Amazon：**
https://www.amazon.com/Graph-Neural-Networks-Foundations-Applications/dp/9811660530

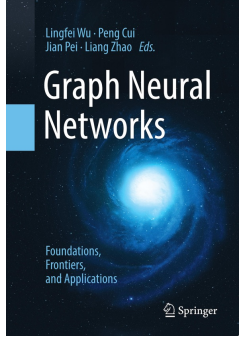
**JD.com (京东商城)：**
https://item.jd.com/10043589466641.html




# GNNs
# Frontiers

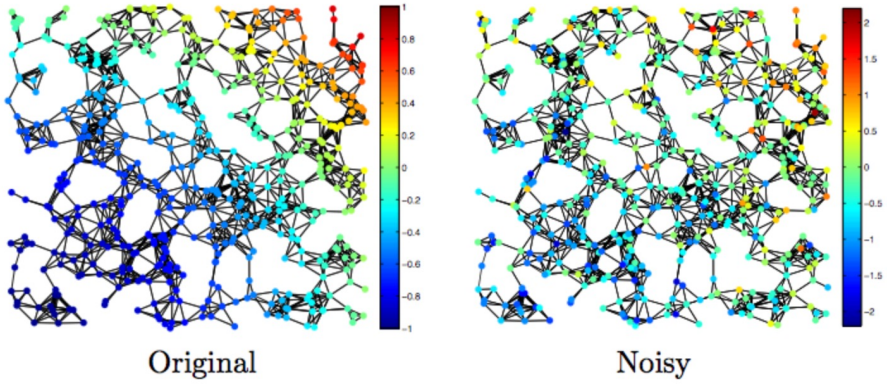


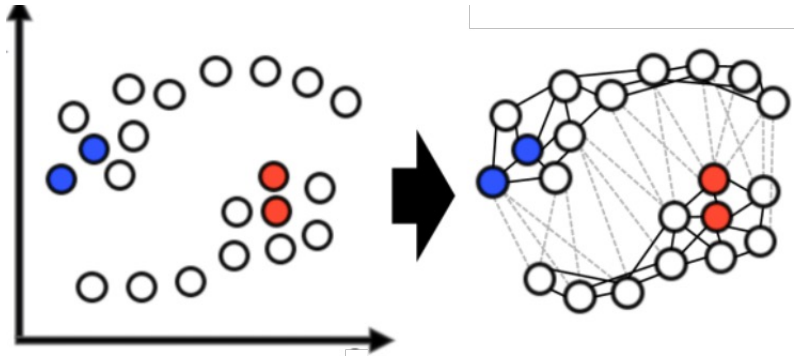

# GNNs: Graph Structure Learning (Chapter 14)

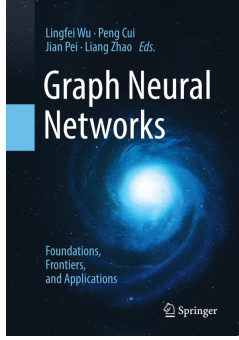



# Graph Learning: Motivations

- GNNs are powerful, unfortunately, it requires graph-structured data available.
- Questionable if the given intrinsic graph-structures are optimal (i.e., noisy, incomplete, etc.) for the downstream tasks.
- Many applications (e.g., NLP tasks) may only have non-graph structured data or even just the original feature matrix, requiring additional graph construction.



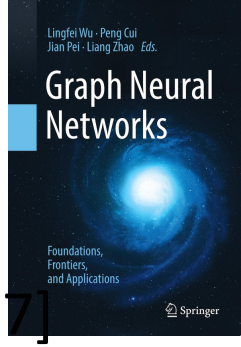





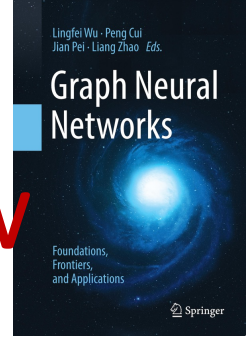

# Graph Learning: Formulation

- Deep graph Learning
  - **Input:** A raw graph input $\mathrm{G} \in \boldsymbol{G}$
    - Given a noisy $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ with $(\mathrm{X}, \mathrm{A}^0)$, where $\mathrm{X} \in \mathrm{R}^{\mathrm{N}\times\mathrm{d}}$, $\mathrm{A}^0 \in \mathrm{R}^{\mathrm{N}\times\mathrm{N}}$
    - Given initial feature matrix X, where $\mathrm{X} \in \mathrm{R}^{\mathrm{N}\times\mathrm{d}}$
  - **Output:** An optimal graph adjacency $\mathrm{A} \in \mathrm{R}^{\mathrm{N}\times\mathrm{N}}$ and node embeddings $\mathrm{Z} \in \mathrm{R}^{\mathrm{N}\times\mathrm{d}'}$

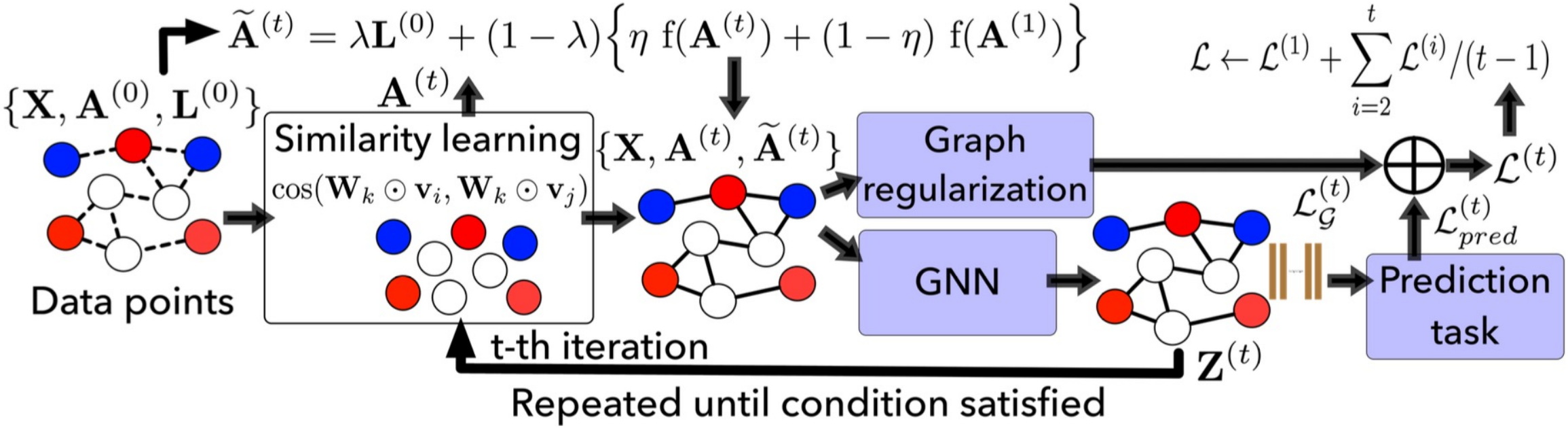
# Existing State-of-the-art Methods




- Graph construction from data [Kalofolias, 2016; Kalofolias and Perraudinl, 2017]
  - Gaussian kernel or KNN-based
  - Directly optimizing the graph adjacency matrix with smoothed graph signals
  - Issues: 1) does not consider downstream task; 2) no refinement
- Dynamic models of interacting systems [Kipf et al., ICML'18]
  - Inferring an explicit interaction structure using a variational graph auto-encoder
  - Issues: 1) cannot joint learn the graph structure and graph representations; 2) transductive
- Jointly optimizing graph structures and GNN parameters [Franceschi, ICML '19]
  - Modeling joint probability distribution on the edges of the graph consisting of N number of vertices
  - Issues: 1) hard to optimize; 2) not scalable; 3) cannot handle inductive learning

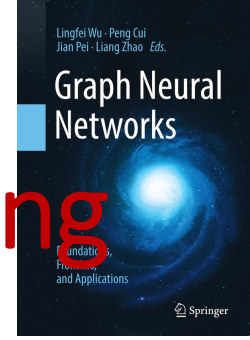



# Iterative Deep Graph Learning : System Overview




- Graph learning as similarity metric learning
- Graph regularization to control smoothness, sparsity, and connectivity
- Iterative method to refine the graph structures and graph embeddings

# IDGL: Graph Learning as Similarity Metric Learning



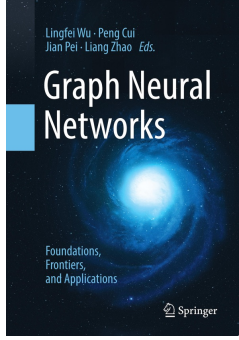

- We design a multi-head weighted cosine similarity metric function to learn a similarity matrix S for all pairs of nodes.

$$s_{ij}^k = \cos(\mathbf{W}_k \odot \mathbf{x}_i, \mathbf{W}_k \odot \mathbf{x}_j) \qquad s_{ij} = \frac{1}{m}\sum_{k=1}^{m} s_{ij}^k$$

- We proceed to extract a symmetric sparse adjacency matrix from the similarity matrix S by considering only the ε-neighborhood for each node.

$$\mathbf{A}_{ij} = \begin{cases} \mathbf{S}_{ij} & \mathbf{S}_{ij} > \varepsilon \\ 0 & \text{otherwise} \end{cases} \qquad \tilde{\mathbf{A}} = \lambda \mathbf{L}_0 + (1-\lambda)\frac{\mathbf{A}_{ij}}{\sum_j \mathbf{A}_{ij}}$$

where $\mathbf{L}_0$ is the normalized adjacency matrix of the initial graph (or kNN-graph).




# IDGL: Graph Regularization

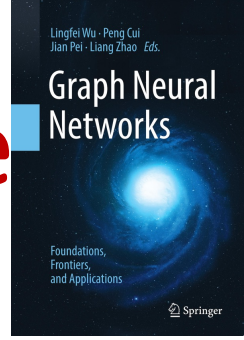
- We adapt the techniques designed for learning graphs from smooth and apply them as regularization for controlling smoothness, connectivity and sparsity

$$\Omega(\mathbf{A}, \mathbf{X}) = \frac{1}{2}\sum_{i,j}\mathbf{A}_{ij}||\mathbf{x}_i - \mathbf{x}_j||^2 = \mathrm{tr}(\mathbf{X}^T\mathbf{L}\mathbf{X})$$ Smoothness

$$f(\mathbf{A}) = -\beta\mathbf{1}^T\log(\mathbf{A}\mathbf{1}) + \gamma||\mathbf{A}||_F^2$$ Connectivity & sparsity

$$\mathcal{L}_{\mathcal{G}} = \alpha\Omega(\mathbf{A}, \mathbf{X}) + f(\mathbf{A})$$ Graph regularization loss

# IDGL: Iterative Method for Joint Graph Structure and Representation Learning




- Iterative method repeatedly
  - refines the adjacency matrix with the updated node embeddings
  - refines the node embeddings with the updated adjacency matrix
- Iterative procedure dynamically stops
  - the learned adjacency matrix converges with certain threshold
  - the maximal number of iterations is reached

**while** $(t == 0 \quad or \quad ||\mathbf{A}^{(t)} - \mathbf{A}^{(t-1)}||_F^2 > \delta||\mathbf{A}^{(0)}||_F^2)$ $c$

$t \leftarrow t + 1$

$\mathbf{A}^{(t)}, \tilde{\mathbf{A}}^{(t)} \leftarrow \{\mathbf{Z}^{(t-1)}, \mathbf{A}_0\}$ using Eqs. (2), (3) and (10)

$\tilde{\mathbf{A}}^{(t)} \leftarrow \eta\tilde{\mathbf{A}}^{(t)} + (1-\eta)\tilde{\mathbf{A}}^{(0)}$

$\mathbf{Z}^{(t)} \leftarrow \{\tilde{\mathbf{A}}^{(t)}, \mathbf{X}\}$ using Eq. (7)

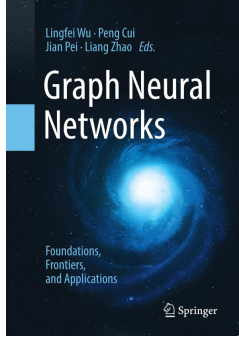
$\hat{\mathbf{y}} \leftarrow \{\tilde{\mathbf{A}}^{(t)}, \mathbf{Z}^{(t)}\}$ using Eq. (8)

$\mathcal{L}_{\text{pred}}^{(t)} \leftarrow \{\hat{\mathbf{y}}, \mathbf{y}\}$ using Eq. (9)

$\mathcal{L}_{\mathcal{G}}^{(t)} \leftarrow \{\mathbf{A}^{(t)}, \mathbf{X}\}$ using Eqs. (4)–(6)

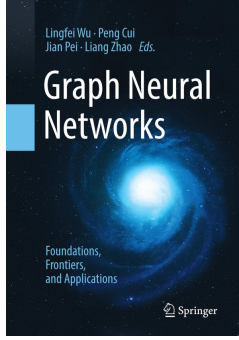
$\mathcal{L}^{(t)} \leftarrow \mathcal{L}_{\text{pred}}^{(t)} + \mathcal{L}_{\mathcal{G}}^{(t)}$

**end**




# Results (Transudative Setting)

| Model | Cora | Citeseer | Pubmed | ogbn-arxiv | Wine | Cancer | Digits |
|---|---|---|---|---|---|---|---|
| GCN | 81.5 | 70.3 | 79.0 | 71.7 (0.3) | — | — | — |
| GAT | 83.0 (0.7) | 72.5 (0.7) | 79.0 (0.3) | — | — | — | — |
| GraphSage | 77.4 (1.0) | 67.0 (1.0) | 76.6 (0.8) | 71.5 (0.3) | — | — | — |
| APPNP | — | **75.7 (0.3)** | 79.7 (0.3) | — | — | — | — |
| H-GCN | **84.5 (0.5)** | 72.8 (0.5) | 79.8 (0.4) | — | — | — | — |
| GCN+GDC | 83.6 (0.2) | 73.4 (0.3) | 78.7 (0.4) | — | — | — | — |
| LDS | 84.1 (0.4) | 75.0 (0.4) | — | — | 97.3 (0.4) | 94.4 (1.9) | 92.5 (0.7) |
| $GCN_{kNN}$* | — | — | — | — | 95.9 (0.9) | 94.7 (1.2) | 89.5 (1.3) |
| LDS* | 83.9 (0.6) | 74.8 (0.3) | — | — | 96.9 (1.4) | 93.4 (2.4) | 90.8 (2.5) |
| IDGL | **84.5 (0.3)** | 74.1 (0.2) | — | — | 97.8 (0.6) | **95.1 (1.0)** | 93.1 (0.5) |
| IDGL-ANCH | 84.4 (0.2) | 72.0 (1.0) | **82.7 (0.4)** | **72.0 (0.3)** | **98.1 (1.1)** | 94.8 (1.4) | **93.2 (0.9)** |

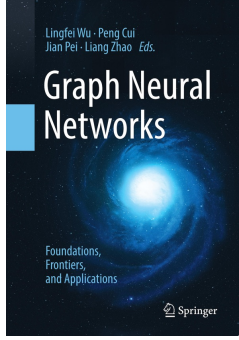


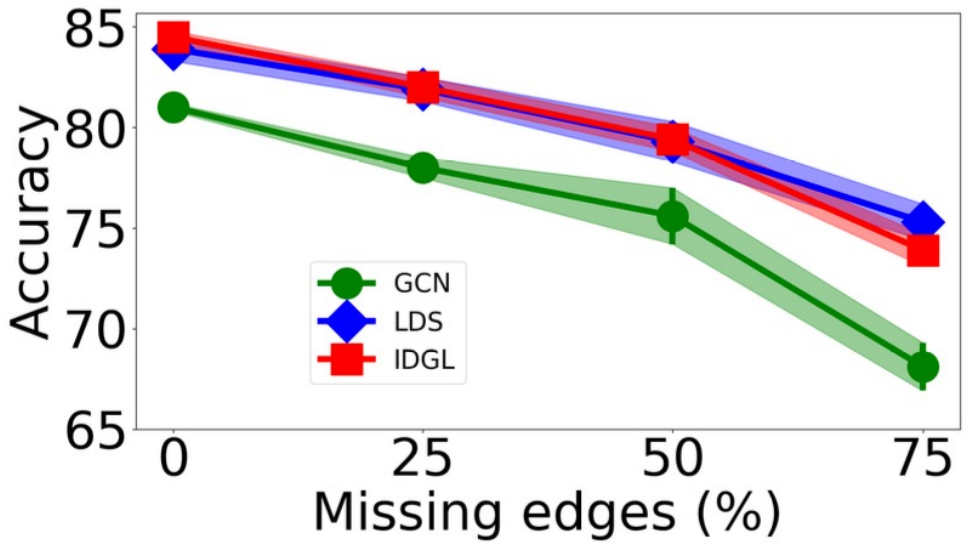

# Results (Inductive Setting & Runtime)

| Methods | 20News | MRD |
|---|---|---|
| BiLSTM | 80.0 (0.4) | 53.1 (1.4) |
| $GCN_{kNN}$ | 81.3 (0.6) | 60.1 (1.5) |
| IDGL | **83.6 (0.4)** | **63.7 (1.8)** |
| IDGL-ANCH | 82.9 (0.3) | 62.9 (0.4) |

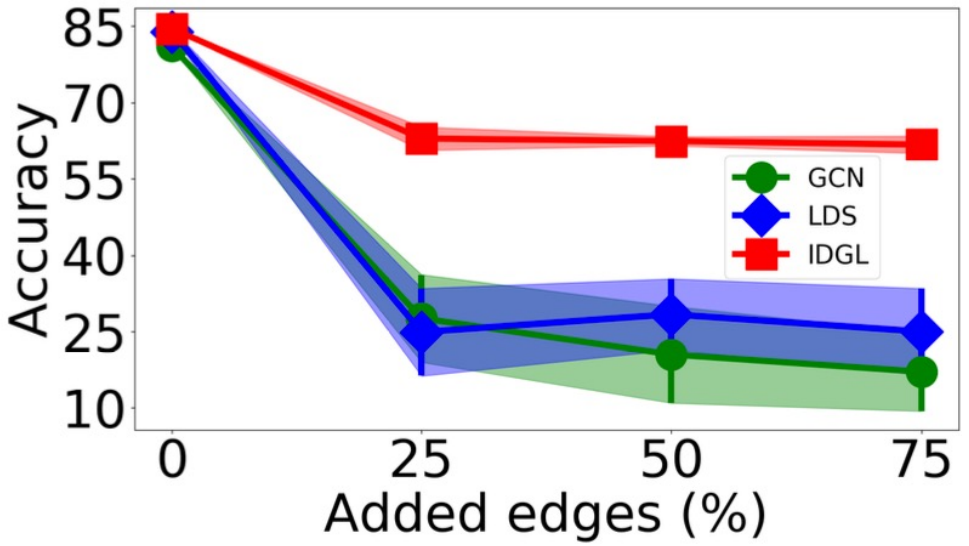
| Data | Cora | Citeseer | Pubmed |
|---|---|---|---|
| GCN | **3 (1)** | **5 (1)** | **28 (3)** |
| GAT | 26 (5) | 28 (5) | — |
| LDS | 390 (82) | 585 (181) | — |
| IDGL | 237 (21) | 563 (100) | — |
| w/o IL | 49 (8) | 61 (15) | — |
| IDGL-ANCH | 83 (6) | 261 (50) | 235 (19) |
| w/o IL | 28 (4) | 69 (9) | 99 (17) |

# Results (Robustness to Missing/Adding Edges)



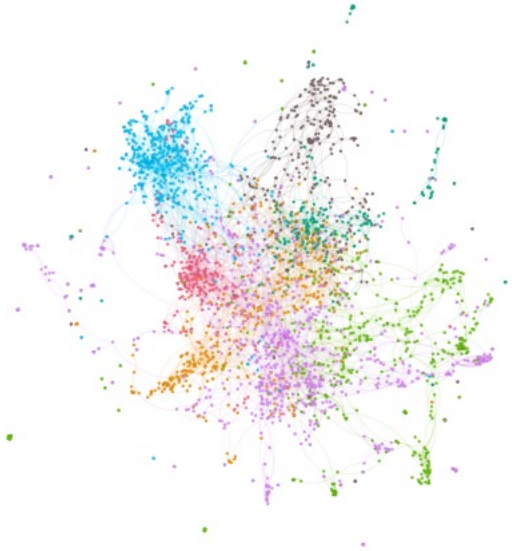




(a) Edge deletion




(b) Edge addition

# Visualization of the Initial and Learned Graphs

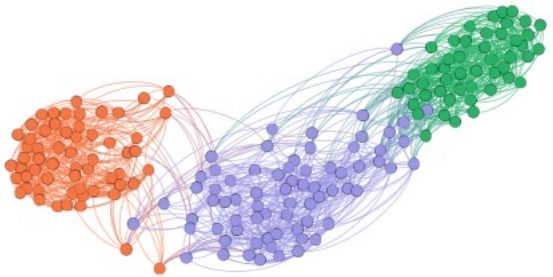


(a) Initial graph ($\mathbf{A}^{(0)}$)

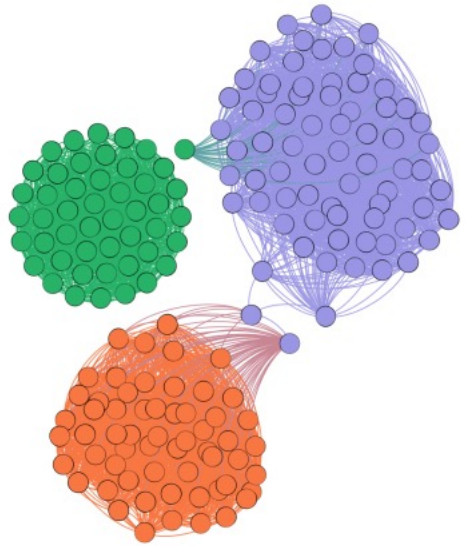
(b) Learned graph ($\mathbf{A}^{(t)}$)

Dataset: Cora (having initial graph data)

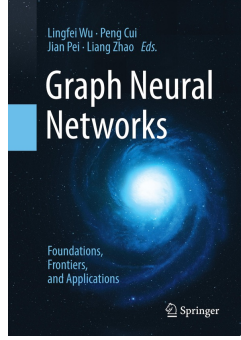
Dataset: Wine (having no initial graph data)



(a) kNN graph ($\mathbf{A}^{(0)}$)



(b) Learned graph ($\mathbf{A}^{(t)}$)




# GNNs: Graph Classification (Chapter 9)

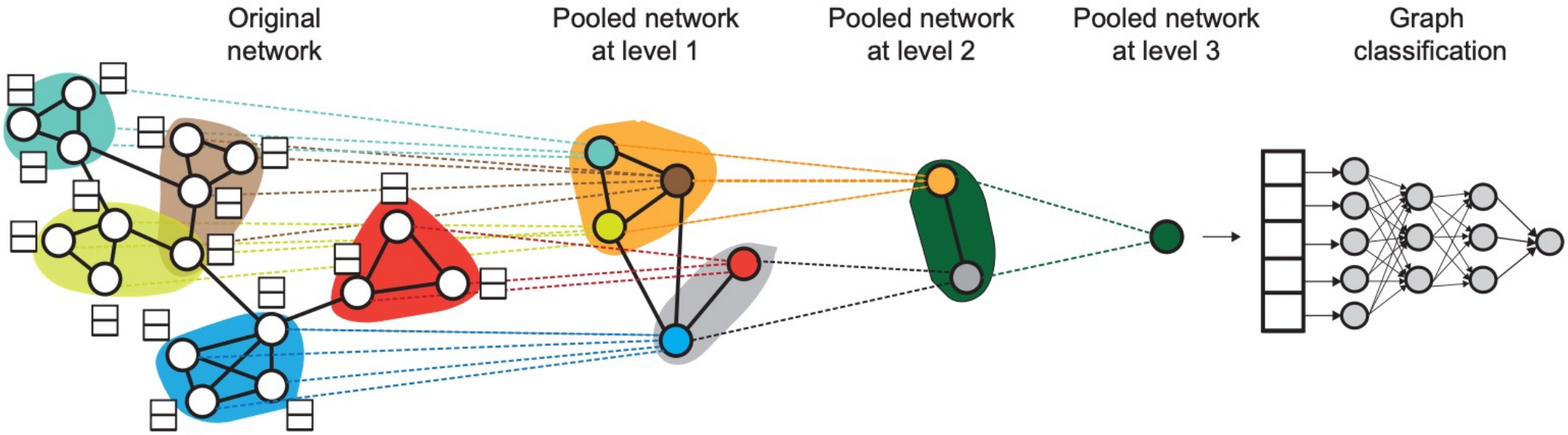
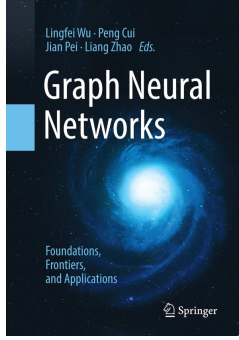
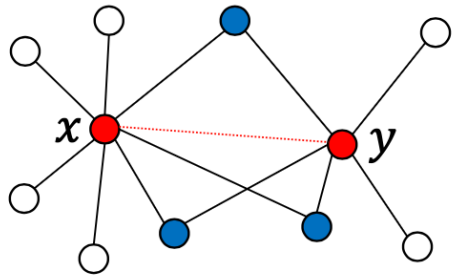


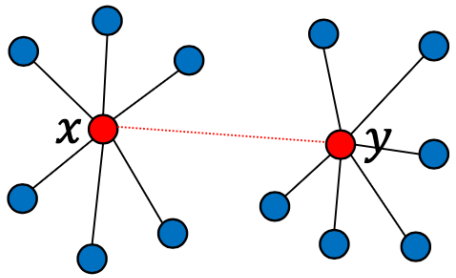

Image credit: Rex Ying et al., “Hierarchical Graph Representation Learning with Differentiable Pooling”, NeurIPS 2018.



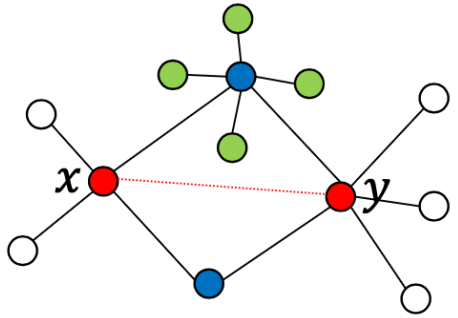

# GNNs: Link Prediction (Chapter 10)



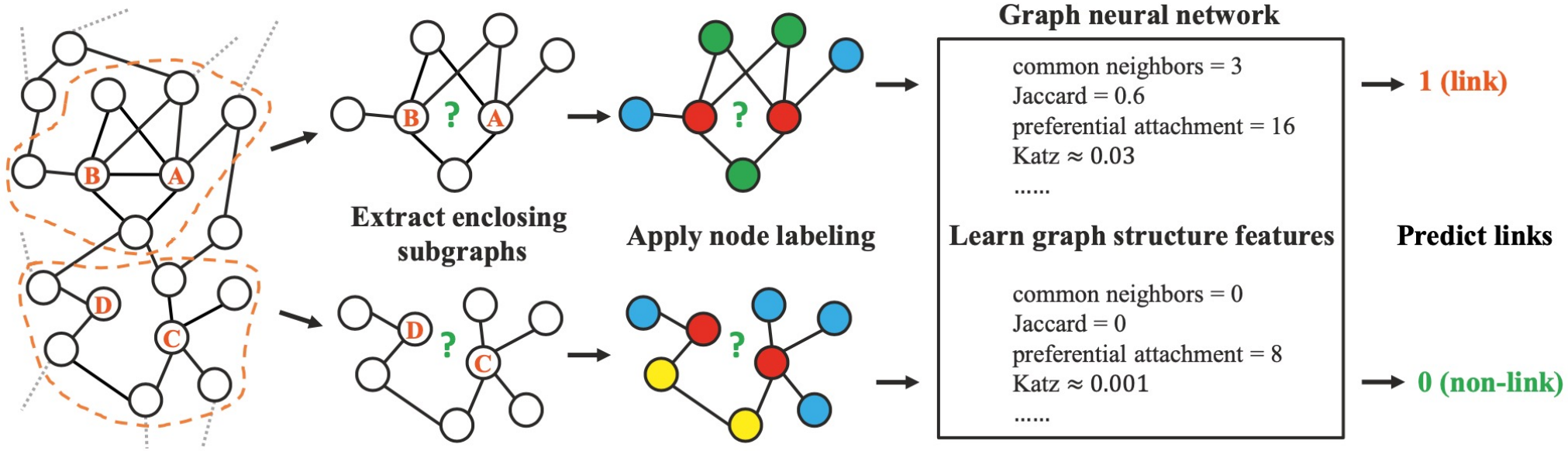

common neighbors (CN):
$|\Gamma(x) \cap \Gamma(y)|$



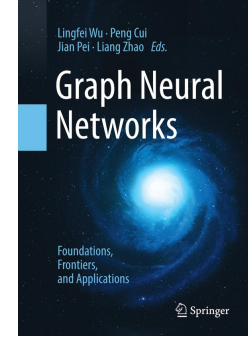

preferential attachment (PA):
$|\Gamma(x)| \cdot |\Gamma(y)|$

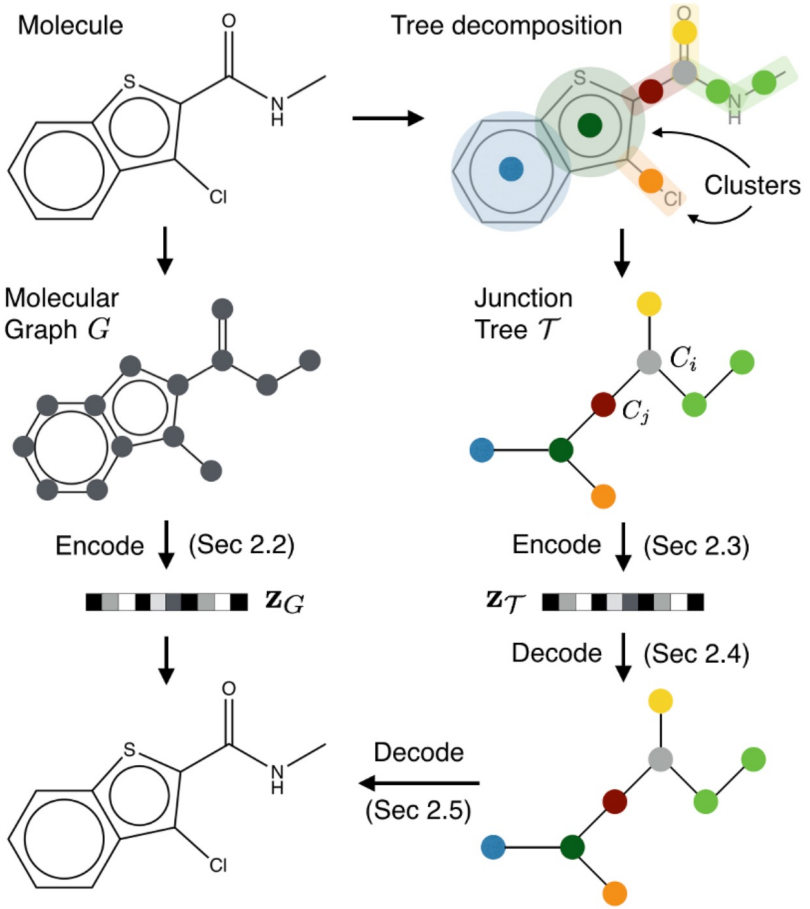


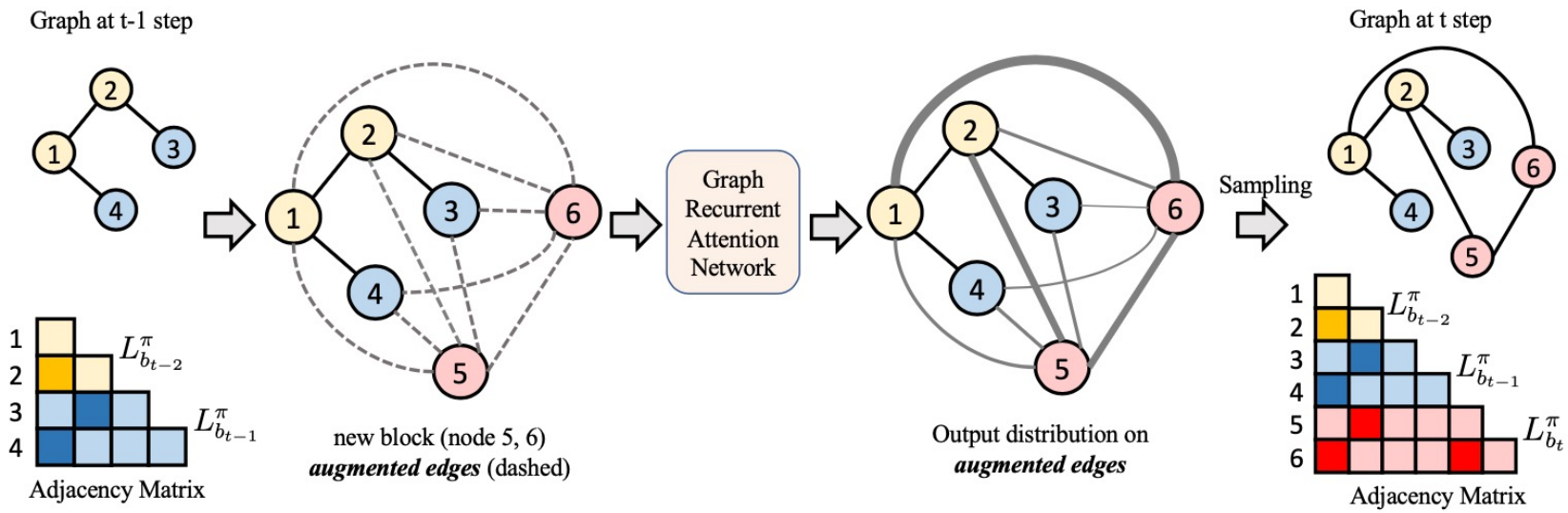

Adamic-Adar (AA):
$\sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log |\Gamma(z)|}$



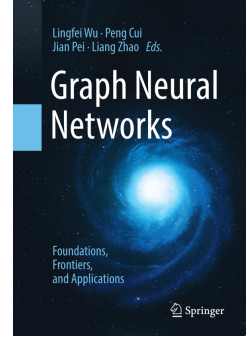
# GNNs: Graph Generation (Chapter 11)



# GNNs: Graph Transformation (Chapter 12)



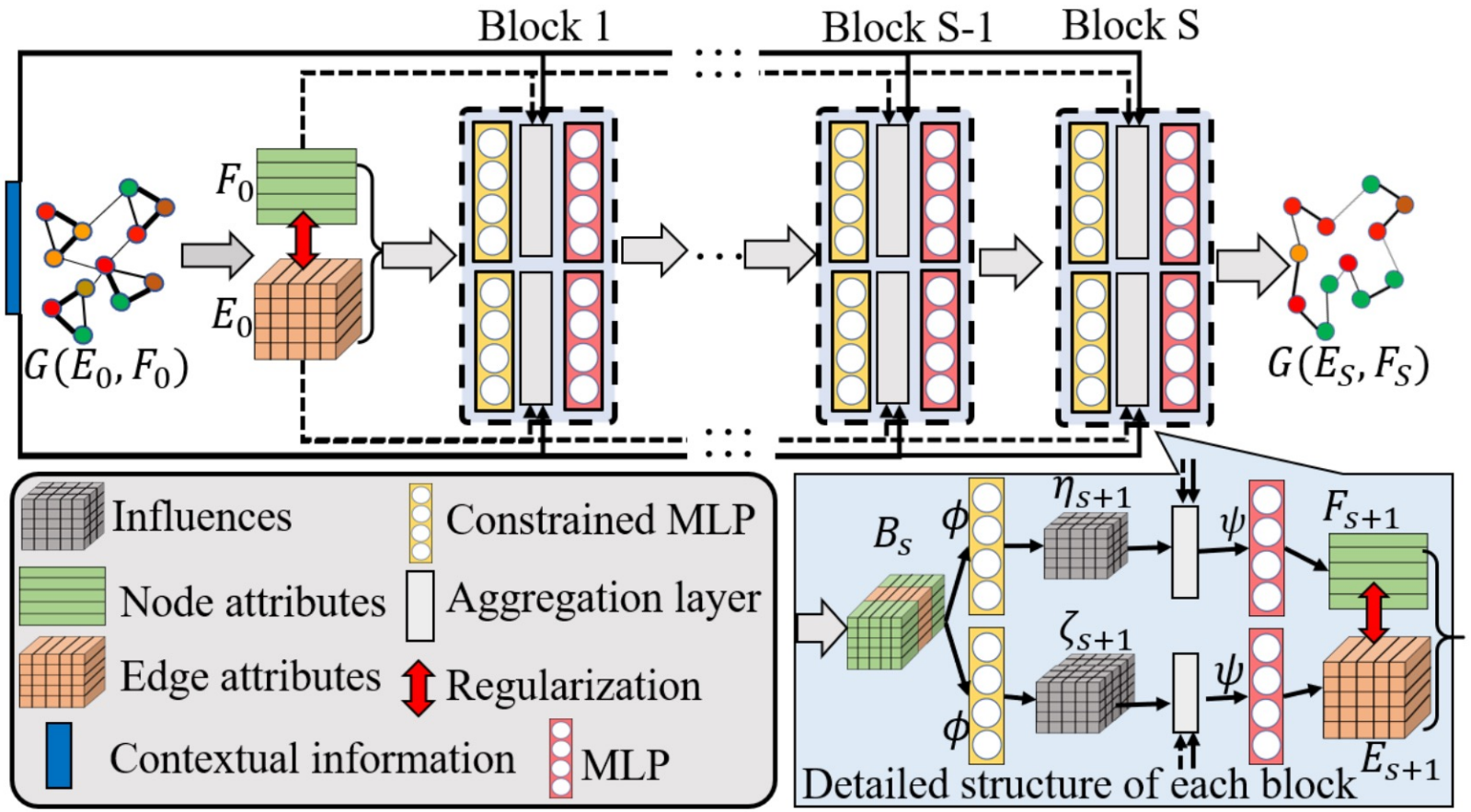




Image credit: Xiaojie Guo et al., “Deep Multi-attributed Graph Translation with Node-Edge Co-evolution”

# GNNs: Graph Matching (Chapter 13)

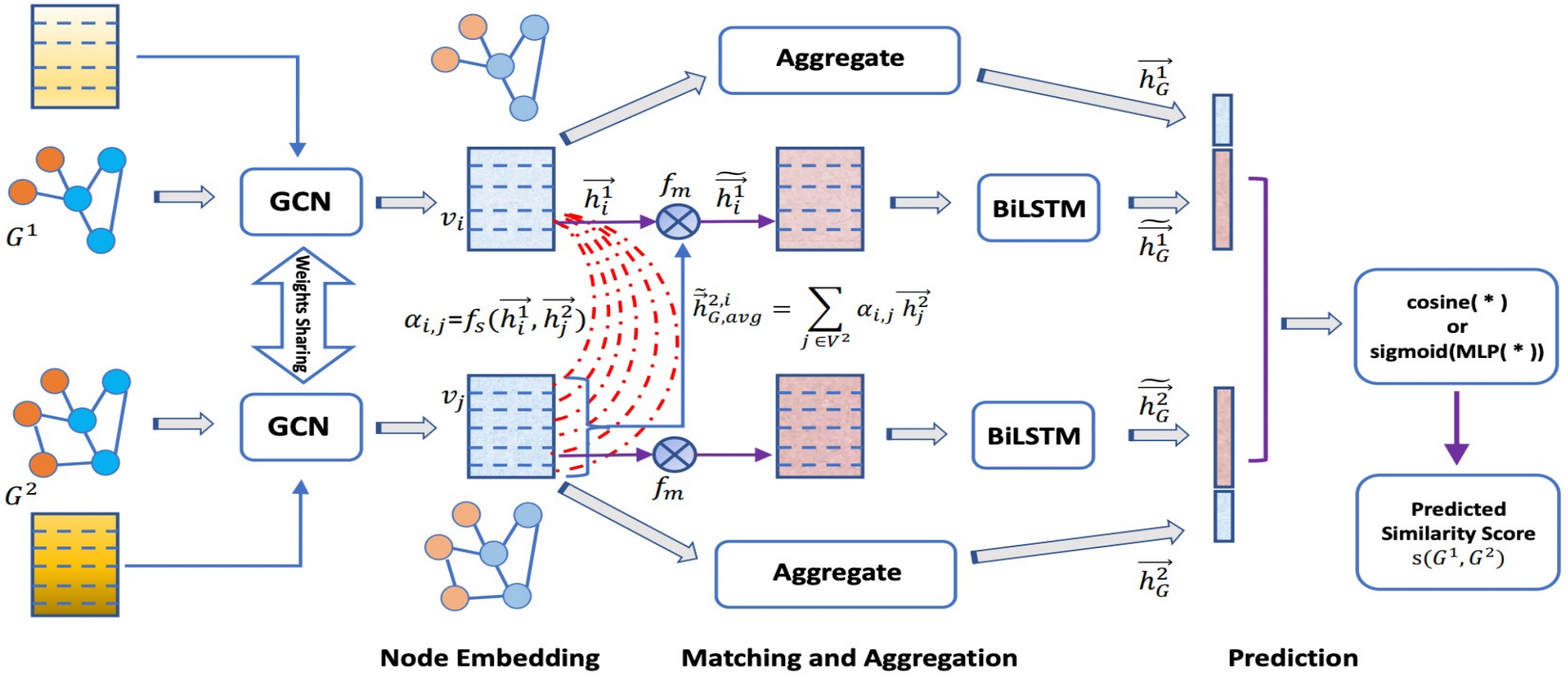
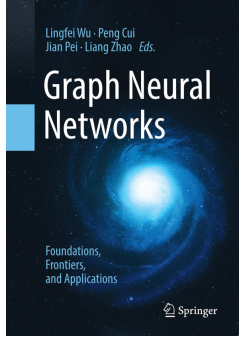
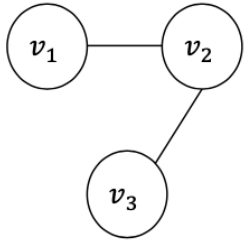



Image credit: Ling and Wu et al. 2022, "Hierarchical Graph Matching Networks for Deep Graph Similarity Learning"




# Dynamic GNNs (Chapter 15)

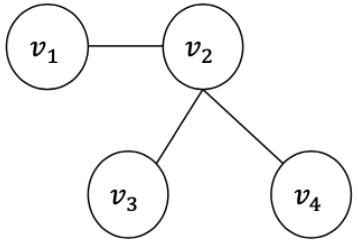
First Snapshot




$\mathcal{V}^{(1)} = \{v_1, v_2, v_3\}$

$$A^{(1)} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \; X^{(1)} = \begin{bmatrix} 0.1 & 1 \\ 0.2 & 1 \\ 0.2 & 2 \end{bmatrix}$$

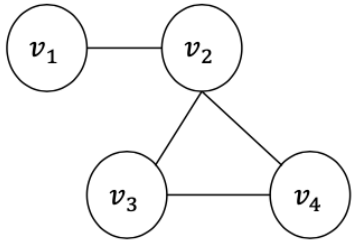
Second Snapshot




$\mathcal{V}^{(2)} = \{v_1, v_2, v_3, v_4\}$

$$A^{(2)} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \; X^{(2)} = \begin{bmatrix} 0.1 & 2 \\ 0.2 & 1 \\ 0.2 & 2 \\ 0.5 & 1 \end{bmatrix}$$

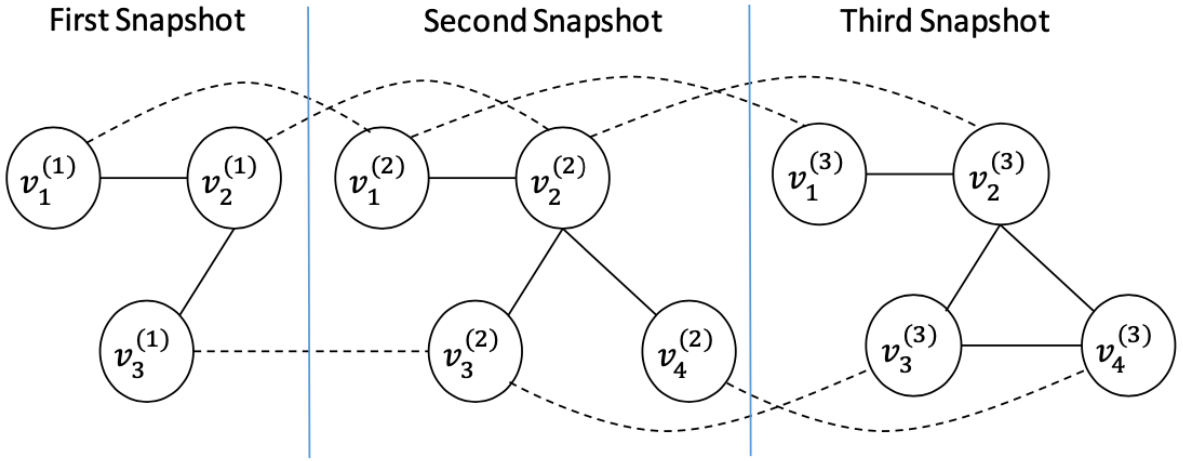
Third Snapshot

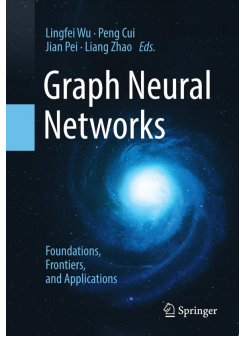



...

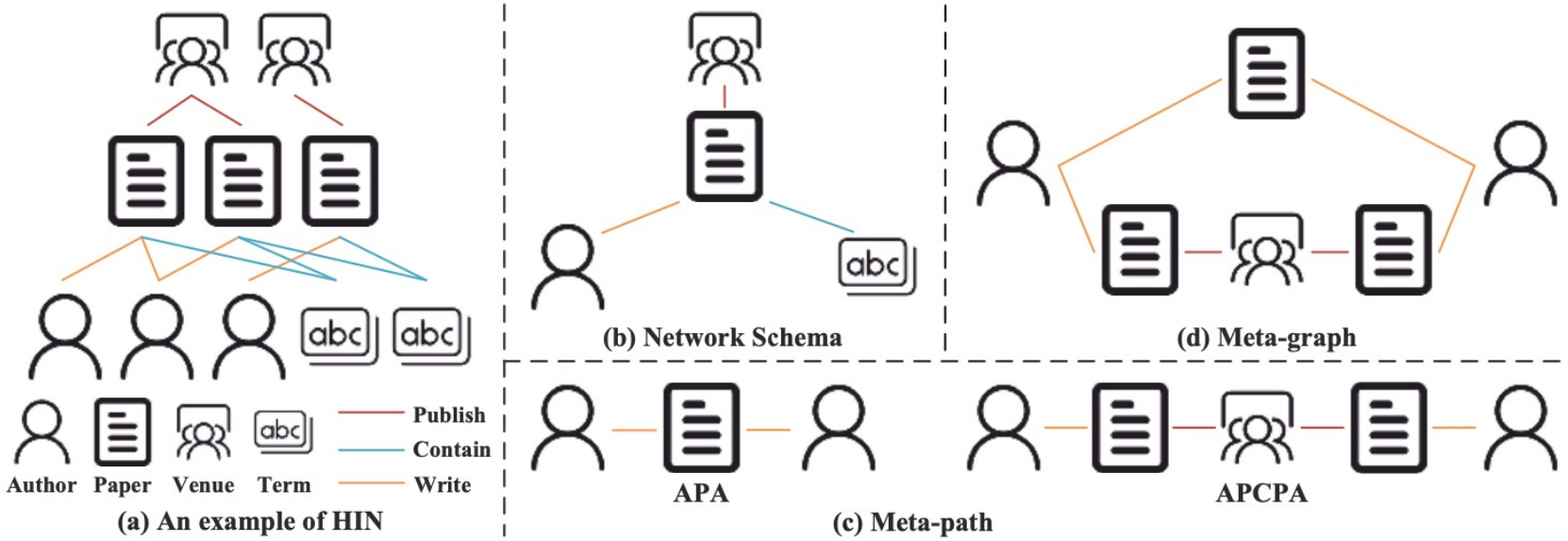
$\mathcal{V}^{(3)} = \{v_1, v_2, v_3, v_4\}$

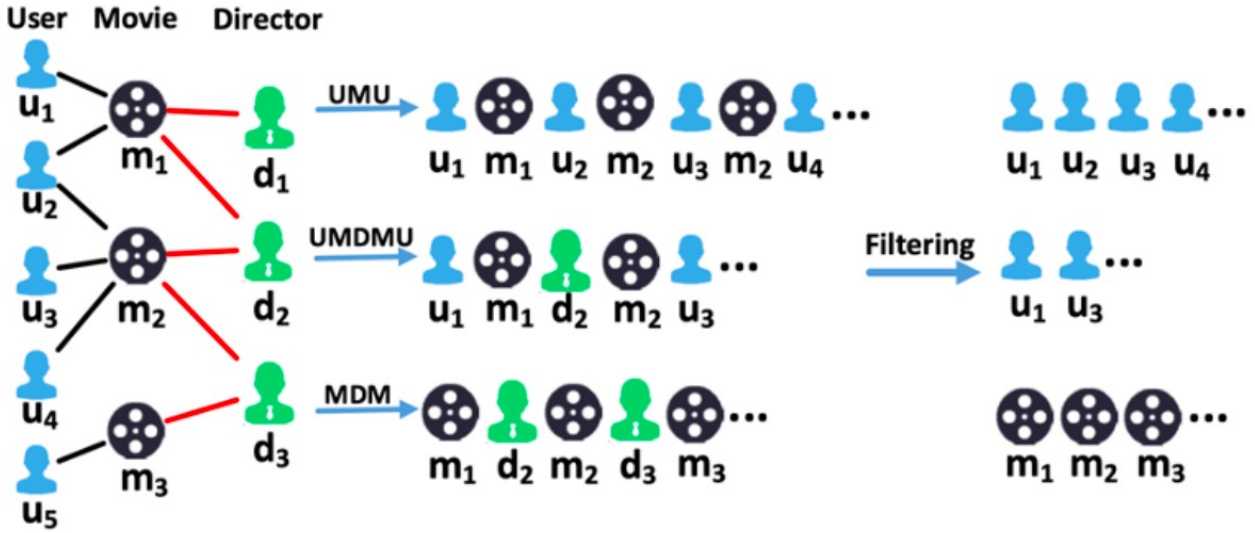
$$A^{(3)} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}, \; X^{(3)} = \begin{bmatrix} 0.1 & 2 \\ 0.2 & 1 \\ 0.2 & 2 \\ 0.5 & 1 \end{bmatrix}$$



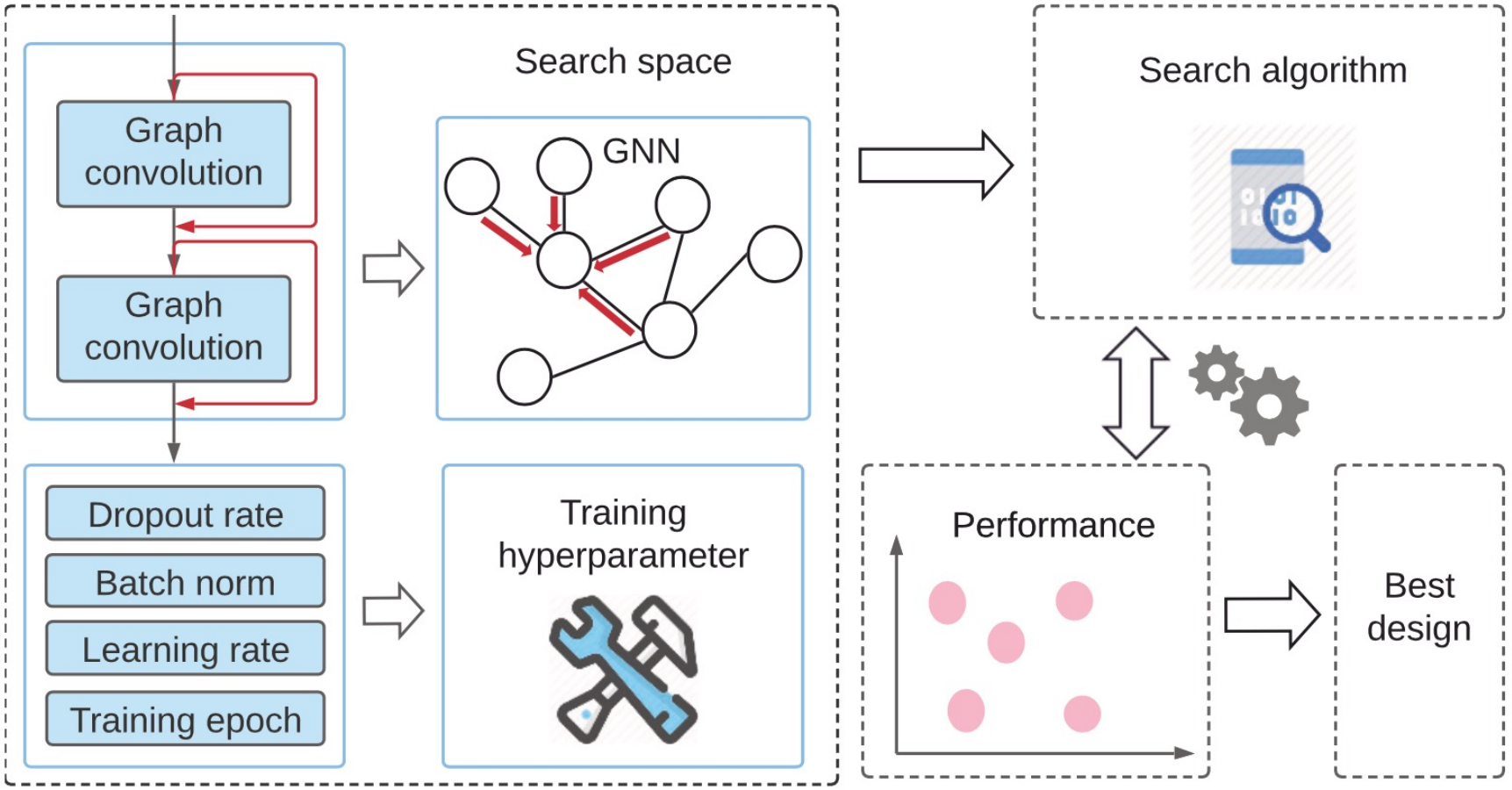



# Heterogeneous GNNs (Chapter 16)

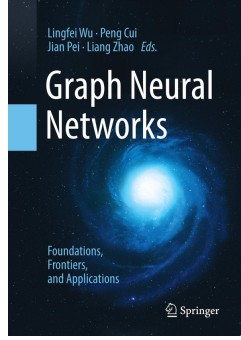


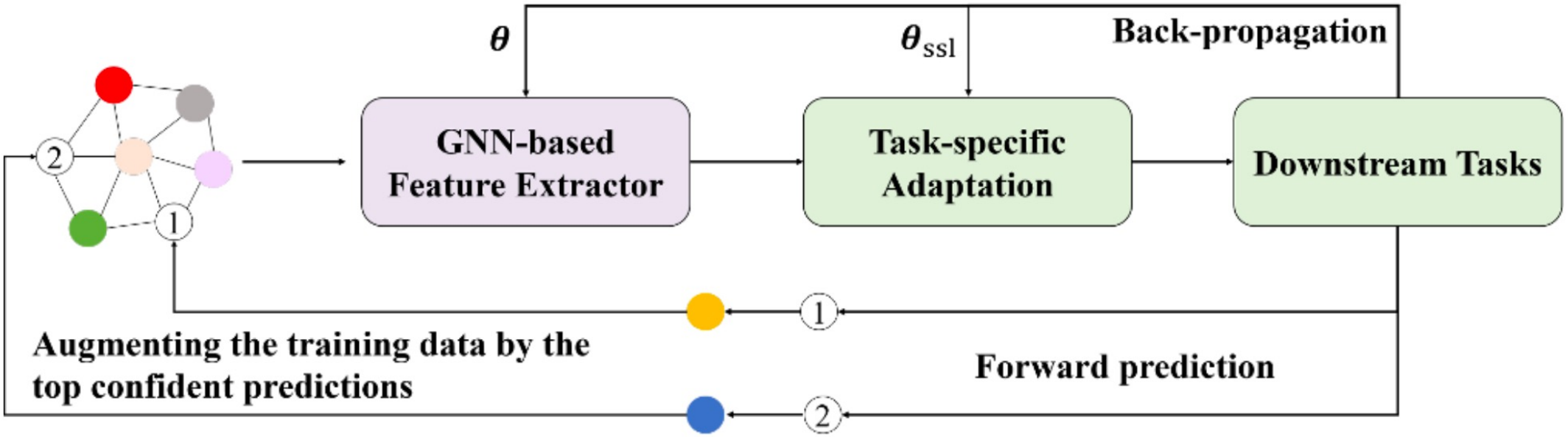



# GNNs: AutoML (Chapter 17)

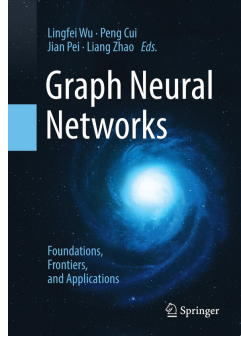


# GNNs: Self-supervised Learning (Chapter 18)



# Outline




**GNNs Foundations**

- Time History of GNNs
- GNNs: Foundations and Models
- GNNs: Theory, Scalability, Interpretability

**GNNs Frontiers**

- Graph Structure Learning
- Other Frontiers in GNNs

**GNNs Applications**

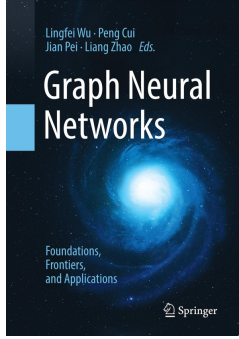
- GNNs in Recommendation System
- Other GNNs-based Applications

**GNN book website：**
https://graph-neural-networks.github.io/index.html

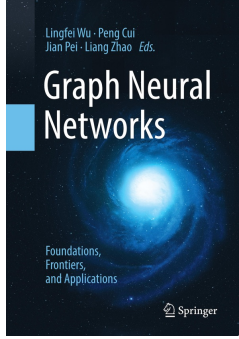
**GNN Springer：**
https://link.springer.com/book/10.1007/978-981-16-6054-2

**Amazon：**
https://www.amazon.com/Graph-Neural-Networks-Foundations-Applications/dp/9811660530

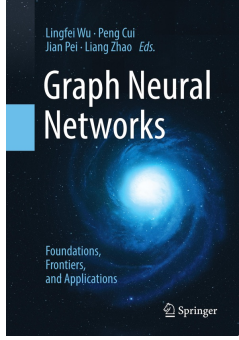
**JD.com (京东商城)：**
https://item.jd.com/10043589466641.html




# GNNs
# Applications

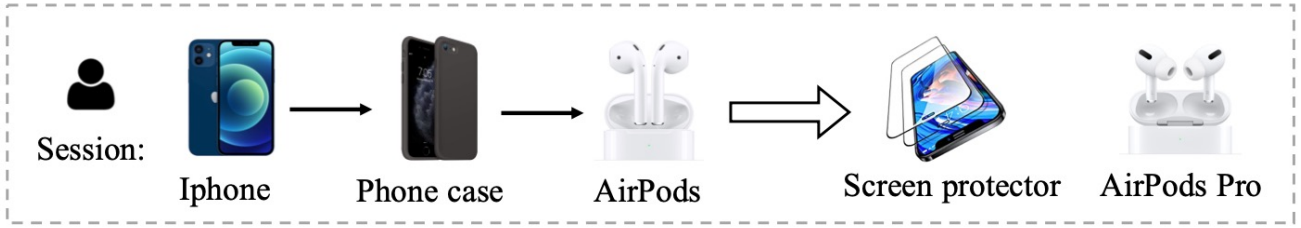


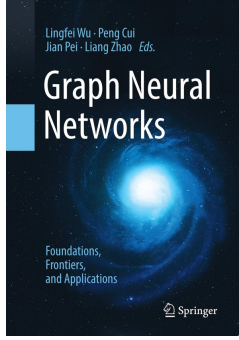

# GNNs in Recommendation System (Chapter 19)




# Session-based Recommendation

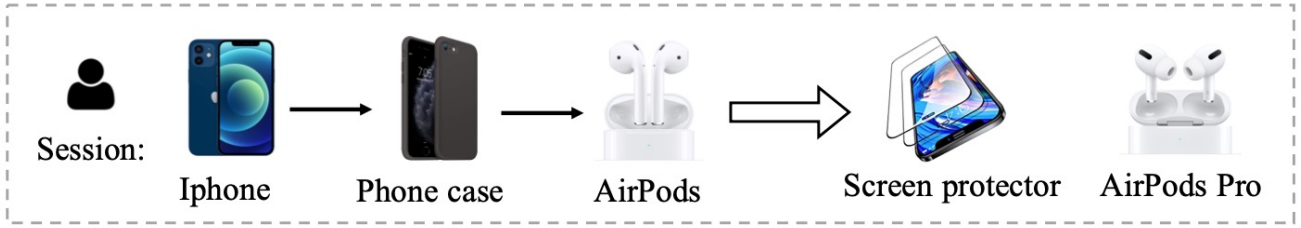
- Task Formulation
  - ➢ Input: an **interaction sequence** of the **ongoing session**
  - ➢ Task: predict the **successive items** that a user is likely to interact with.
  - ➢ **Key**: modeling the complex sequential dependencies embedded in the sequence of user-item interactions.



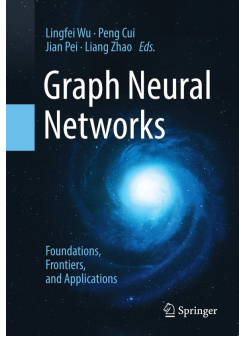



# Limitations

- Neglect **user historical sessions** leading to **non-personalized recommendation**
- Personalized recommender: ignore **other user's historical sessions**.

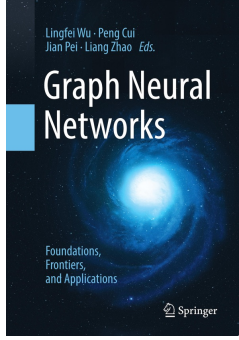



*Given similar session sequences from two different users,*
*the methods neglecting user historical sessions usually generate the same candidate items for different users.*



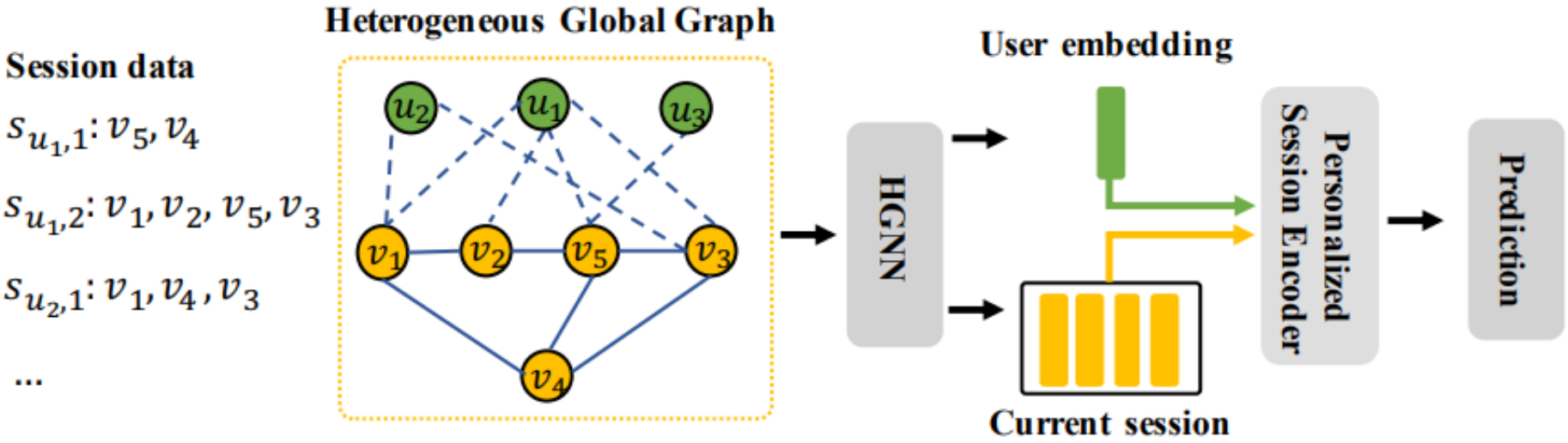

# Problem Formulation

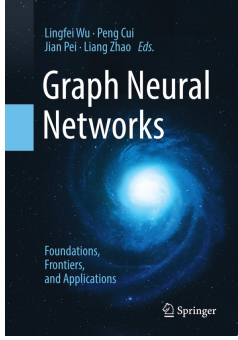
- Personalized session-based recommendation
  - Given item set $V = \{v_1, v_2, \dots, v_{|V|}\}$, user set $U = \{u_1, u_2, \dots, u_{|U|}\}$, and historical sessions:$S_{u_i} = \{S_{u_{i,1}}, \dots, S_{u_{i,n}}\}$ where $S_{u_{i,j}} = \{v_1^{u_{i,j}}, \dots, v_l^{u_{i,j}}\}$ and $v_t^{u_{i,j}} \in V$.
  - Input: **current session** $S_{u_{i,t}}$ of user $u_i$.
  - Output: interaction probability $\hat{y}_{v_i}$ for each candidate.




# Method

- Overview
  - ➢Heterogeneous Global Graph Construction
  - ➢Heterogeneous Global Graph Neural Network
  - ➢Personalized Session Encoder






# Method

- Build the Heterogeneous Global Graph
  - Effectively organize the historical sessions.
  - ***Item-to-Item***:
    - Adjacent interaction items in the same session. (*item transition patterns,* $r_{in}, r_{out}$)
    - Similar items based on global co-occurrence information. (*item correlations,* $r_{similar}$)

*co-occurrence frequency*

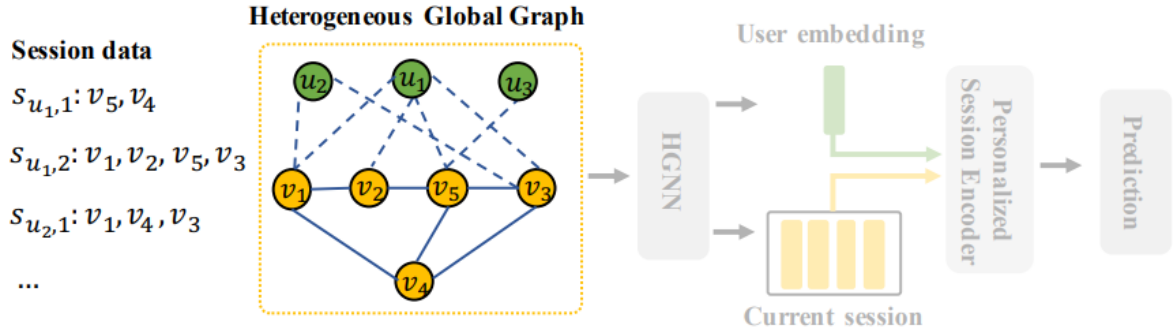
$$f_v(v_i, v_j) = \frac{\sum_{s \in N(v_i) \cap N(v_j)} \frac{1}{|N(s)|}}{\sqrt{|N(v_i)||N(v_j)|}}$$

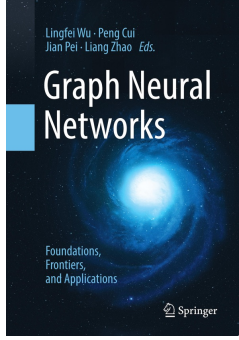
$$K'_{v_i} = min\{K, |N_{v_i}|\},$$

$$N_{v_i} = \{v_j | (v_i, v_j, r) \ where \ r \in \{r_{in}, r_{out}\}\}$$

*Similar item Cut-off*

  - ***Item-to-User***:
    - Interaction behaviors between the user and items. (*long-term user preferences,* $r_u$)






# Method

- Heterogeneous Global Graph Neural Network (HGNN)
  - Given the heterogeneous global graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
  - **Message propagation**

$$\mathbf{p}_{\mathcal{N}_{r_x}(v_i)}^{(k+1)} = \frac{1}{|\mathcal{N}_{r_x}(v_i)|} \sum_{n \in \mathcal{N}_{r_x}(v_i)} \mathbf{e}_n^{(k)},$$

$$\mathbf{p}_{v_i,r_x}^{(k+1)} = f(\mathbf{W}_{r_x}^{(k+1)} [\mathbf{p}_{\mathcal{N}_{r_x}(v_i)}^{(k+1)} || \mathbf{p}_{v_i}^{(k)}] + \mathbf{b}_{r_x}^{(k+1)})$$

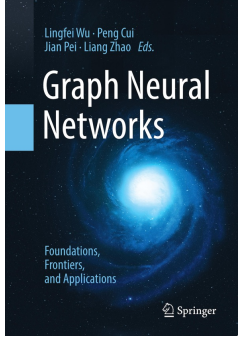
  - **Aggregation** and **Node update**

*Item node* $\mathbf{p}_{v_i}^{(k+1)} = \text{accum}(\mathbf{p}_{v_i,r_u}^{(k+1)}, \mathbf{p}_{v_i,r_{in}}^{(k+1)}, \mathbf{p}_{v_i,r_{out}}^{(k+1)}, \mathbf{p}_{v_i,r_{similar}}^{(k+1)})$

*Accumulate different messages*

*User node* $\mathbf{q}_{u_i}^{(k+1)} = \text{accum}(\mathbf{q}_{u_i,r_v}^{(k+1)})$

*Final node representation* $\mathbf{q}_{u_i} = \sum_{k=0}^{K} \alpha_k \mathbf{q}_{u_i}^{(k)}; \quad \mathbf{p}_{v_i} = \sum_{k=0}^{K} \alpha_k \mathbf{p}_{v_i}^{(k)}$




# Method

- Personalized Session Encoder
  - ➢ **Current Preference Learning**
    - User preferences are always dynamic and can be learned **from the current session**

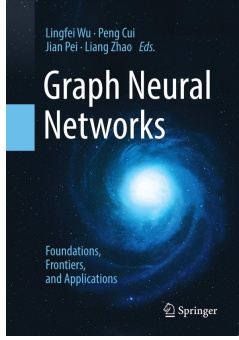
$$\mathbf{p}'_{v_i^s} = \mathbf{W}_c[\mathbf{p}_{v_i^s}||\mathbf{l}_i]$$ *Add position embedding*

$$\mathbf{p}'_s = \frac{1}{l}\sum_{i=1}^{l}\mathbf{p}'_{v_i^s}$$

$$\alpha_i = \mathrm{softmax}_i(\epsilon_i),$$

$$\epsilon_i = \mathbf{v}_0^T\sigma(\mathbf{W}_0\mathbf{p}'_{v_i^s} + \mathbf{W}_1\mathbf{p}'_s + \mathbf{b}_0)$$ *Attention mechanism*

$$\mathbf{C}_u = \sum_{i=1}^{l}\alpha_i\mathbf{p}'_{v_i^s}$$




# Method

- Personalized Session Encoder
  - ➢ **General Preference Learning**
    - Utilize user embedding learnt from HGNN which contains user long-term stable preference.
    - Consider the correlation between items in the current session and the user general preference.

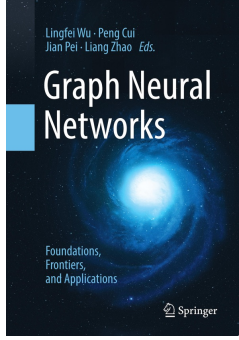
$$\gamma_i = \mathrm{softmax}_i(e_i),$$

$$e_i = \mathbf{v}_1^T \sigma(\mathbf{W}_2 \mathbf{p}_{v_i^s} + \mathbf{W}_3 \mathbf{q}_u + \mathbf{b}_1)$$

$$\mathbf{O}_u = \sum_{i=1}^{l} \gamma_i \mathbf{p}_{v_i^s}$$

*Item embeddings in the current session*

*User embedding*



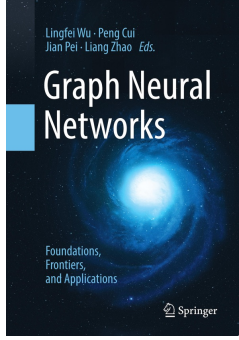

# Method

- Personalized Session Representation

$$\alpha_c = \sigma(\mathbf{W}_s[\mathbf{C}_u||\mathbf{O}_u]),$$
$$\mathbf{S}_u = \alpha_c \cdot \mathbf{C}_u + (1-\alpha_c)\cdot \mathbf{O}_u$$

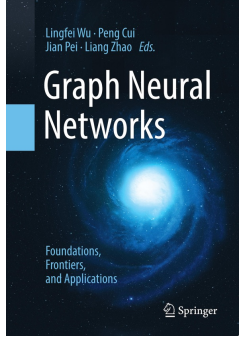
*Using **Gate mechanism to** combine representations*

- Prediction and Training
  - ➢Next-item Probability: $\hat{y}_i = \text{softmax}(\mathbf{S}^T \mathbf{p}_{v_i}^{(0)})$
  - ➢Objective Function: $\mathcal{L}(\hat{\mathbf{y}}) = -\sum_{i=1}^{|V|} y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)$




# Experiments: Overview

| Models | Last.fm | | | | Xing | | | | Reddit | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | HR@5 | HR@10 | MRR@5 | MRR@10 | HR@5 | HR@10 | MRR@5 | MRR@10 | HR@5 | HR@10 | MRR@5 | MRR@10 |
| ItemKNN | 6.73 | 10.90 | 4.02 | 4.81 | 8.79 | 11.85 | 5.01 | 5.42 | 21.71 | 30.32 | 11.74 | 12.88 |
| GRU4Rec | 8.47 | 12.86 | 4.71 | 5.29 | 10.35 | 13.15 | 5.94 | 6.36 | 33.72 | 41.73 | 24.36 | 25.42 |
| NARM | 10.29 | 15.03 | 6.09 | 6.71 | 13.51 | 17.31 | 8.87 | 9.37 | 33.25 | 40.52 | 24.56 | 25.52 |
| SR-GNN | 11.89 | 16.90 | 7.23 | 7.85 | 13.38 | 16.71 | 8.95 | 9.39 | 34.96 | 42.38 | 25.90 | 26.88 |
| LESSR | 12.96* | 17.88 | 8.24* | 8.82* | 14.84 | 16.77 | 11.98* | 12.13* | 36.03 | 43.27 | 26.45 | 27.41 |
| GCE-GNN | 12.83 | 18.28* | 7.60 | 8.32 | 16.98* | 20.86* | 11.14 | 11.65 | 36.30 | 45.16 | 26.65 | 27.70 |
| H-RNN | 10.92 | 15.83 | 6.71 | 7.39 | 10.72 | 14.36 | 7.22 | 7.74 | 44.76 | 53.44 | 32.13 | 33.29 |
| A-PGNN | 12.10 | 17.13 | 7.37 | 8.01 | 14.23 | 17.01 | 10.26 | 10.58 | 49.10* | 58.23* | 33.54* | 34.62* |
| HG-GNN | **13.09**† | **19.39**† | 7.35 | 8.18 | **17.25**† | 20.30 | **12.23**† | **12.79**† | **51.08**† | **60.51**† | **35.46**† | **36.89**† |
| *Improv.* | 1.00% | 6.07% | - | - | 1.59% | - | 2.09% | 5.44% | 4.03% | 3.92% | 5.72% | 6.56% |



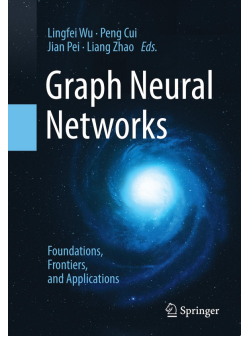

# Experiments: Ablation Study

**Table 4: Impact of different layer of HG-GNN.**

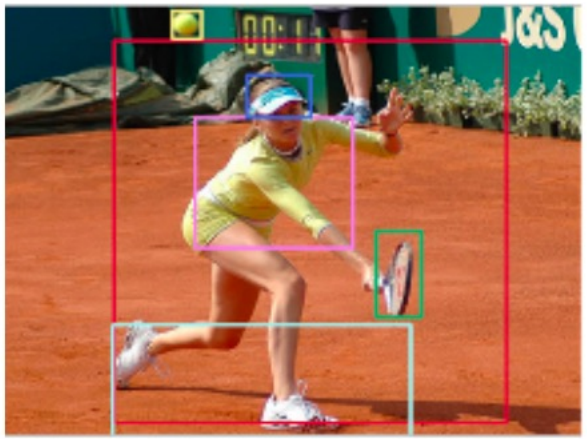
| Model setting | Last.fm | | | | Xing | | | |
|---|---|---|---|---|---|---|---|---|
| | HR@5 | HR@10 | MRR@5 | MRR@10 | HR@5 | HR@10 | MRR@5 | MRR@10 |
| w/o HGNN module | 11.98 | 17.42 | 6.81 | 7.53 | 16.54 | 20.14 | 11.76 | 12.24 |
| w/o CPL module | 12.83 | 19.16 | 7.07 | 7.91 | 13.16 | 16.53 | 8.70 | 9.15 |
| w/o GPL module | 12.96 | 19.22 | 7.13 | 8.04 | 17.24 | 20.38 | 12.21 | 12.74 |
| HG-GNN | 13.09 | 19.39 | 7.35 | 8.18 | 17.25 | 20.30 | 12.23 | 12.79 |

**Table 5: The performance comparison w.r.t. different graph design on *Xing*.**

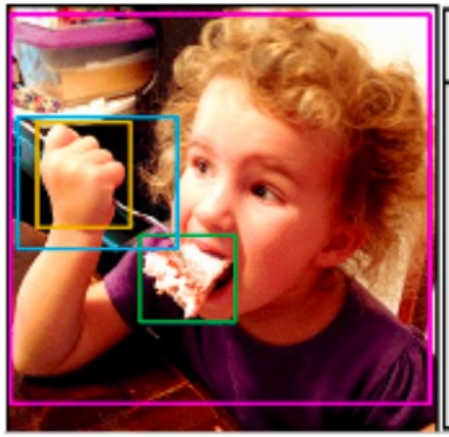
| Graph Setting | HR@5 | HR@10 | MRR@5 | MRR@10 |
|---|---|---|---|---|
| w/o *user* nodes | 17.03 | 19.82 | 12.05 | 12.86 |
| w/o *in* edges | 17.14 | 20.02 | 12.12 | 12.93 |
| w/o *out* edges | 17.11 | 20.00 | 12.02 | 12.87 |
| w/o *similar* edges | 17.24 | 20.09 | 12.32 | 13.04 |
| HG-GNN | 17.25 | 20.30 | 12.23 | 12.79 |

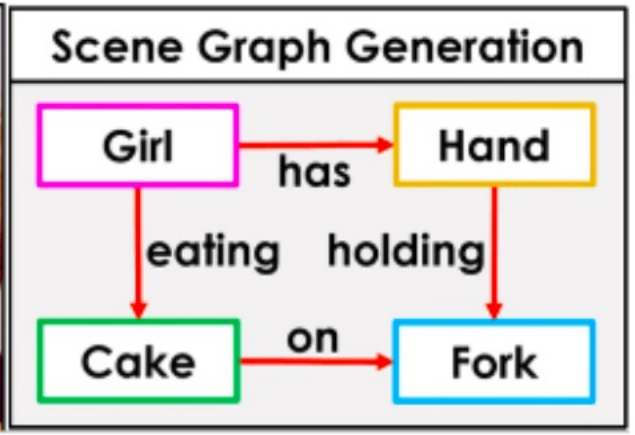



# GNNs in Computer Vision (Chapter 20)

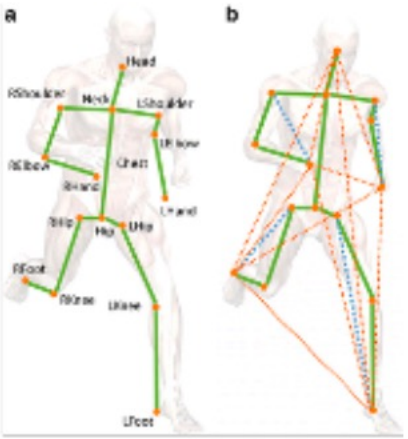


object detection

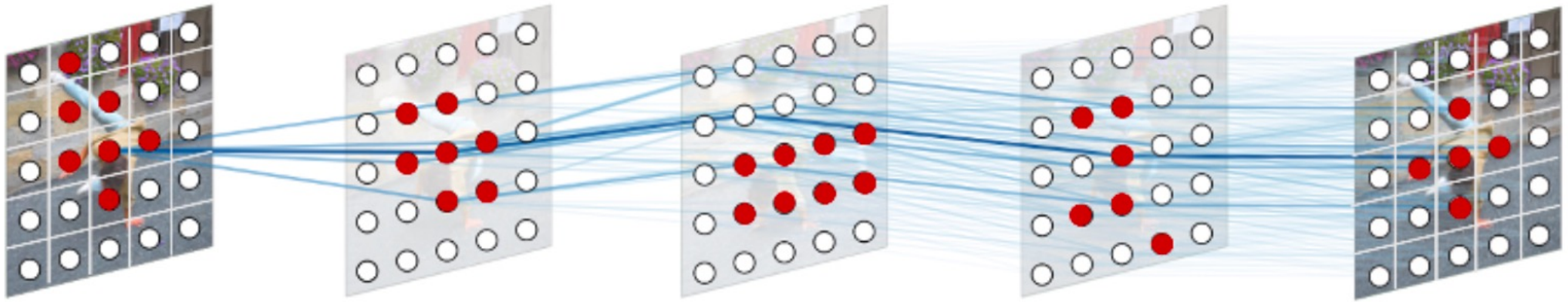




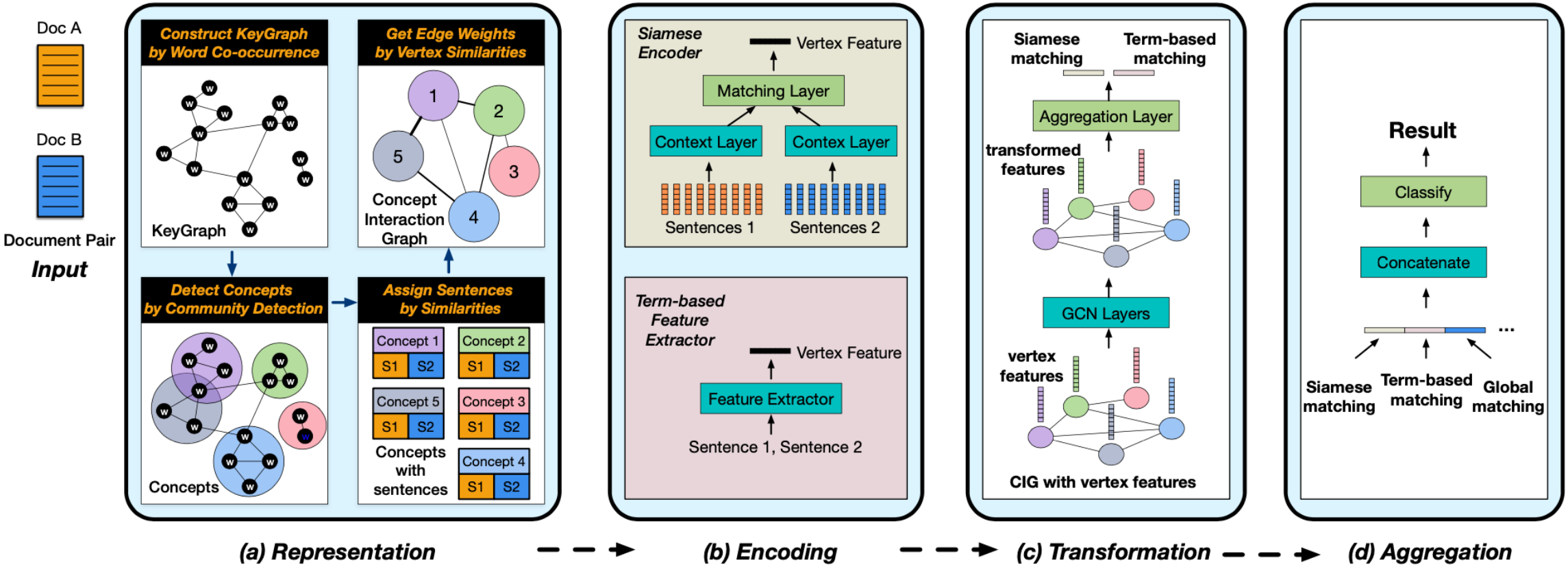

scene graph generation



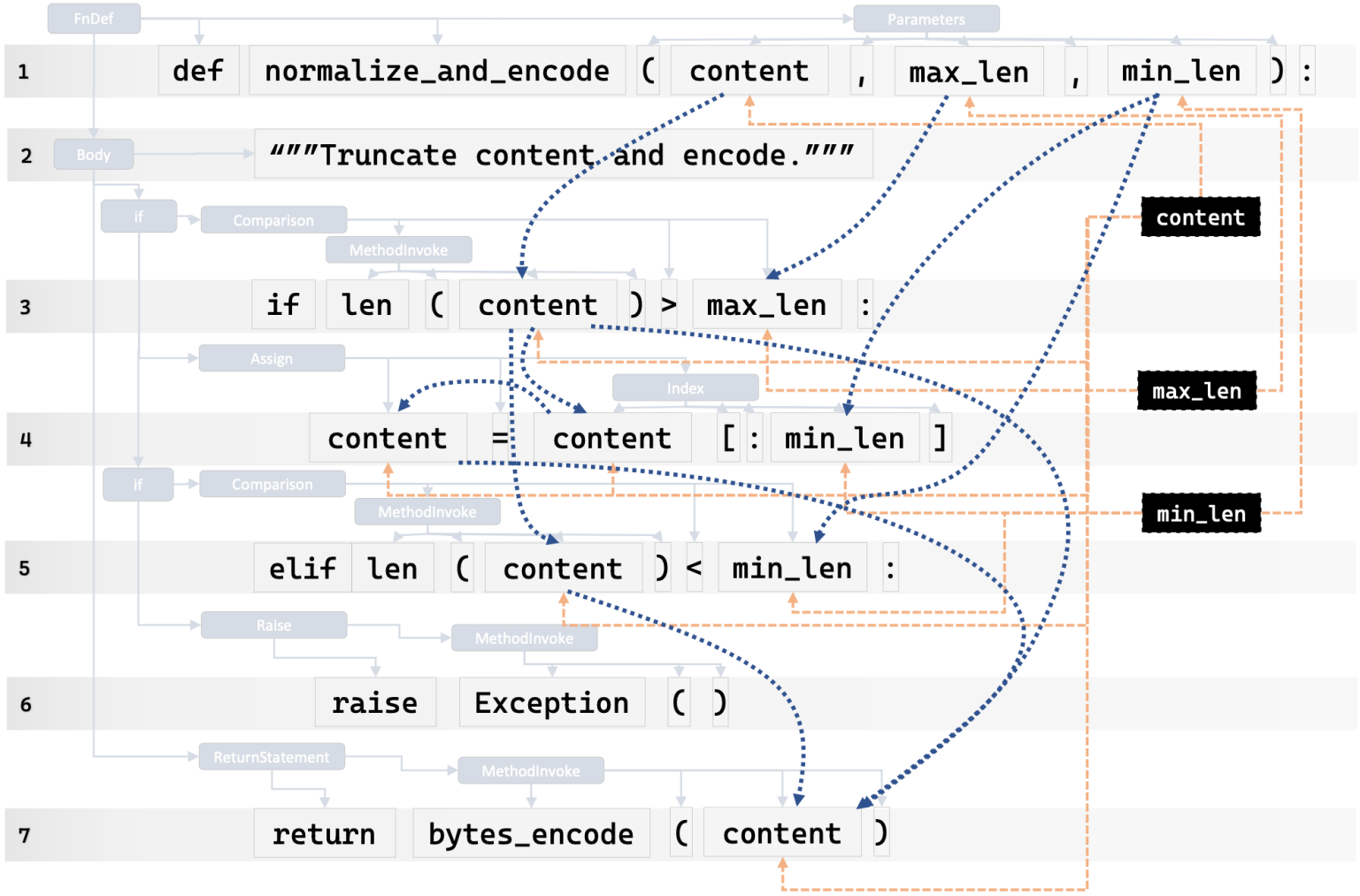
skeleton



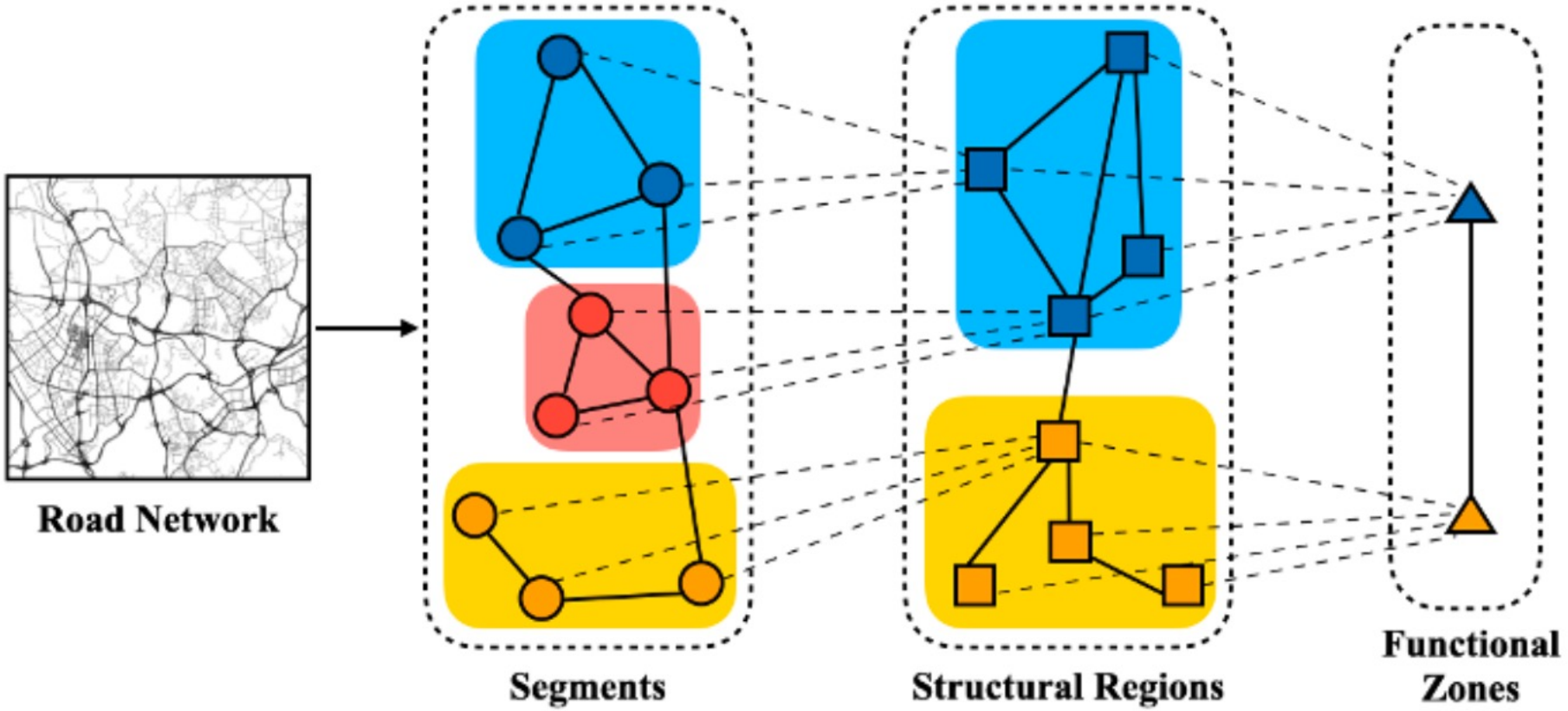
# GNNs in Natural Language Processing (Chapter 21)

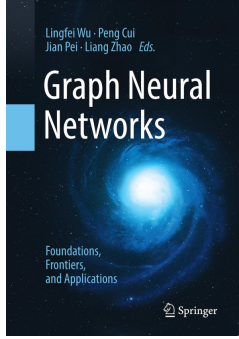


# GNNs in Program Analysis (Chapter 22)

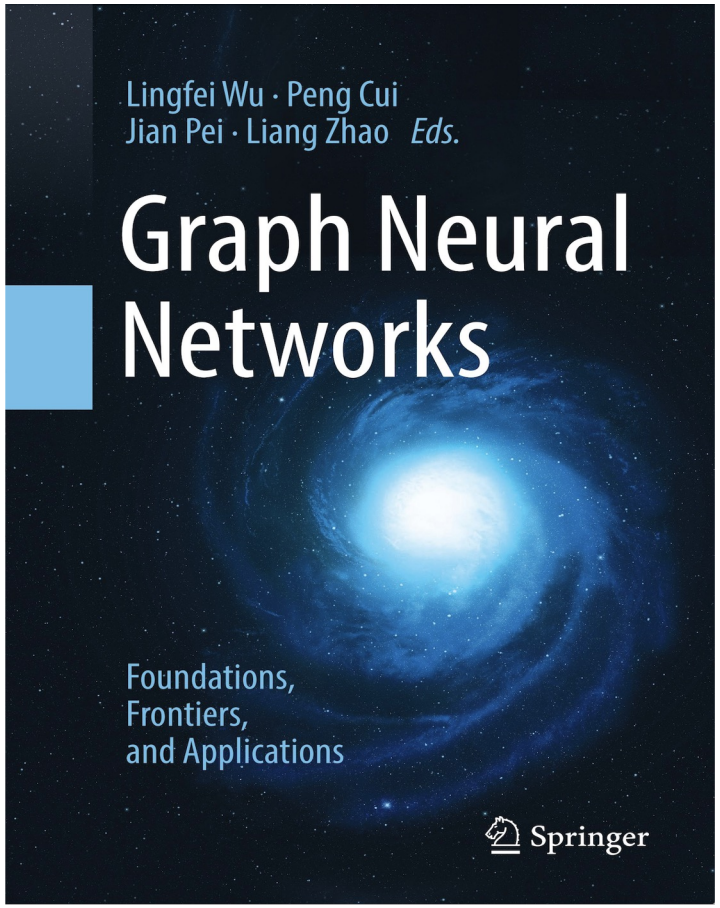


# GNNs in Urban Intelligence (Chapter 27)

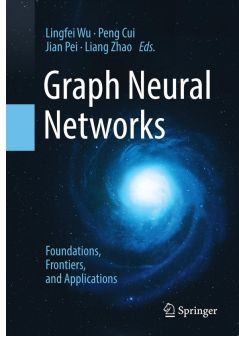


# Graph Neural Networks Book








**Free download: https://graph-neural-networks.github.io/index.html**

The English version of the book is available for on Springer, Amazon and JD.COM !
The Chinese version (Posts & Telecom Press, 人民邮电出版社) will be officially published in the end of 2022!!