

# COMP541

## DEEP LEARNING

### Lecture #9 — Graph Neural Networks

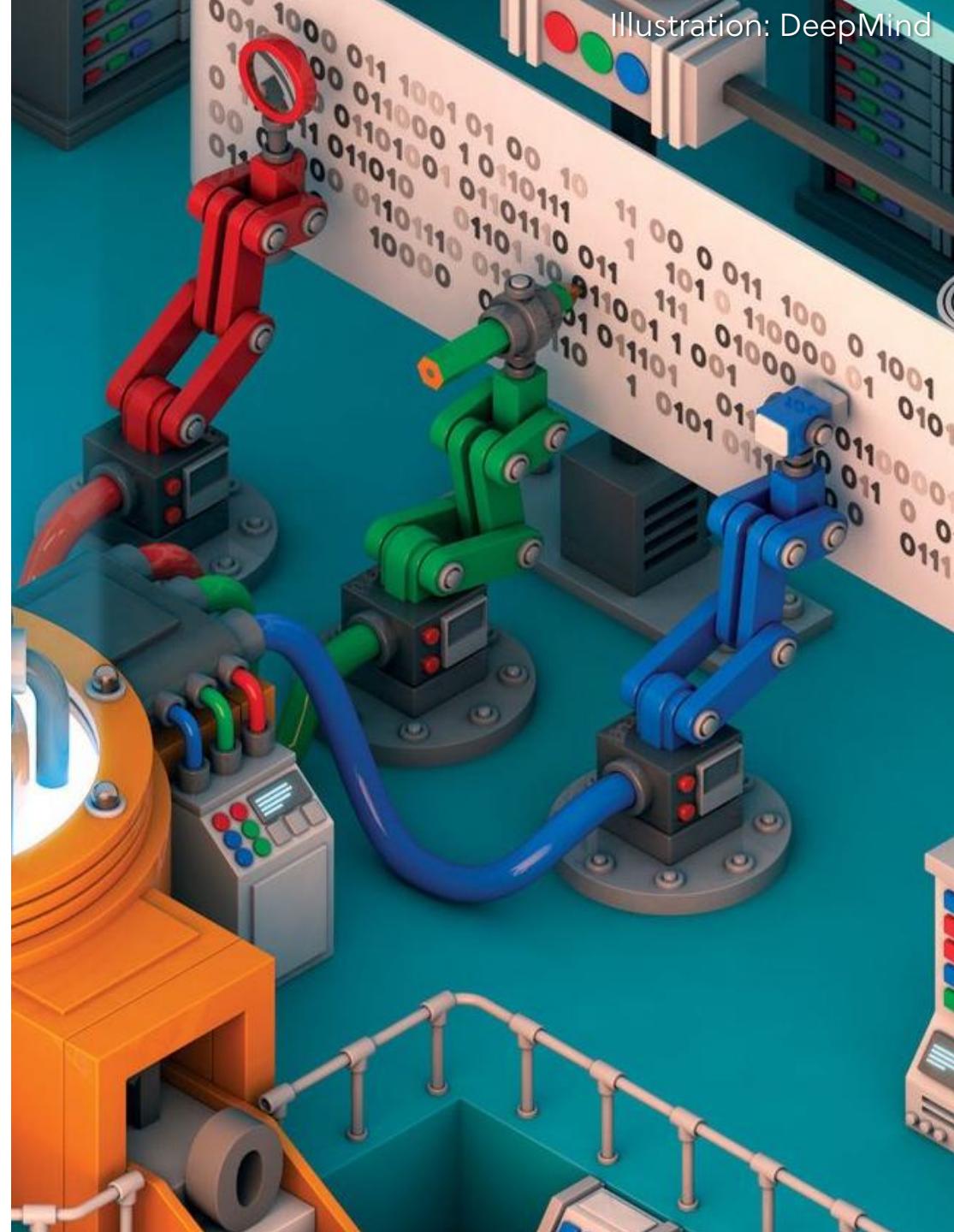


**KOÇ**  
**UNIVERSITY**

Aykut Erdem // Koç University // Fall 2023

# Previously on COMP541

- content-based attention
- location-based attention
- soft vs. hard attention
- case study: Show, Attend and Tell
- self-attention
- case study: Transformer networks



# Lecture overview

- graph structured data
- graph neural nets (GNNs)
- GNNs for “classical” network problems
  
- **Disclaimer:** Much of the material and slides for this lecture were borrowed from
  - Yujia Li and Oriol Vinyals' tutorial on Graph Nets
  - Thomas Kipf's talk on structured deep models: deep Learning on graphs and beyond
  - Minji Yoon's CMU 10707 slides

# Deep Learning

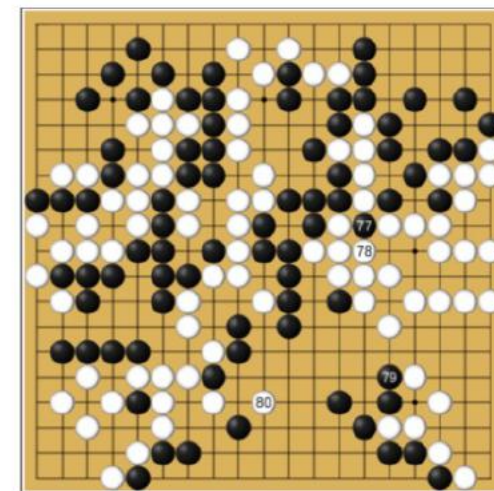
IMAGENET



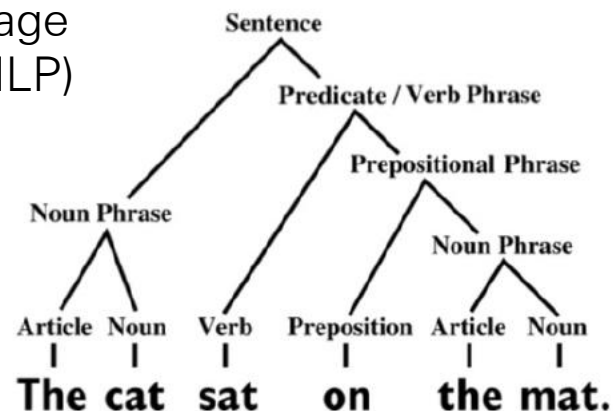
Speech data



Grid games

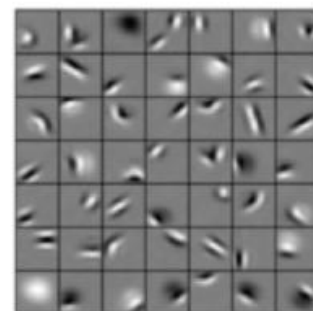


Natural language processing (NLP)



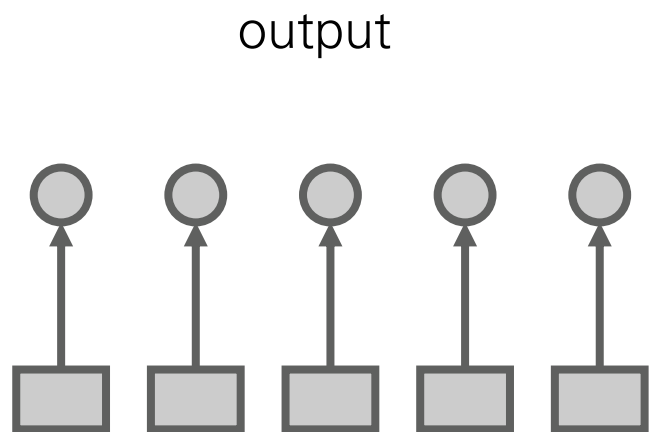
Deep neural nets that exploit:

- translation equivariance (weight sharing)
- hierarchical compositionality

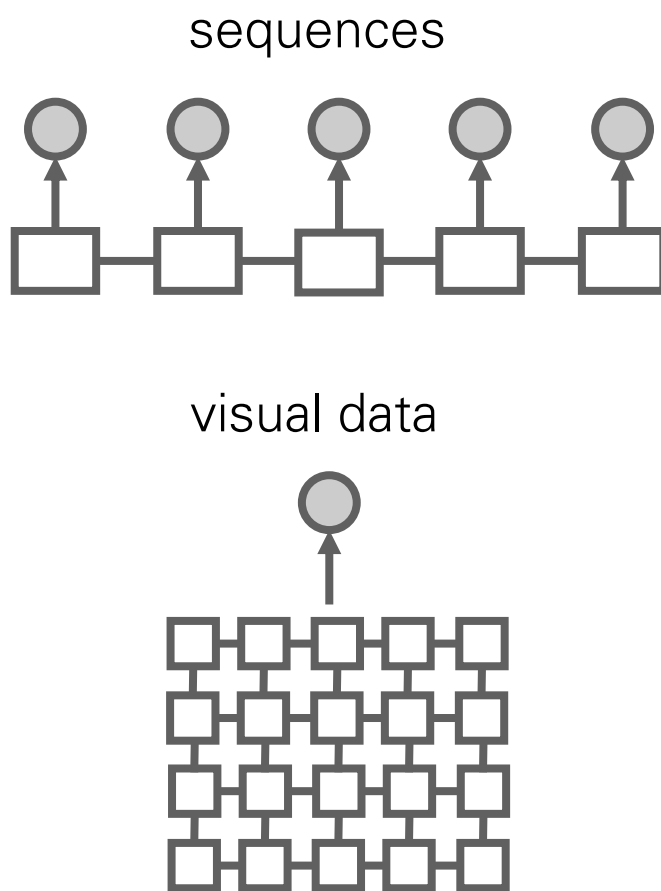


# Modeling Structured Data

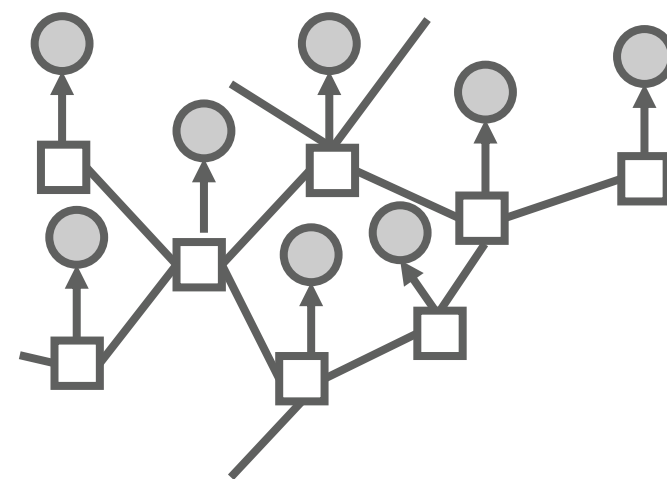
Unstructured Data



Data with Rigid Structure

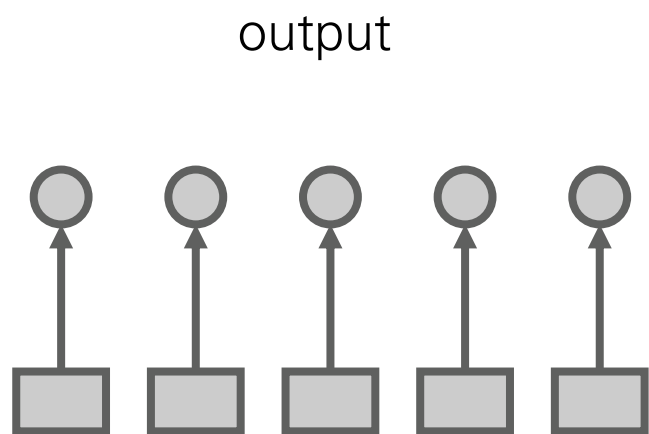


Graph Structured Data

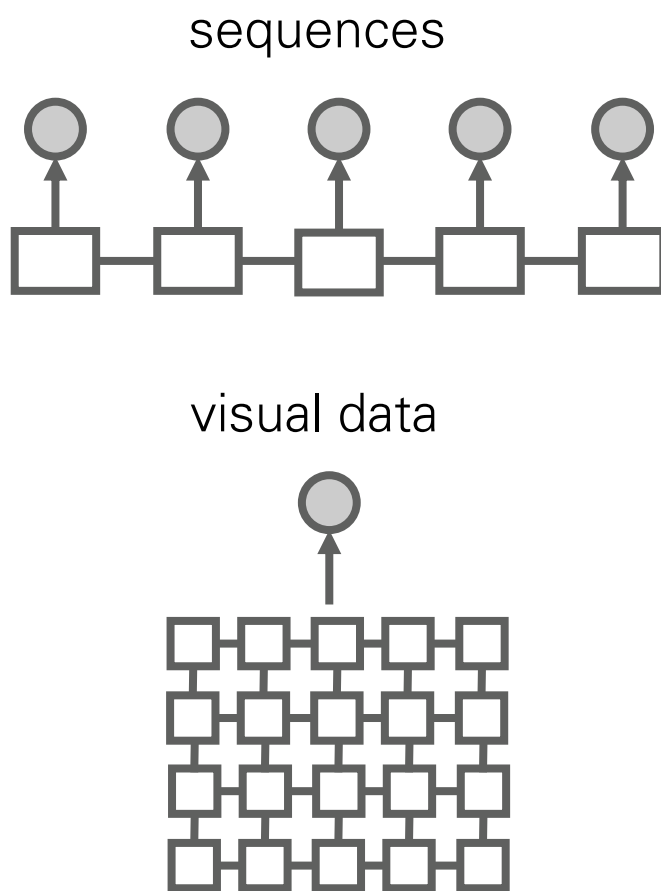


# Modeling Structured Data

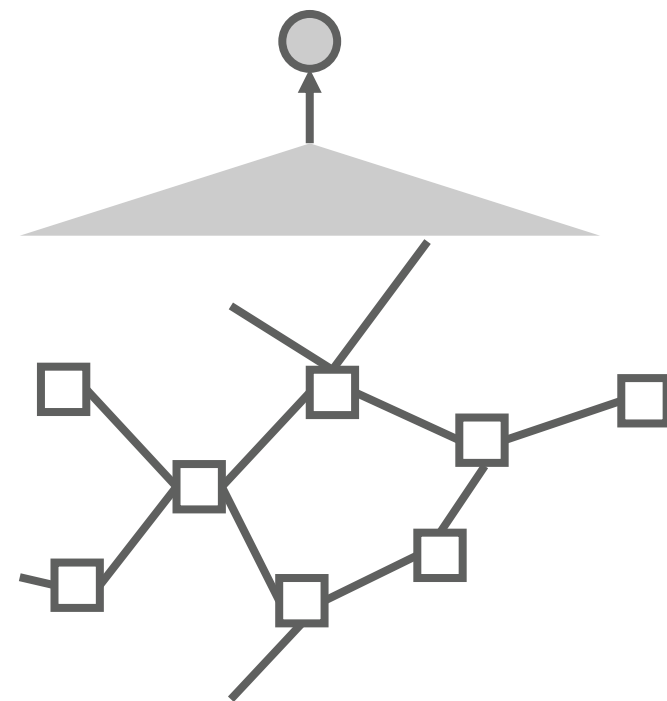
Unstructured Data



Data with Rigid Structure



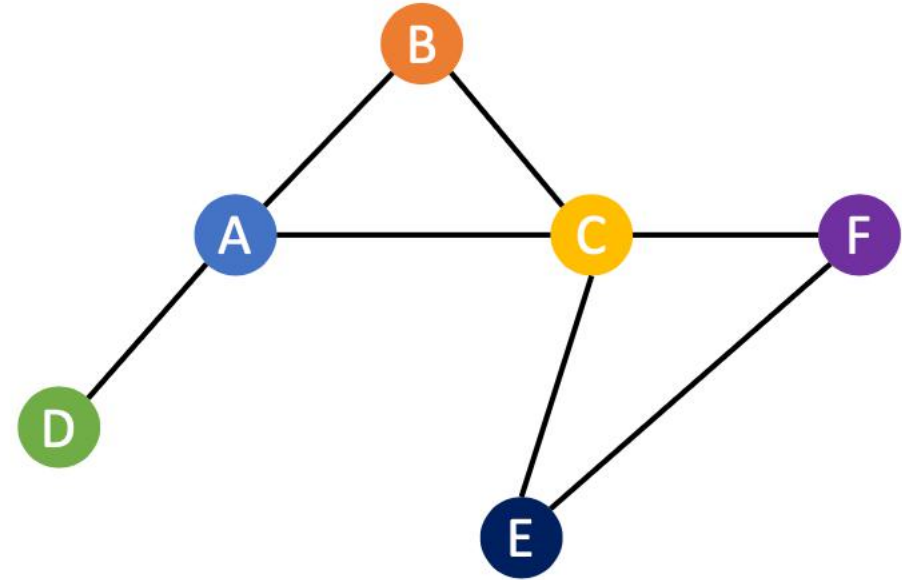
Graph Structured Data



# What is a graph

- A graph is composed of
  - **Nodes** (also called vertices)
  - **Edges** connecting a pair of nodespresented in an **adjacency matrix**

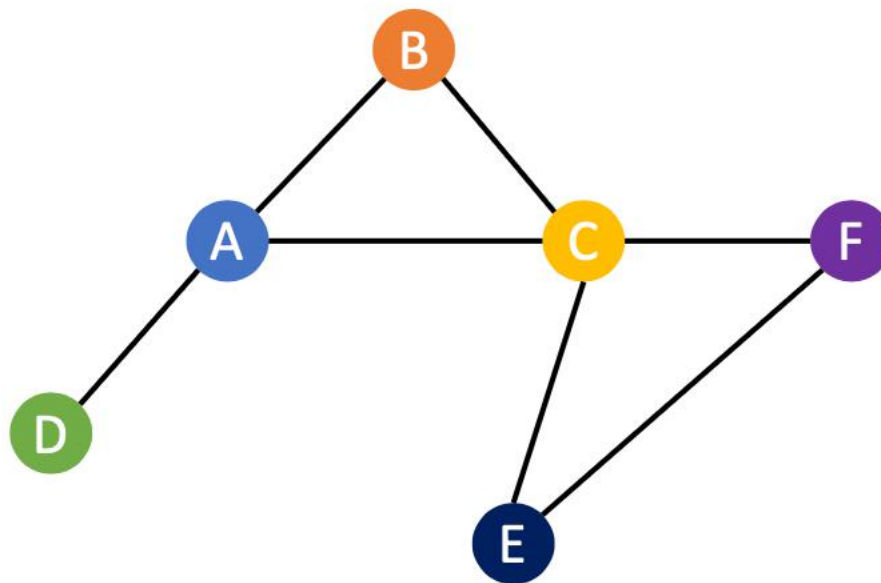
	A	B	C	D	E	F
A		1	1	1		
B	1		1			
C	1	1			1	1
D	1					
E			1			1
F			1		1	



# What is a graph

- A graph is composed of
  - **Nodes** (also called vertices)
  - **Edges** connecting a pair of nodespresented in an **adjacency matrix**
- Nodes can have **feature vectors**

A	$X_A$
B	$X_B$
C	$X_C$
D	$X_D$
E	$X_E$
F	$X_F$

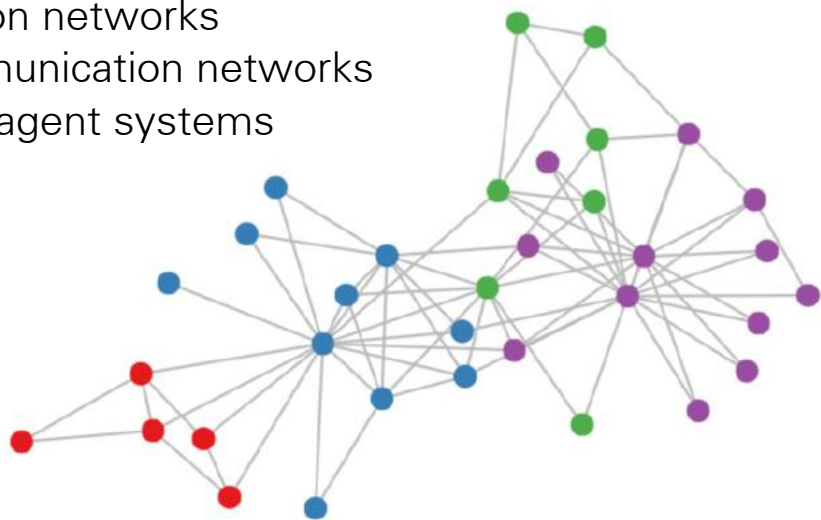




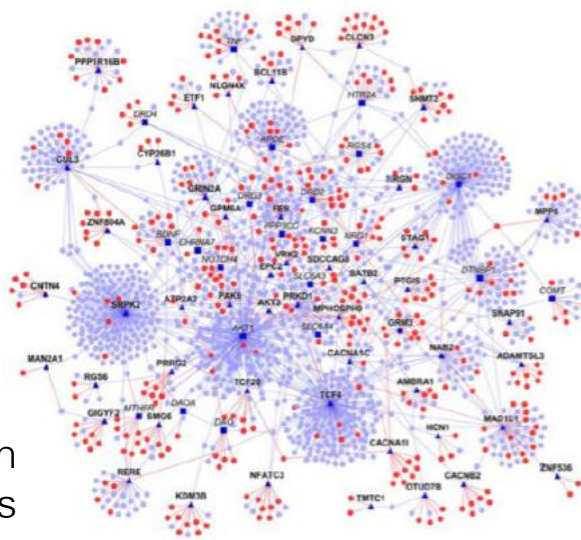
# Graph structured data

- A lot of real-world data does not “live” on grids

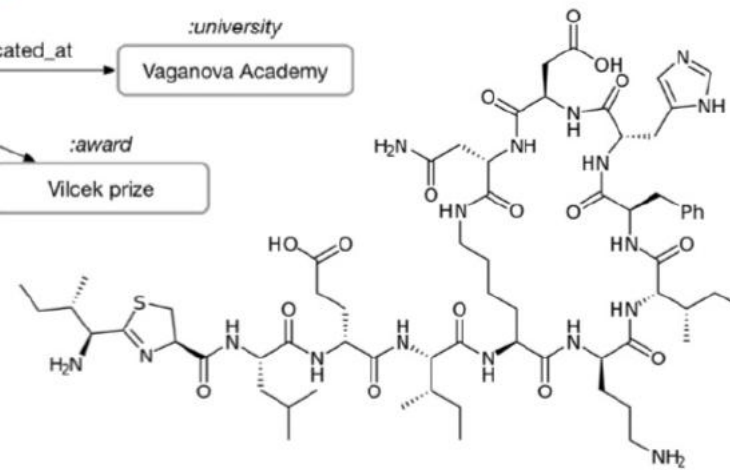
Social networks  
Citation networks  
Communication networks  
Multi-agent systems



Protein interaction networks

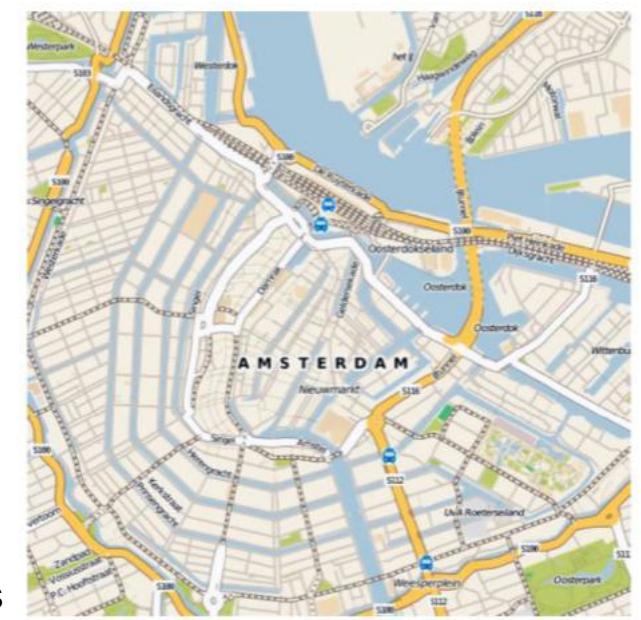


Knowledge graphs



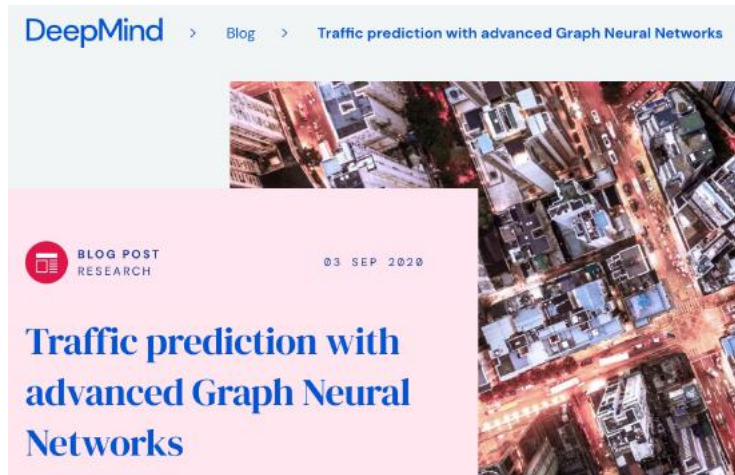
Molecules

**Standard deep learning architectures like CNNs and RNNs don't work here!**



Road maps

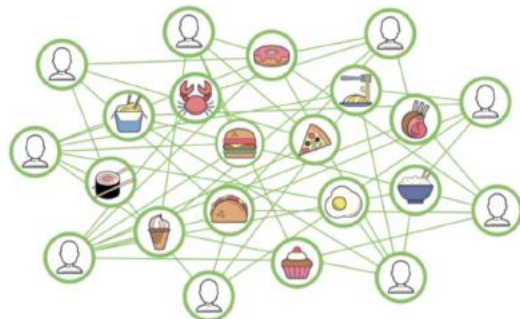
# Graph Neural Networks have a large impact on...




## Food Discovery with Uber Eats: Using Graph Learning to Power Recommendations

Ankit Jain, Isaac Liu, Ankur Sarda, and Piero Molino

December 4, 2019



 Pinterest Engineering  
Aug 15, 2018 · 8 min read

## PinSage: A new graph convolutional neural network for web-scale recommender systems

Ruining He | Pinterest engineer, Pinterest Labs

### Web image search gets better with graph neural networks

A new approach to image search uses images returned by traditional search methods as nodes in a graph neural network through which similarity signals are linking in cross-modal retrieval.

 | science



FOLLOW

PUBLICATION

> P-Companion: A principled framework for diversified complementary product recommendation

By Junheng Hao, Tong Zhao, Jin Li, Xin Luna Dong, Christos Faloutsos, Yizhou Sun, Wei Wang  
2020

# Graph Neural Networks have a large impact on...

## GCN-RL Circuit Designer: Transferable Transistor Sizing with Graph Neural Networks and Reinforcement Learning

Hanrui Wang<sup>1</sup>, Kuan Wang<sup>1</sup>, Jiacheng Yang<sup>1</sup>, Linxiao Shen<sup>2</sup>, Nan Sun<sup>2</sup>, Hae-Seung Lee<sup>1</sup>, Song Han<sup>1</sup>

<sup>1</sup>Massachusetts Institute of Technology

<sup>2</sup>UT Austin



## The next big thing: the use of graph neural networks to discover particles

September 24, 2020 | Zack Savitsky



Machine learning algorithms can beat the world's hardest video games in minutes and solve complex equations faster than the collective efforts of generations of physicists. But the conventional algorithms still struggle to pick out stop signs on a busy street.

Object identification continues to hamper the field of machine learning — especially when the pictures are multidimensional and complicated, like the ones particle detectors take of collisions in high-energy physics experiments. However, a new class of neural networks is helping these models boost their pattern recognition abilities, and the technology may soon be implemented in particle physics experiments to optimize data analysis.

npj | computational materials

Explore content ▾ About the journal ▾ Publish with us ▾

[nature](#) > [npj computational materials](#) > [articles](#) > [article](#)

Article | [Open Access](#) | [Published: 03 June 2021](#)

## Benchmarking graph neural networks for materials chemistry

[Victor Fung](#) , [Jiaxin Zhang](#), [Eric Juarez](#) & [Bobby G. Sumpter](#)

[npj Computational Materials](#) **7**, Article number: 84 (2021) | [Cite this article](#)

7807 Accesses | 7 Citations | 41 Altmetric | [Metrics](#)

nature

[View all journals](#) [Search](#) 

Explore content ▾ About the journal ▾ Publish with us ▾

[nature](#) > [articles](#) > [article](#)

Article | [Published: 09 June 2021](#)

## A graph placement methodology for fast chip design

[Azalia Mirhoseini](#) , [Anna Goldie](#) , [Mustafa Yazgan](#), [Joe Wenjie Jiang](#), [Ebrahim Songhori](#), [Shen Wang](#), [Young-Joon Lee](#), [Eric Johnson](#), [Omkar Pathak](#), [Azade Nazi](#), [Jiwoo Pak](#), [Andy Tong](#), [Kavya Srinivasa](#), [William Hang](#), [Emre Tuncer](#), [Quoc V. Le](#), [James Laudon](#), [Richard Ho](#), [Roger Carpenter](#) & [Jeff Dean](#)

# Graph Neural Networks have a large impact on...

nature

Explore content ▾ About the journal ▾ Publish with us ▾ Subscribe

[nature](#) > [news](#) > [article](#)

NEWS | 01 December 2021

## DeepMind's AI helps untangle the mathematics of knots

The machine-learning techniques could benefit other areas of maths that involve large data sets.

Patterns

Opinion

## Neural algorithmic reasoning

Petar Veličković<sup>1,\*</sup> and Charles Blundell<sup>1</sup>

<sup>1</sup>DeepMind, London, Greater London, UK

\*Correspondence: [petarv@google.com](mailto:petarv@google.com)

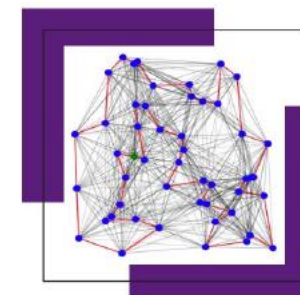
<https://doi.org/10.1016/j.patter.2021.100273>

We present neural algorithmic reasoning—the art of building neural networks that are able to execute algorithmic computation—and provide our opinion on its transformative potential for running classical algorithms on inputs previously considered inaccessible to them.

**ipam** institute for pure & applied mathematics

## Deep Learning and Combinatorial Optimization

February 22 - 25, 2021

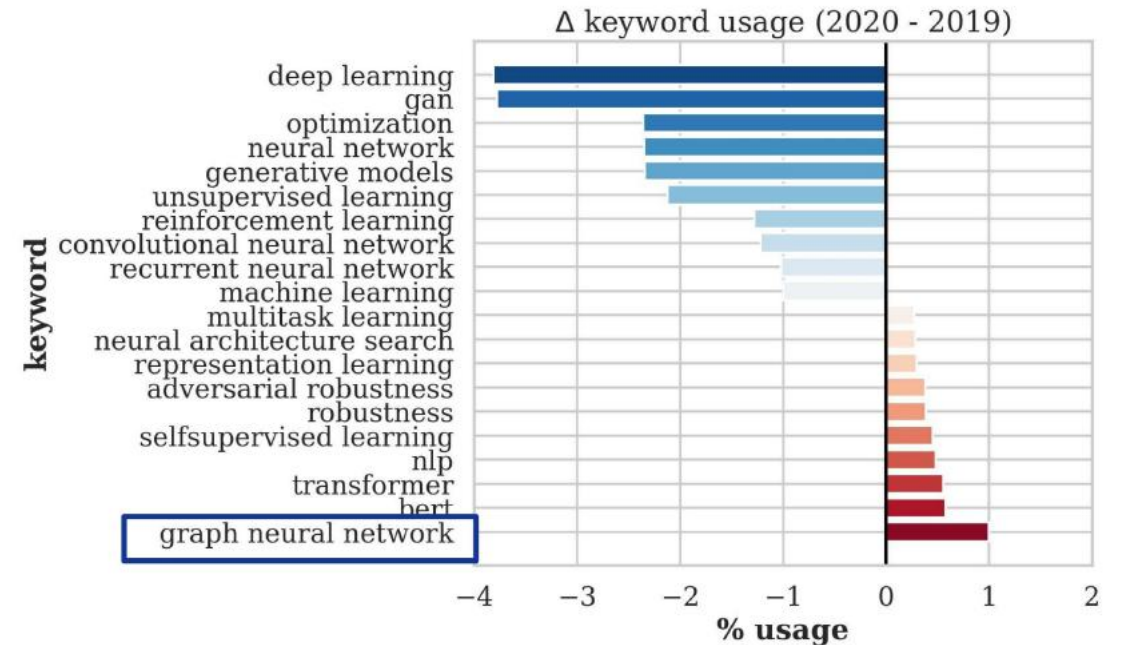
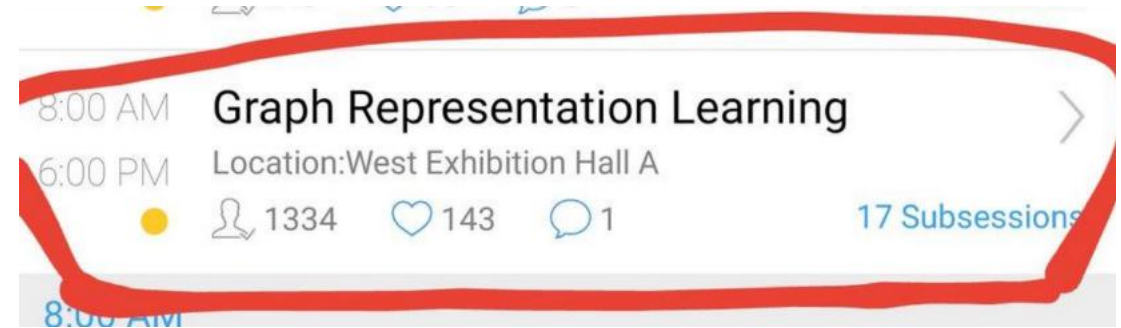
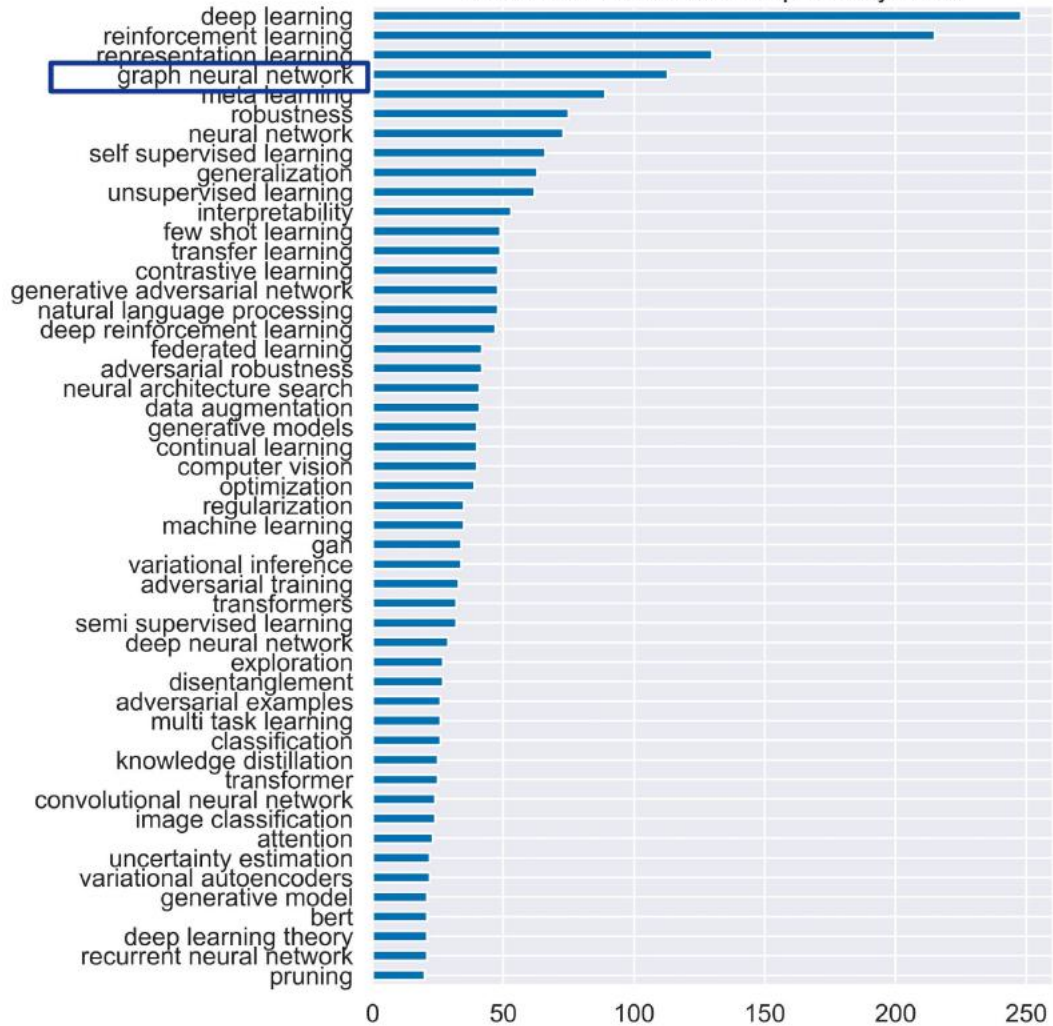


CellPress  
OPEN ACCESS



# A very hot research topic

ICLR 2021 Submission Top 50 Keywords



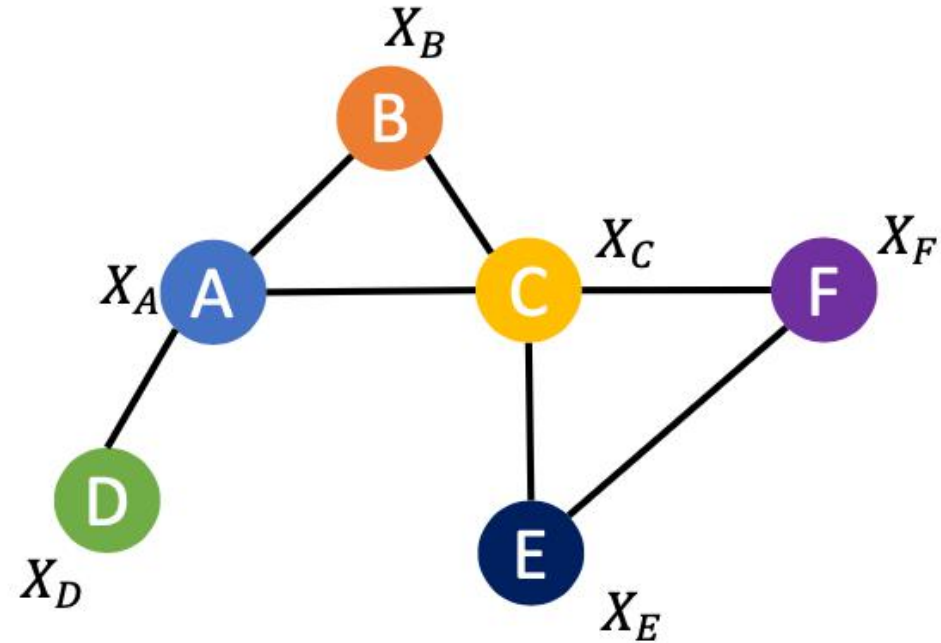
# Recipe for a good model for graphs

- Handle different types of graph prediction problems  
Requires: Representations for graphs, nodes and edges
- Handle graphs of varying sizes and structure  
Requires: A parametrization independent of graph size and structure
- Handle arbitrary node ordering  
Requires: A model invariant to node permutations
- Utilize graph structure  
Requires: A mechanism to communicate information on graphs

# What is Graph Neural Network?

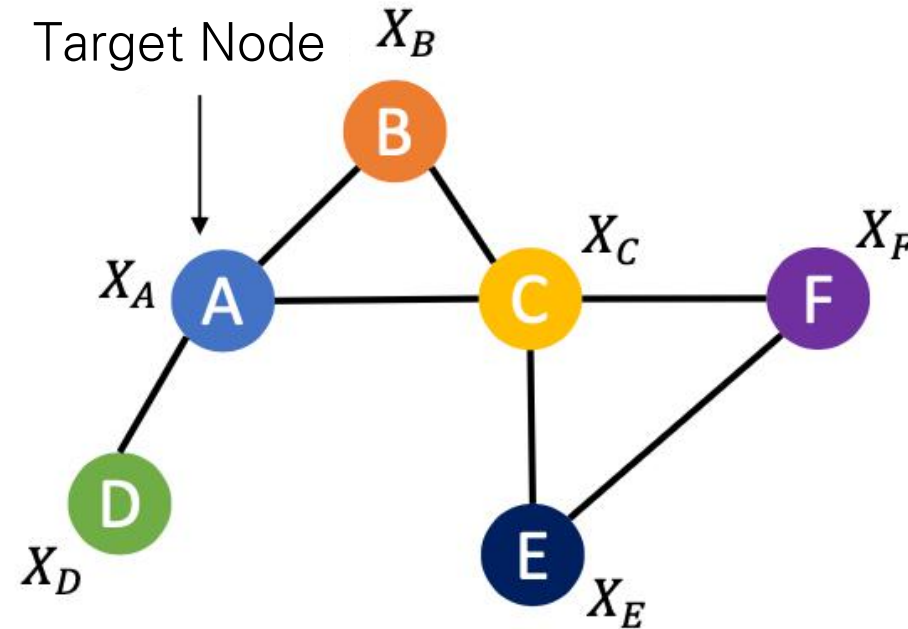
# Problem definition

- Given
  - A graph
  - Node attributes
  - (part of nodes are labeled)
- Find
  - Node embeddings
- Predict
  - Labels for the remaining nodes



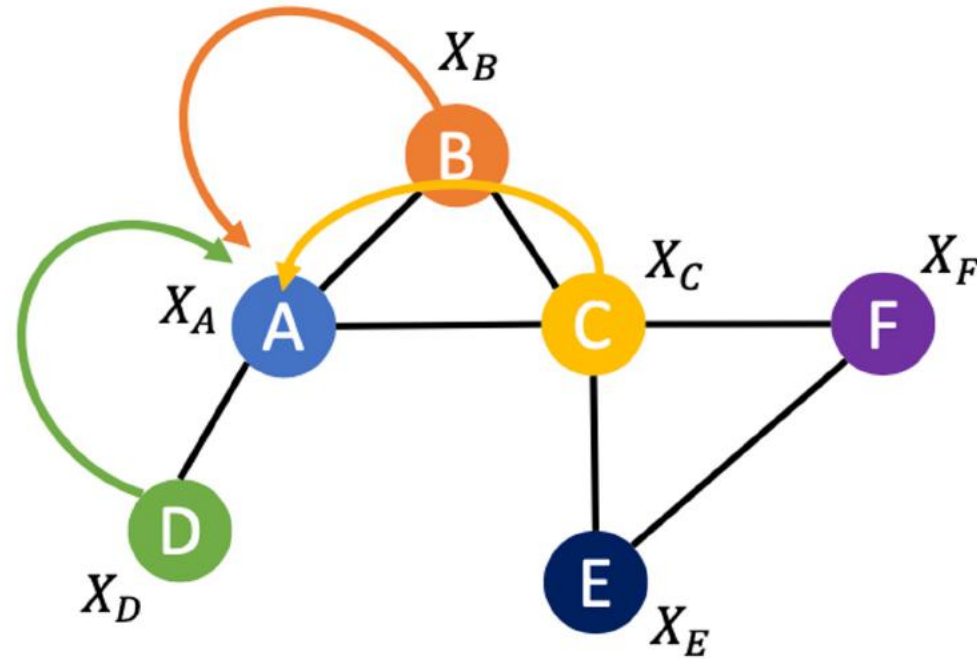


# Graph Neural Networks (GNNs)



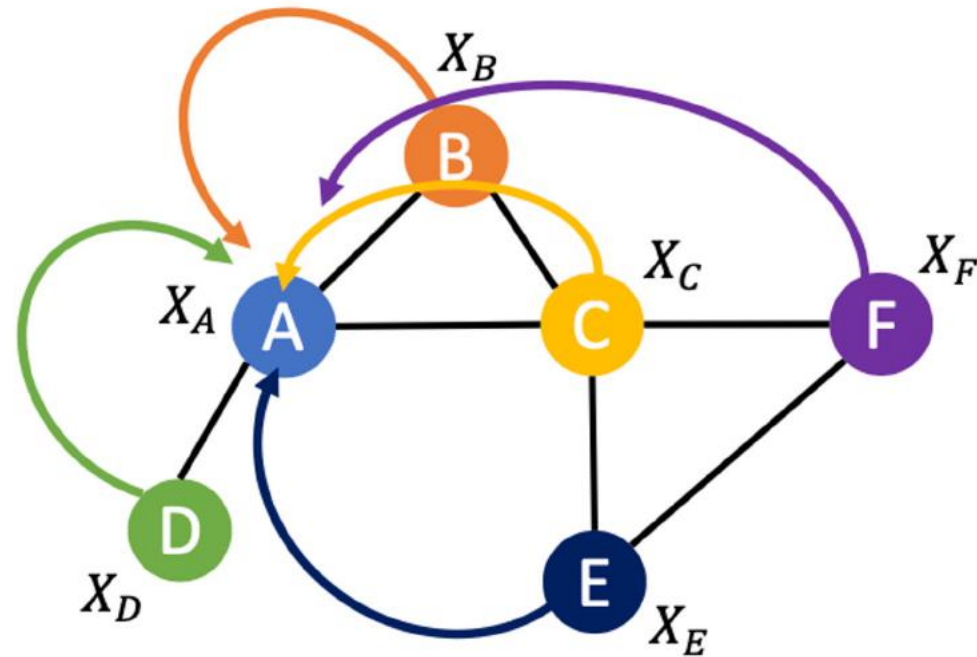
**“Homophily: connected nodes  
are  
related/informative/similar”**

# Graph Neural Networks (GNNs)



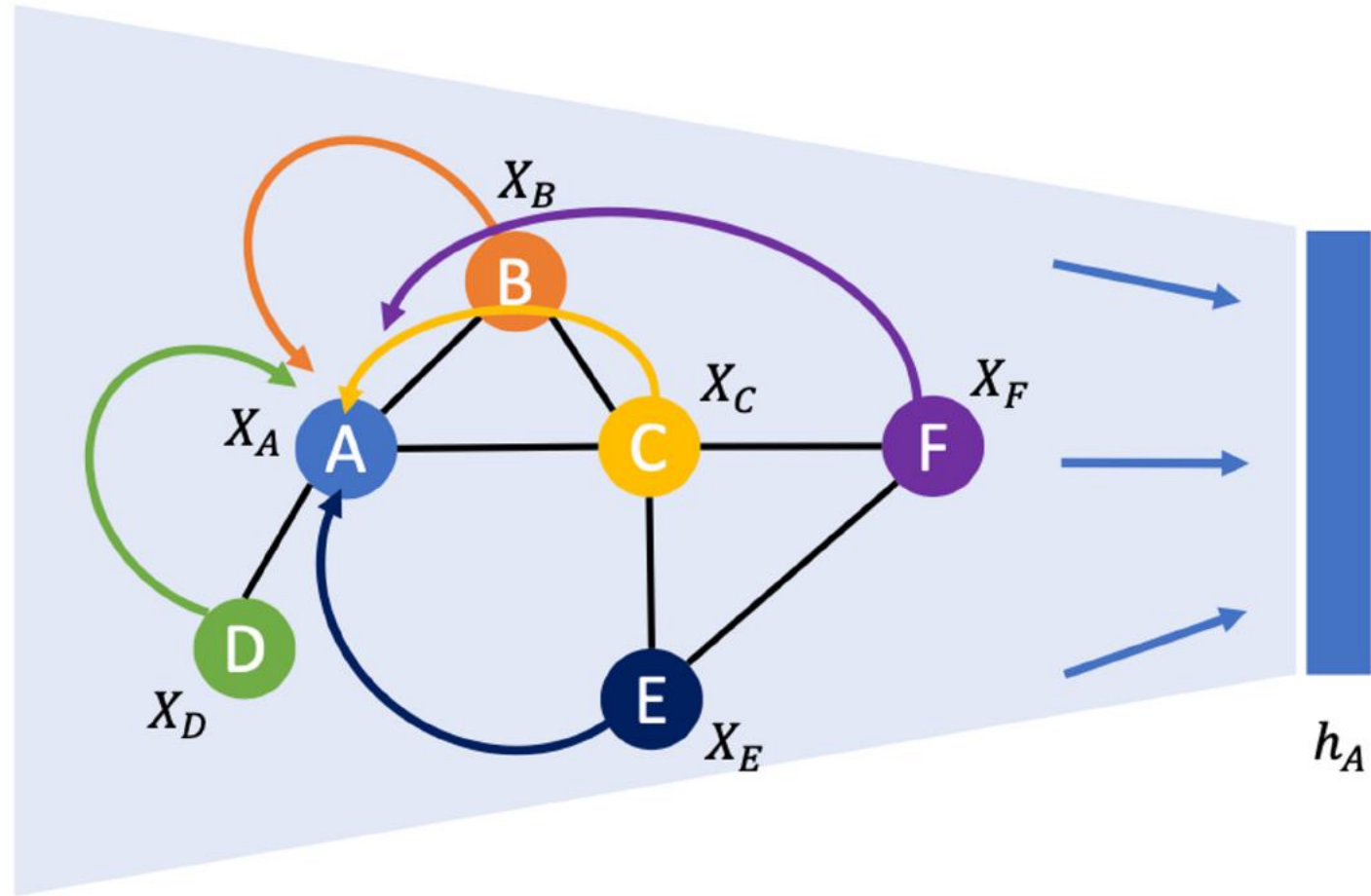
**“Homophily: connected nodes  
are  
related/informative/similar”**

# Graph Neural Networks (GNNs)

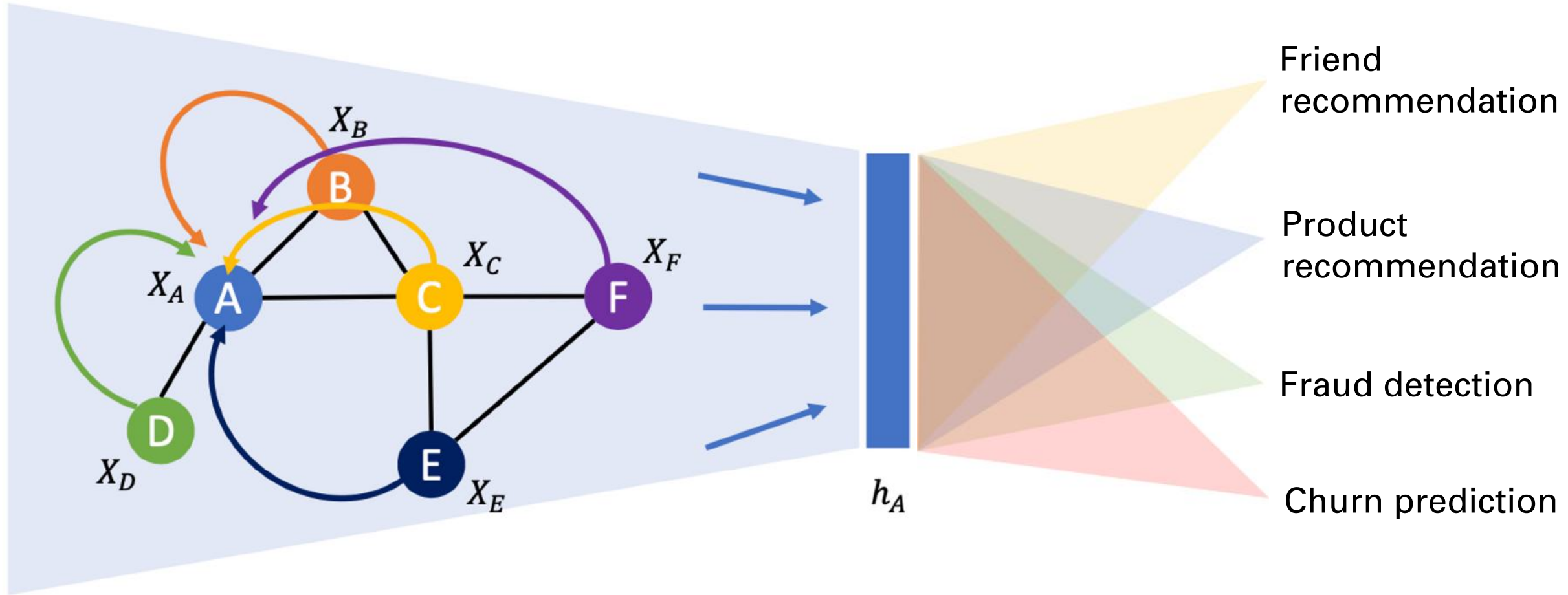


**“Homophily: connected nodes  
are  
related/informative/similar”**

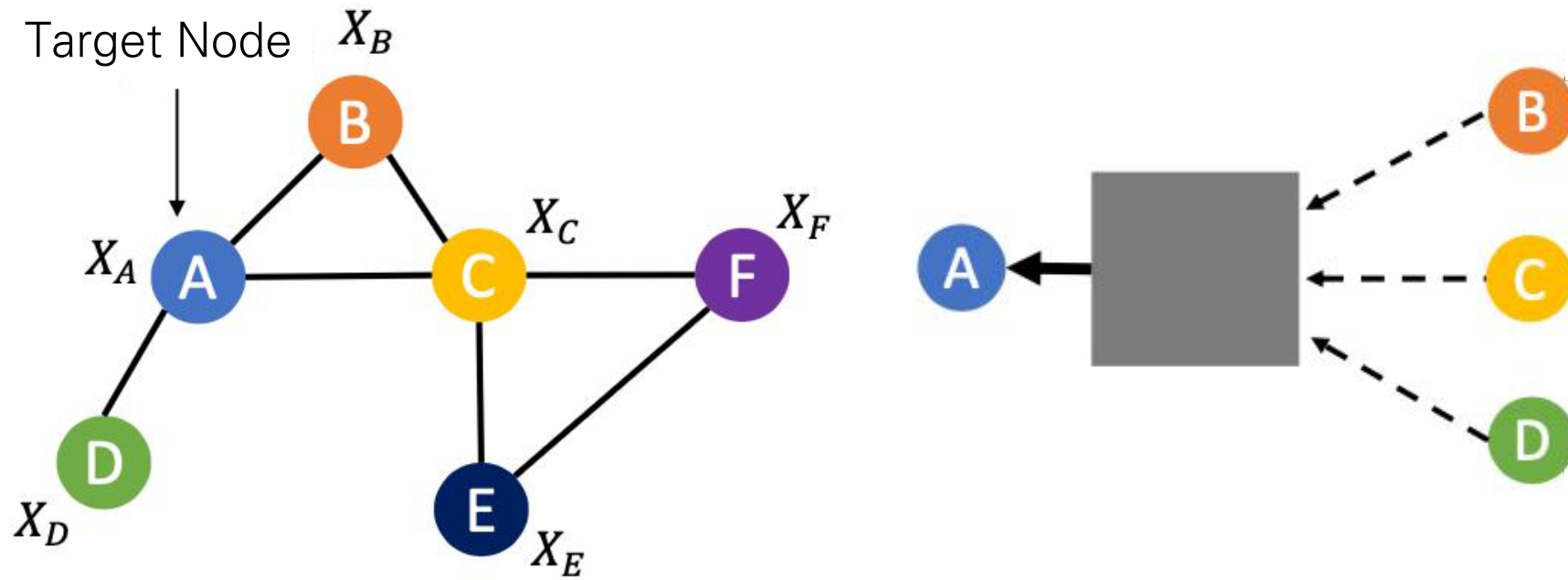
# Graph Neural Networks (GNNs)



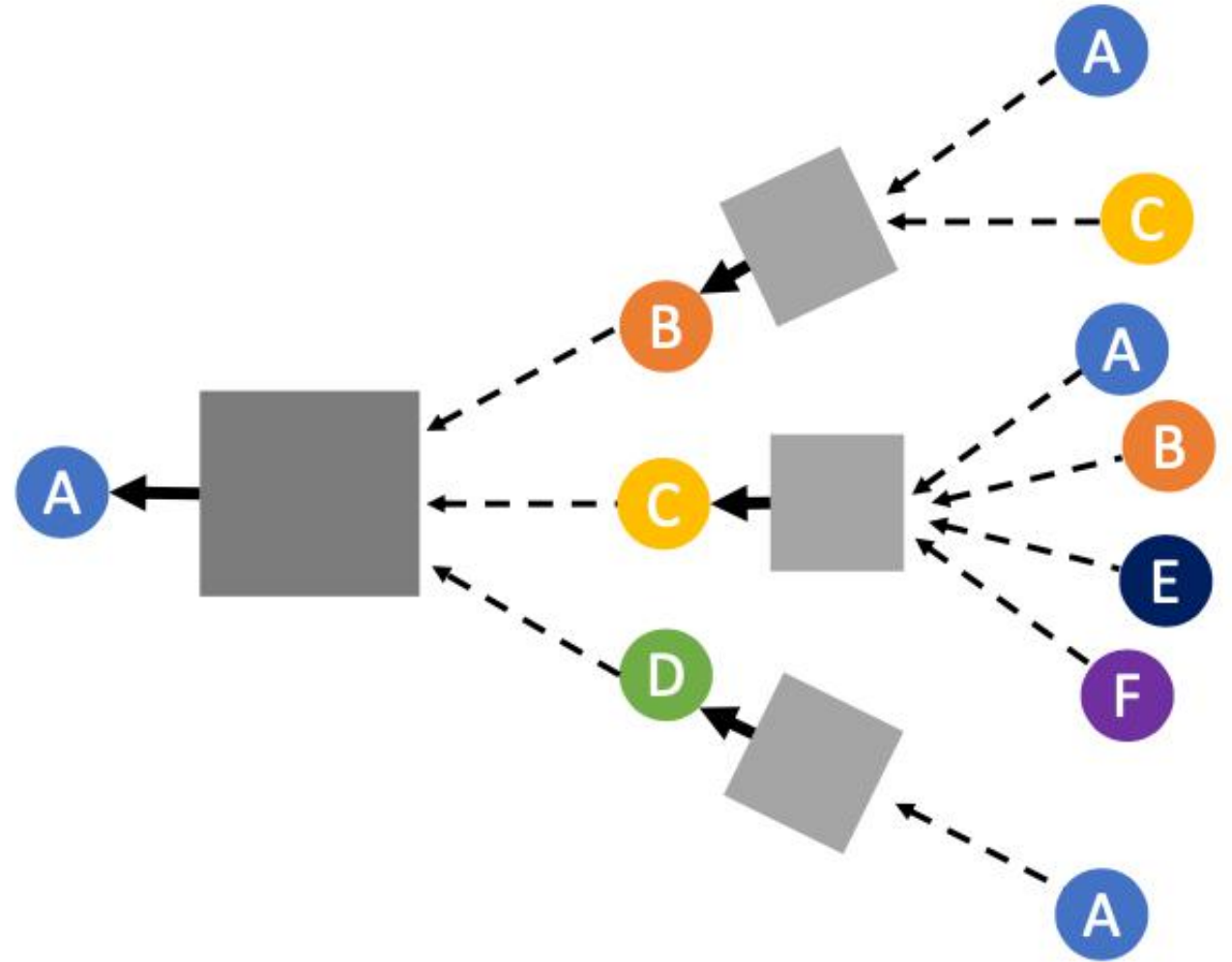
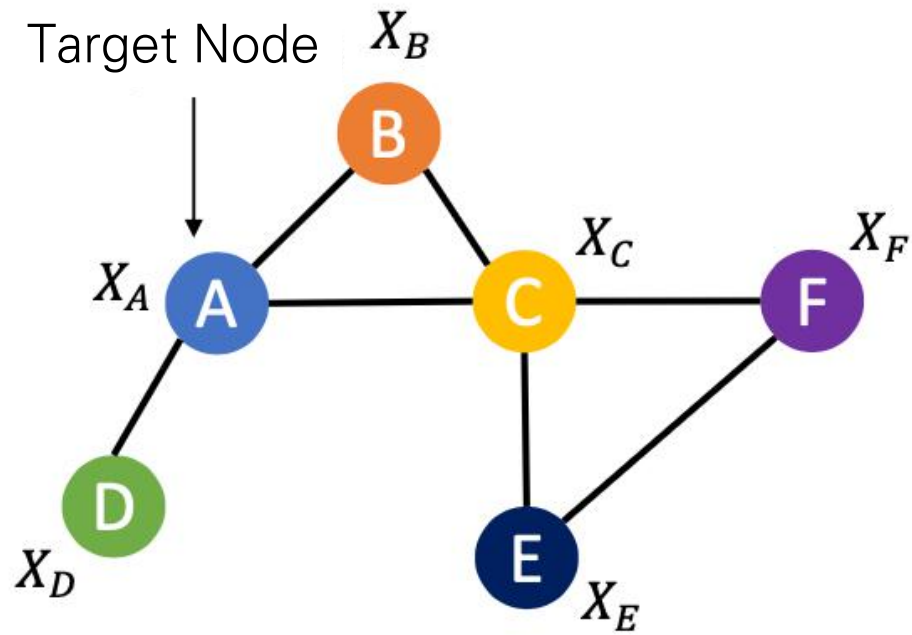
# Graph Neural Networks (GNNs)



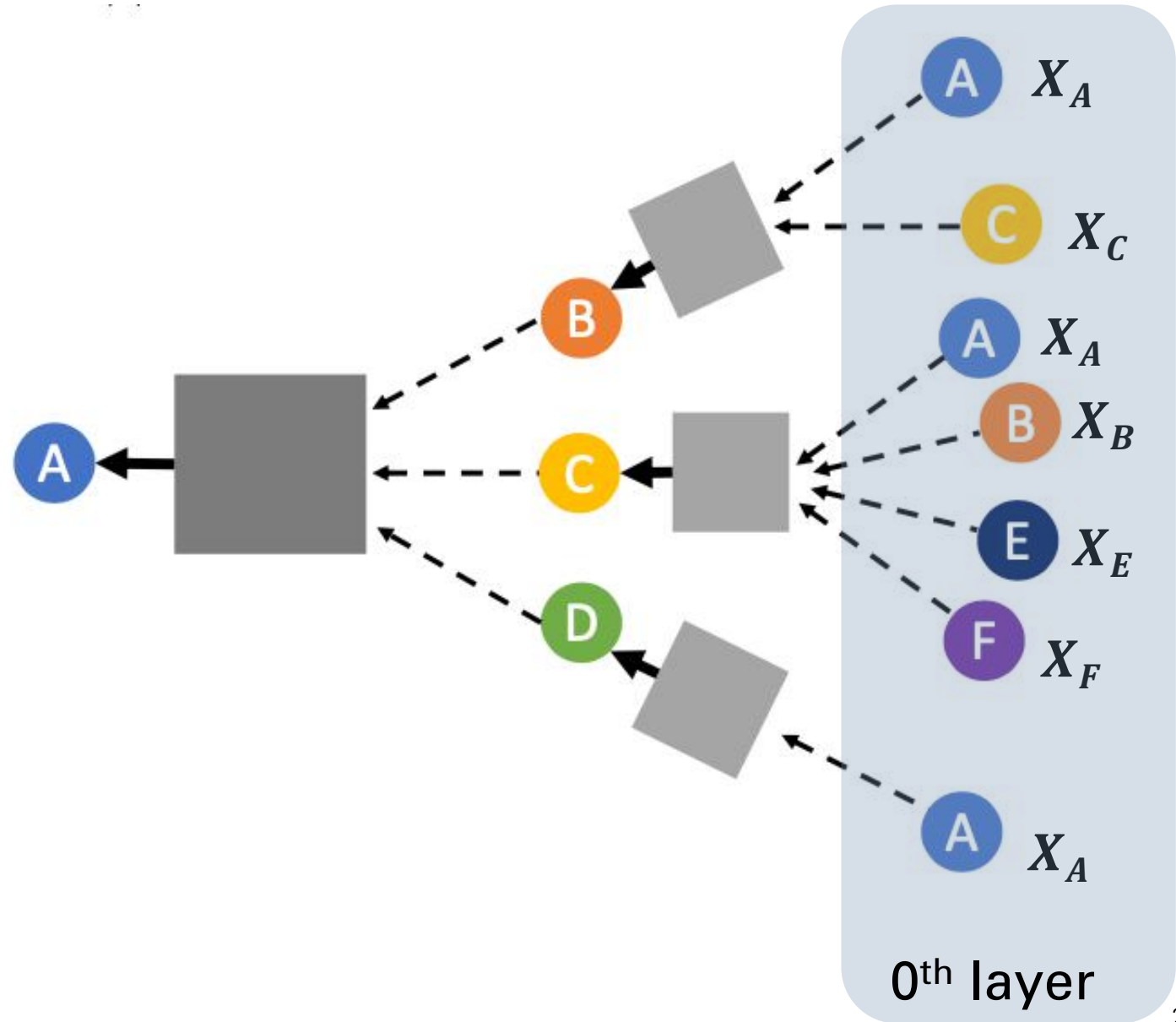
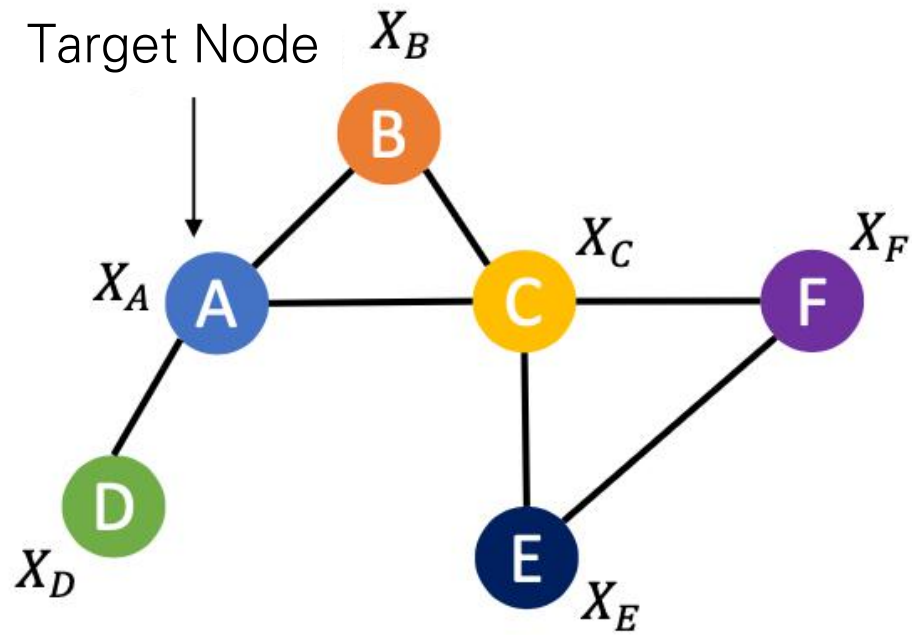
# Graph Neural Networks (GNNs)



# Graph Neural Networks (GNNs)

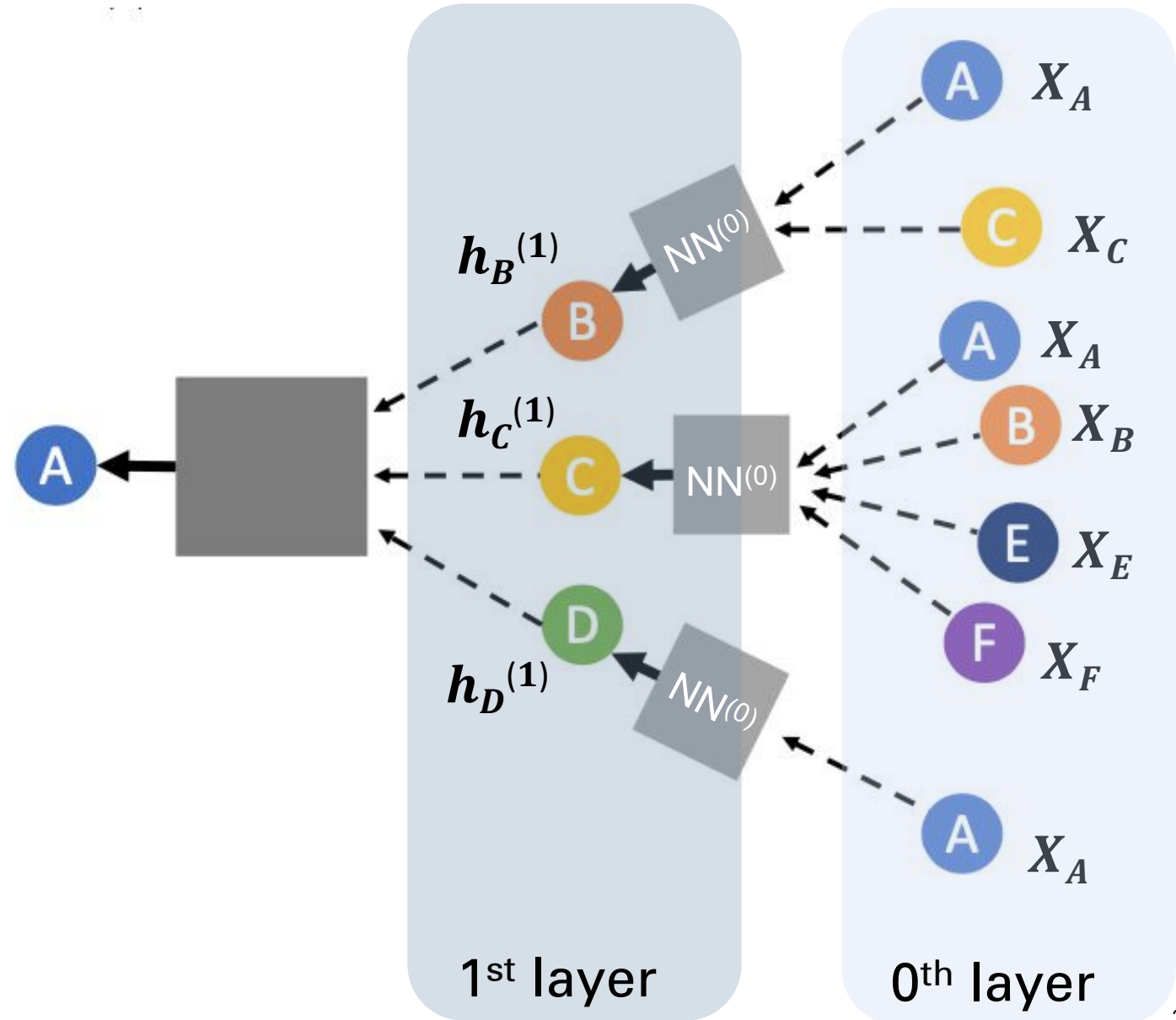
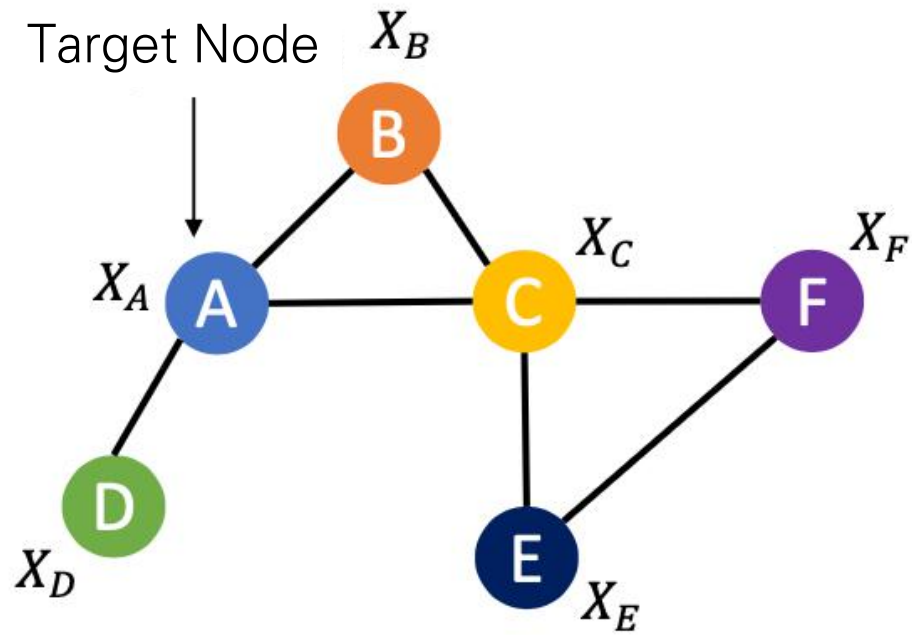


# Graph Neural Networks (GNNs)

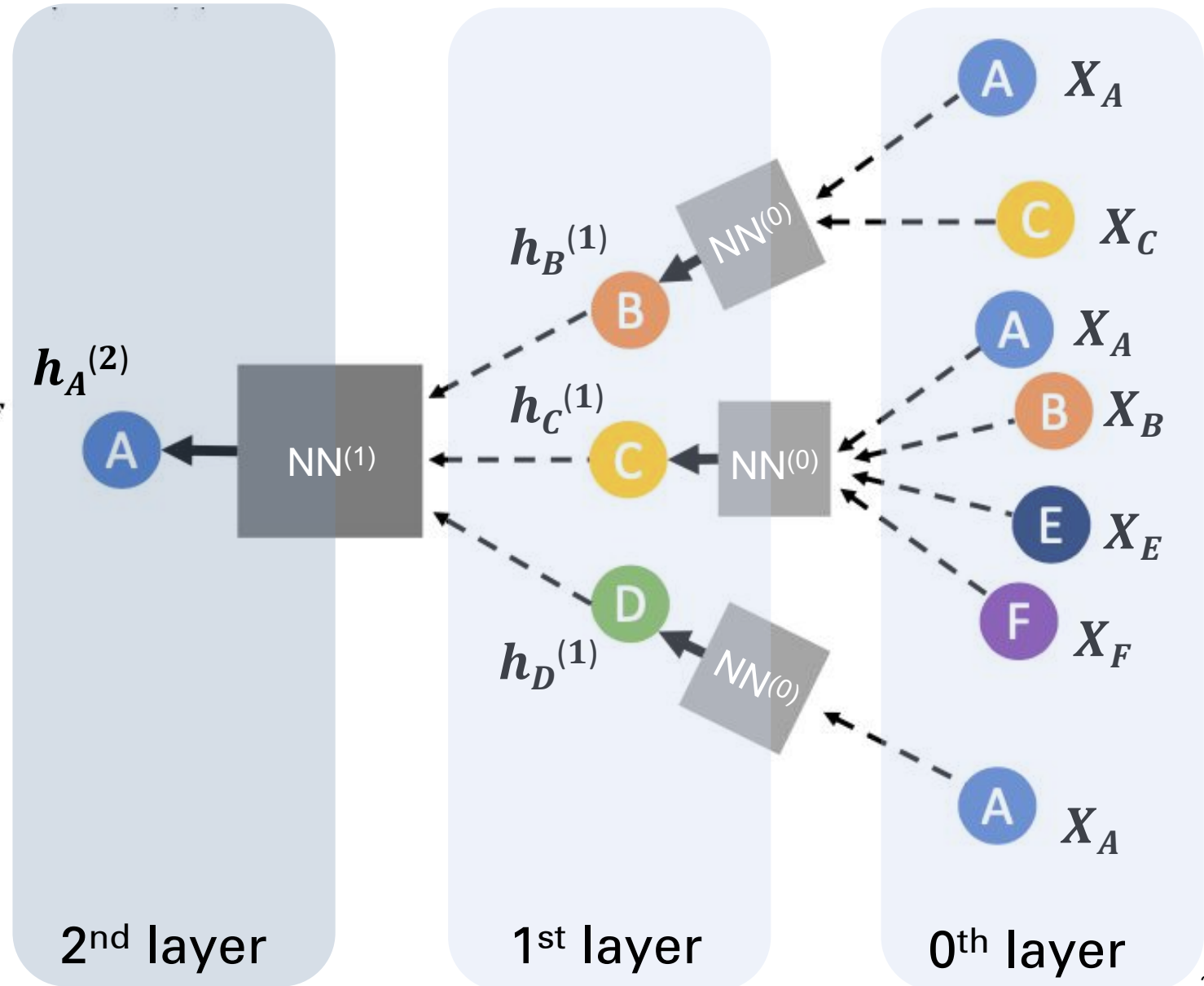
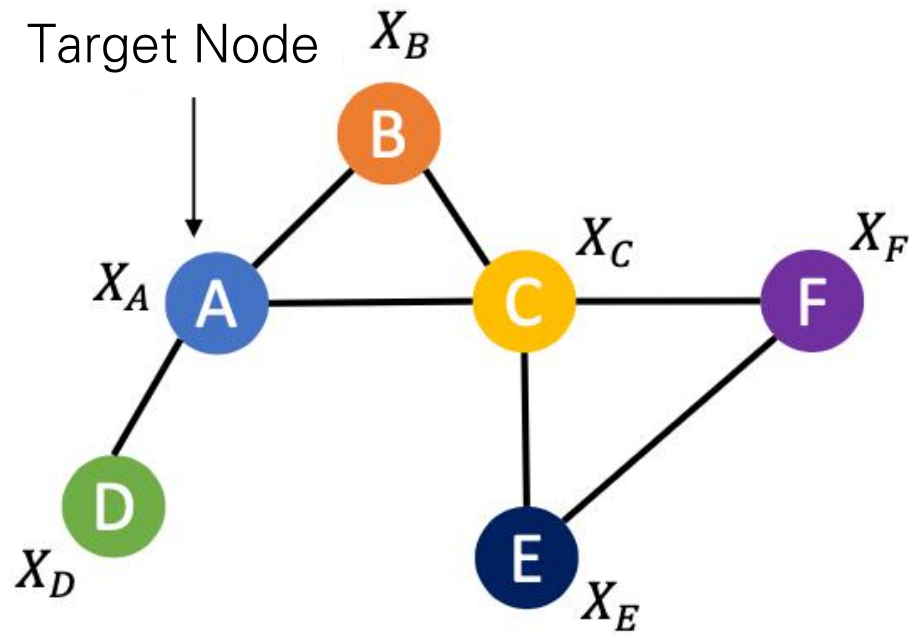




# Graph Neural Networks (GNNs)



# Graph Neural Networks (GNNs)



# Graph Neural Networks (GNNs)

## 1. Aggregate messages from neighbors

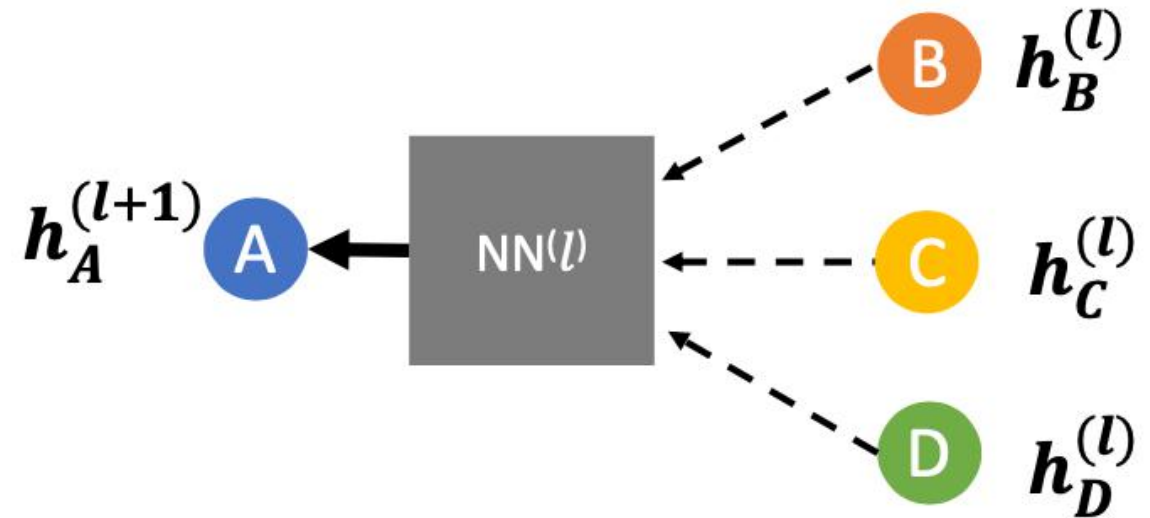
$h_v^{(l)}$ : node embedding of  $v$  at  $l$ -th layer

$\mathcal{N}(v)$ : neighboring nodes of  $v$

$f^{(l)}$ : aggregation function at  $l$ -th layer

$m_v^{(l)}$ : message vector of  $v$  at  $l$ -th layer

$$\begin{aligned} m_A^{(l)} &= f^{(l)} \left( h_A^{(l)}, \{h_u^{(l)} : u \in \mathcal{N}(A)\} \right) \\ &= f^{(l)} \left( h_A^{(l)}, h_B^{(l)} h_C^{(l)} h_D^{(l)} \right) \end{aligned}$$



Neighbors of node A

$$\mathcal{N}(A) = \{B, C, D\}$$

# Graph Neural Networks (GNNs)

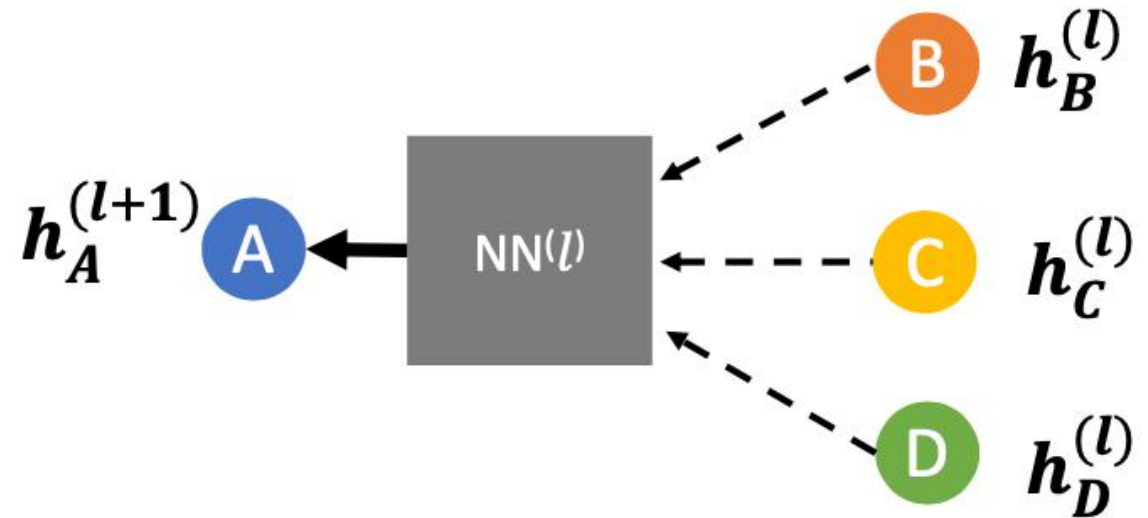
## 1. Aggregate messages from neighbors

$$\begin{aligned} m_A^{(l)} &= \mathbf{f}^{(l)} \left( h_A^{(l)}, \{h_u^{(l)} : u \in \mathcal{N}(A)\} \right) \\ &= \mathbf{f}^{(l)} \left( h_A^{(l)}, h_B^{(l)} h_C^{(l)} h_D^{(l)} \right) \end{aligned}$$

## 2. Transform messages

$\mathbf{g}^{(l)}$ : transformation function at  $l$ -th layer

$$h_A^{(l+1)} = \mathbf{g}^{(l)}(m_A^{(l)})$$



Neighbors of node A

$$\mathcal{N}(A) = \{B, C, D\}$$

# Graph Neural Networks (GNNs)

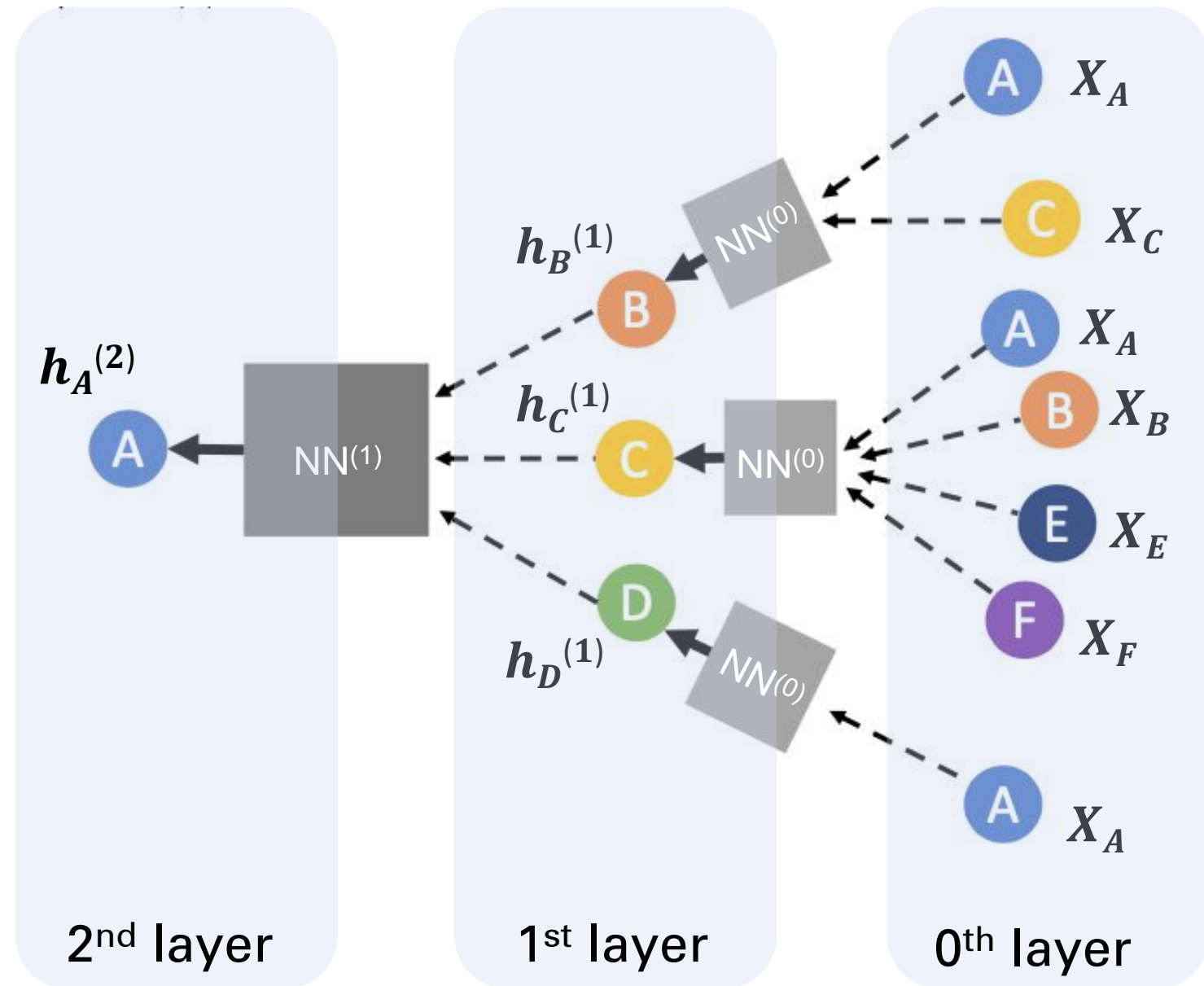
In each layer  $l$ ,  
for each target node  $v$ :

1. Aggregate messages

$$m_v^{(l)} = \mathbf{f}^{(l)} \left( h_v^{(l)}, \{h_u^{(l)} : u \in \mathcal{N}(v)\} \right)$$

2. Transform messages

$$h_v^{(l+1)} = \mathbf{g}^{(l)}(m_v^{(l)})$$



# Graph Neural Networks (GNNs)

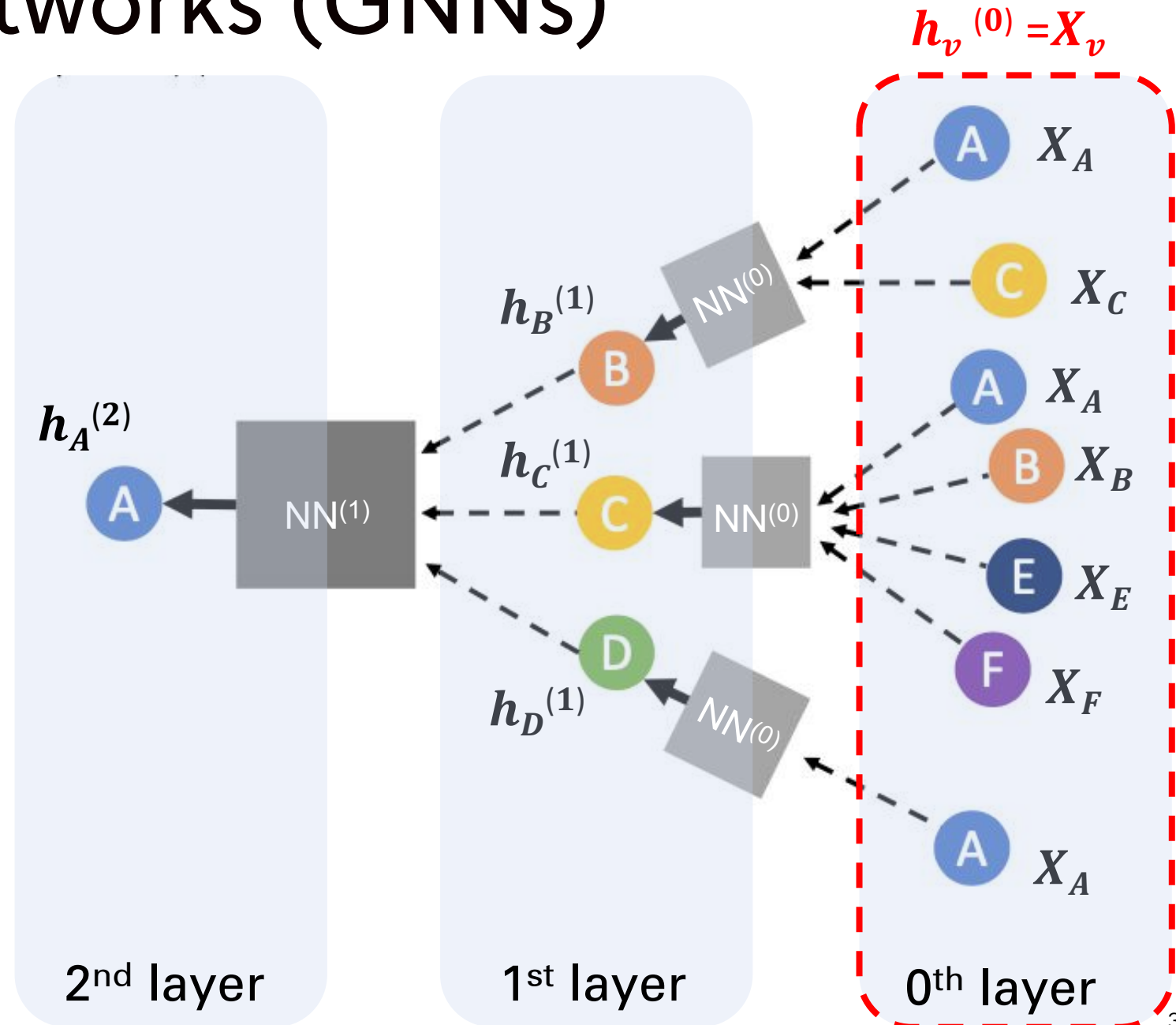
In each layer  $l$ ,  
for each target node  $v$ :

1. Aggregate messages

$$m_v^{(l)} = \mathbf{f}^{(l)} \left( h_v^{(l)}, \{h_u^{(l)} : u \in \mathcal{N}(v)\} \right)$$

2. Transform messages

$$h_v^{(l+1)} = \mathbf{g}^{(l)}(m_v^{(l)})$$



# Graph Neural Networks (GNNs)

In each layer  $l$ ,  
for each target node  $v$ :

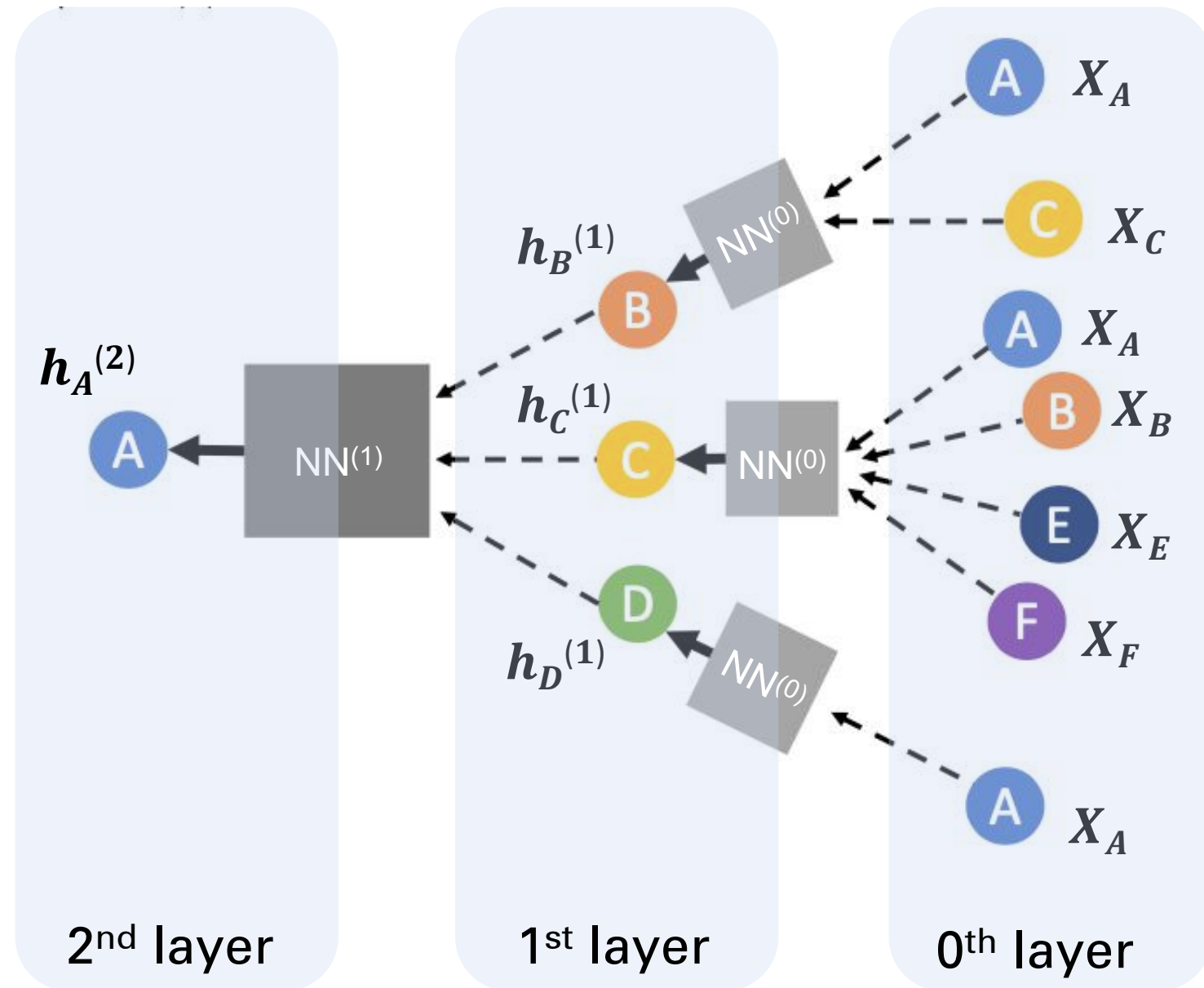
1. Aggregate messages

$$m_v^{(l)} = \mathbf{f}^{(l)}(h_v^{(l)}, \{h_u^{(l)} : u \in \mathcal{N}(v)\})$$

2. Transform messages

$$h_v^{(l+1)} = \mathbf{g}^{(l)}(m_v^{(l)})$$

GNN models mostly differ  
in how these functions are  
defined..



# Graph Neural Networks (GNNs)

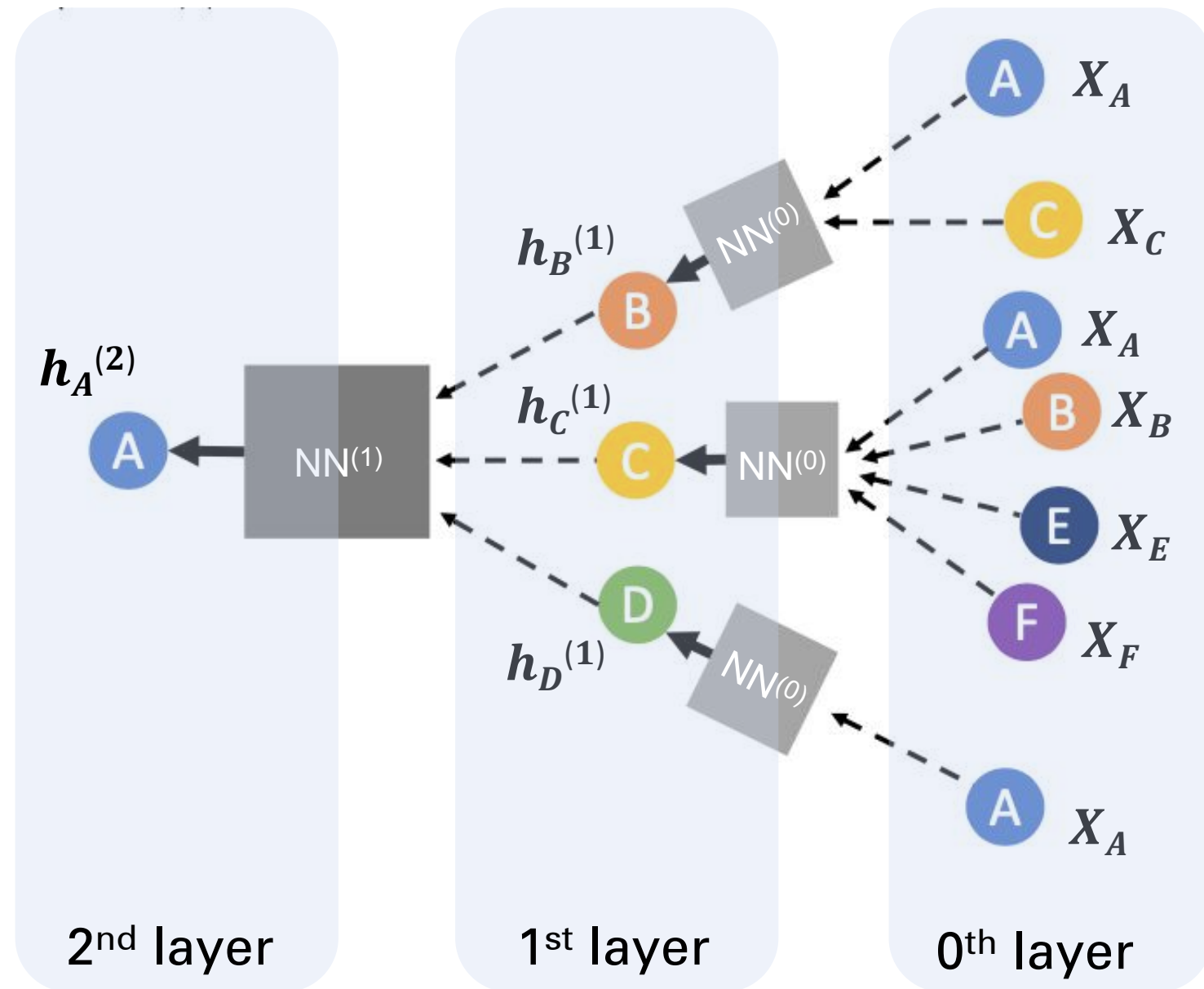
In each layer  $l$ ,  
for each target node  $v$ :

1. Aggregate messages

$$m_v^{(l)} = \mathbf{f}^{(l)} \left( h_v^{(l)}, \{h_u^{(l)} : u \in \mathcal{N}(v)\} \right)$$

2. Transform messages

$$h_v^{(l+1)} = \mathbf{g}^{(l)}(m_v^{(l)})$$





# Graph Neural Networks (GNNs)

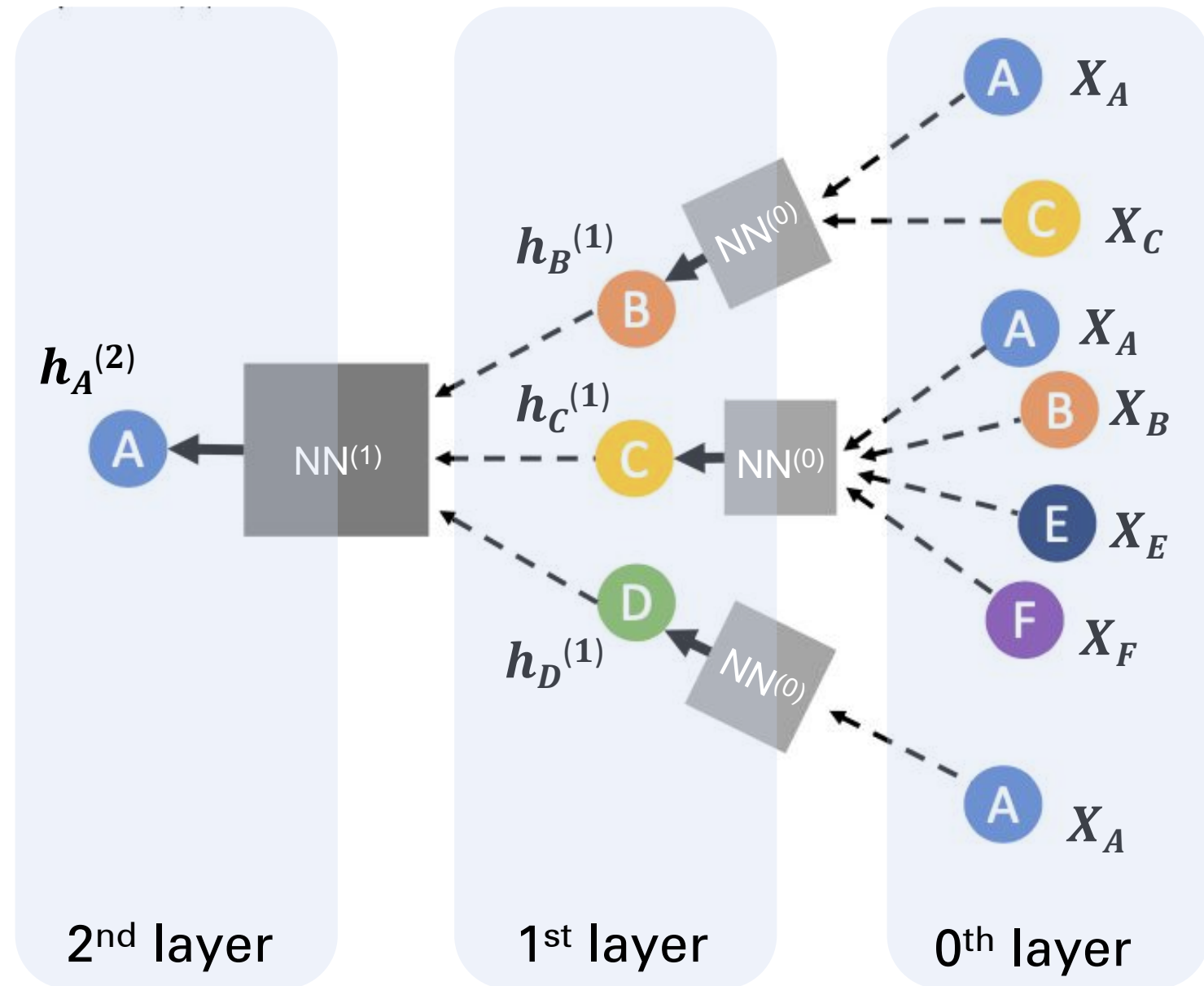
## Graph Convolutional Networks<sup>[1]</sup>

### 1. Aggregate messages

$$m_v^{(l)} = \frac{1}{|\mathcal{N}(v) + 1|} \sum_{u \in \mathcal{N}(v) \cup \{v\}} h_u^{(l)}$$

### 2. Transform messages

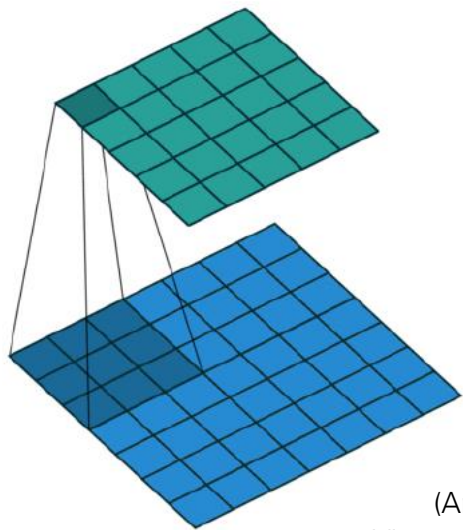
$$h_v^{(l+1)} = \sigma(\mathbf{W}^{(l)} \circ m_v^{(l)})$$



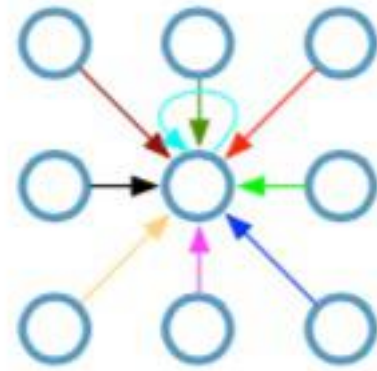
[1] Kipf, Thomas N., et al. "Semi-supervised classification with graph convolutional networks."

# Recap: Convolutional neural networks (on grids)

Single CNN layer  
with 3x3 filter:

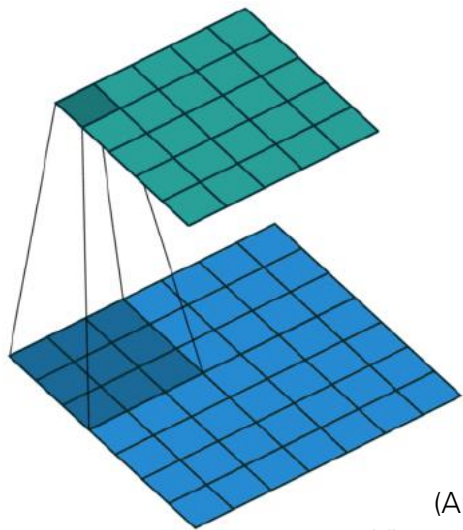


(Animation by  
Vincent Dumoulin)

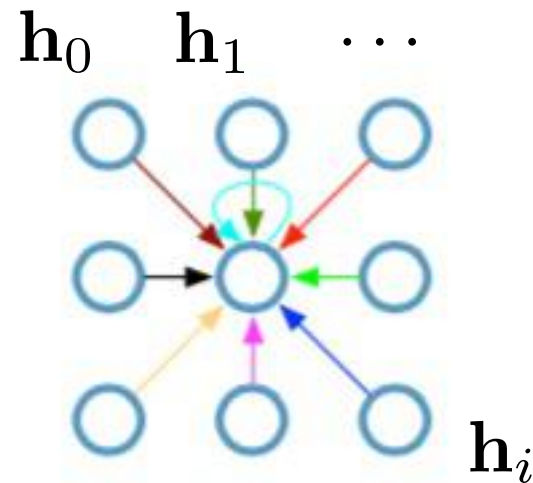


# Recap: Convolutional neural networks (on grids)

Single CNN layer  
with 3x3 filter:

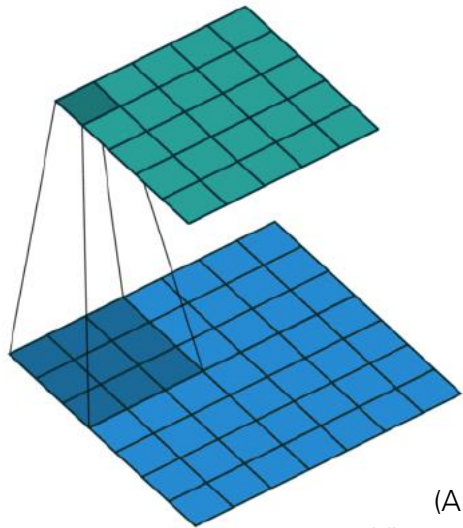


(Animation by  
Vincent Dumoulin)

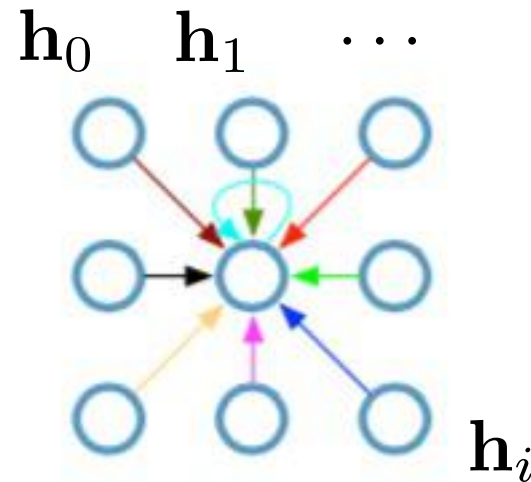


# Recap: Convolutional neural networks (on grids)

Single CNN layer  
with 3x3 filter:



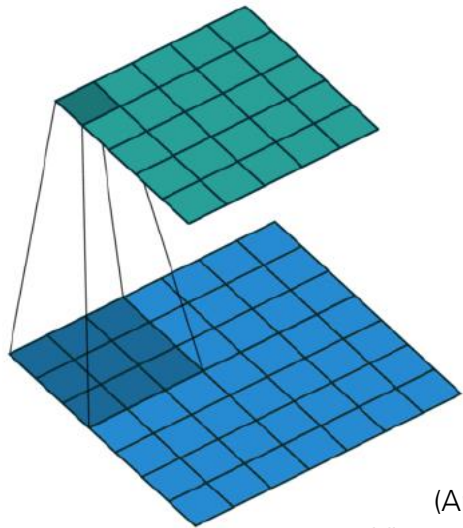
(Animation by  
Vincent Dumoulin)



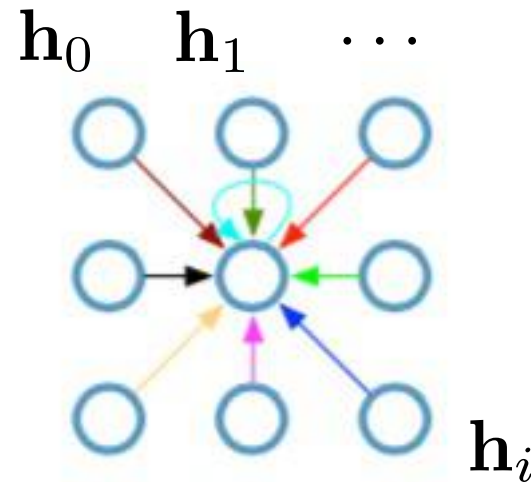
$\mathbf{h}_i \in \mathbb{R}^F$  are (hidden layer) activations of a pixel/node

# Recap: Convolutional neural networks (on grids)

Single CNN layer  
with 3x3 filter:



(Animation by  
Vincent Dumoulin)



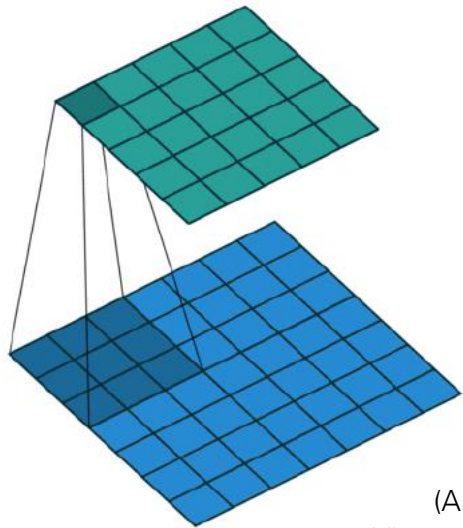
**Update for a single pixel:**

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

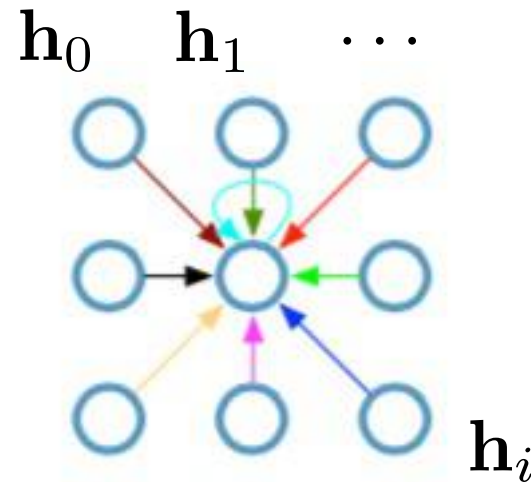
$\mathbf{h}_i \in \mathbb{R}^F$  are (hidden layer) activations of a pixel/node

# Recap: Convolutional neural networks (on grids)

Single CNN layer  
with 3x3 filter:



(Animation by  
Vincent Dumoulin)



**Update for a single pixel:**

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

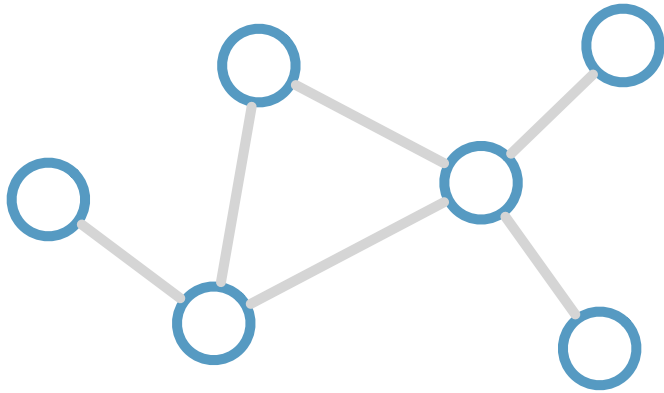
$\mathbf{h}_i \in \mathbb{R}^F$  are (hidden layer) activations of a pixel/node

**Full update:**

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

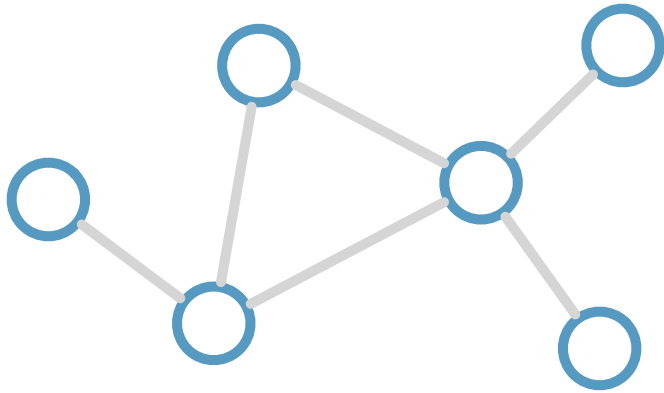
# Graph Convolutional Networks (GCNs)

Consider this  
undirected graph:

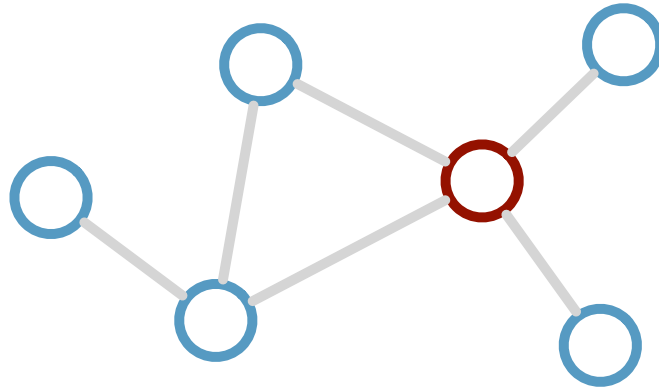


# Graph Convolutional Networks (GCNs)

Consider this undirected graph:



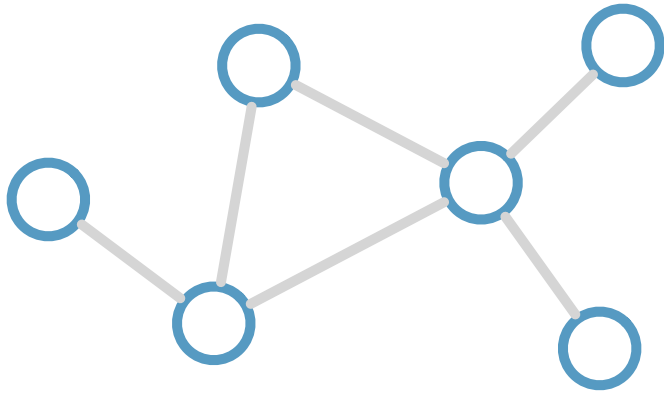
Consider update for node in red:



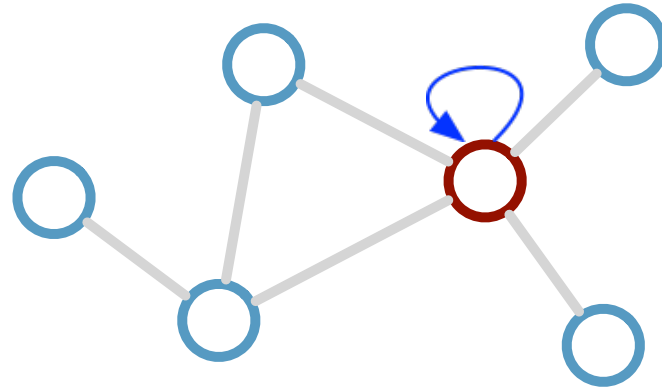


# Graph Convolutional Networks (GCNs)

Consider this undirected graph:

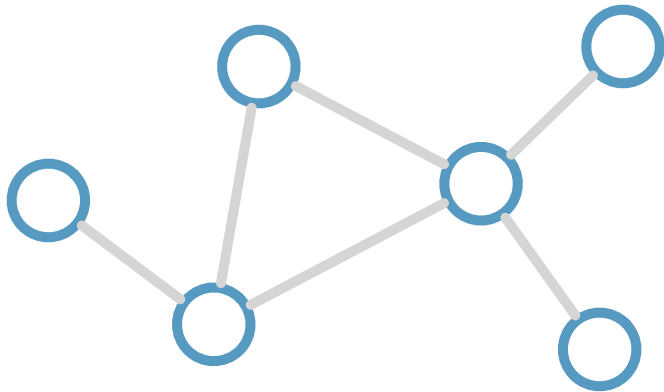


Consider update for node in red:

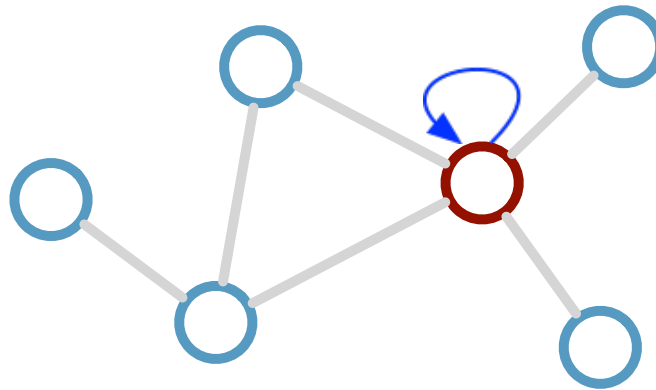


# Graph Convolutional Networks (GCNs)

Consider this undirected graph:



Consider update for node in red:



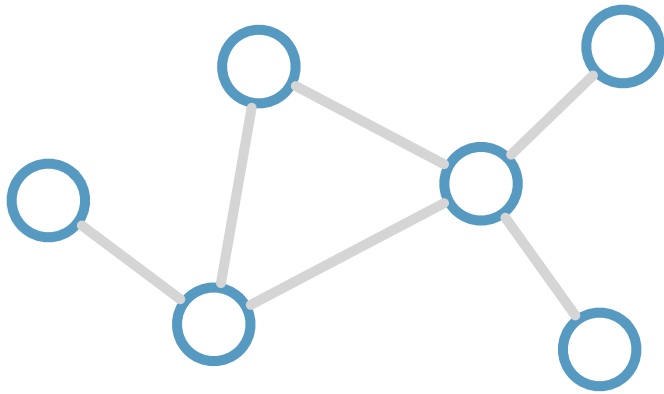
**Update rule:**

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

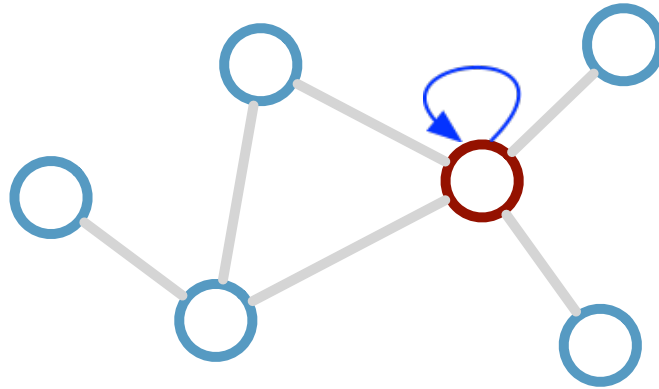
$\mathcal{N}_i$  : neighbor indices       $c_{ij}$  : norm. constant (fixed/trainable)

# Graph Convolutional Networks (GCNs)

Consider this undirected graph:



Consider update for node in red:



Desirable properties:

- Weight sharing over all locations
- Invariance to permutations
- Linear complexity  $O(E)$
- Applicable both in transductive and inductive settings

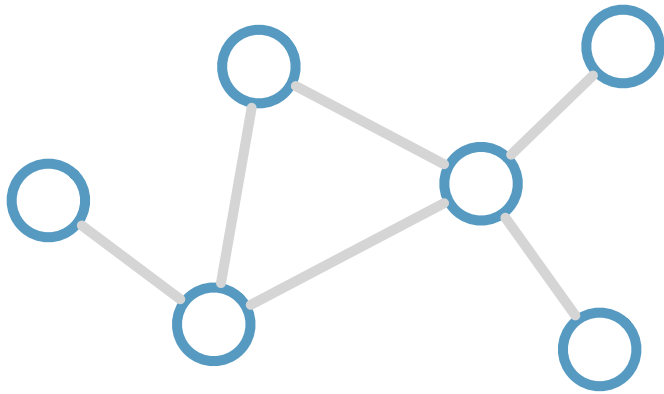
**Update rule:**

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

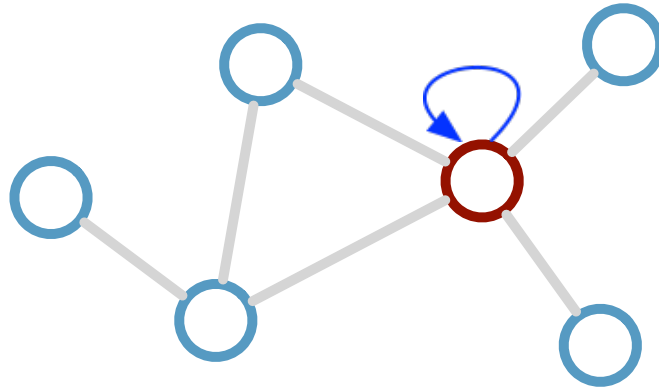
$\mathcal{N}_i$  : neighbor indices       $c_{ij}$  : norm. constant (fixed/trainable)

# Graph Convolutional Networks (GCNs)

Consider this undirected graph:



Consider update for node in red:



**Update rule:**

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

$\mathcal{N}_i$  : neighbor indices       $c_{ij}$  : norm. constant (fixed/trainable)

**Desirable properties:**

- Weight sharing over all locations
- Invariance to permutations
- Linear complexity  $O(E)$
- Applicable both in transductive and inductive settings

**Limitations:**

- Requires gating mechanism / residual connections for depth
- Only indirect support for edge features

# Graph Neural Networks (GNNs)

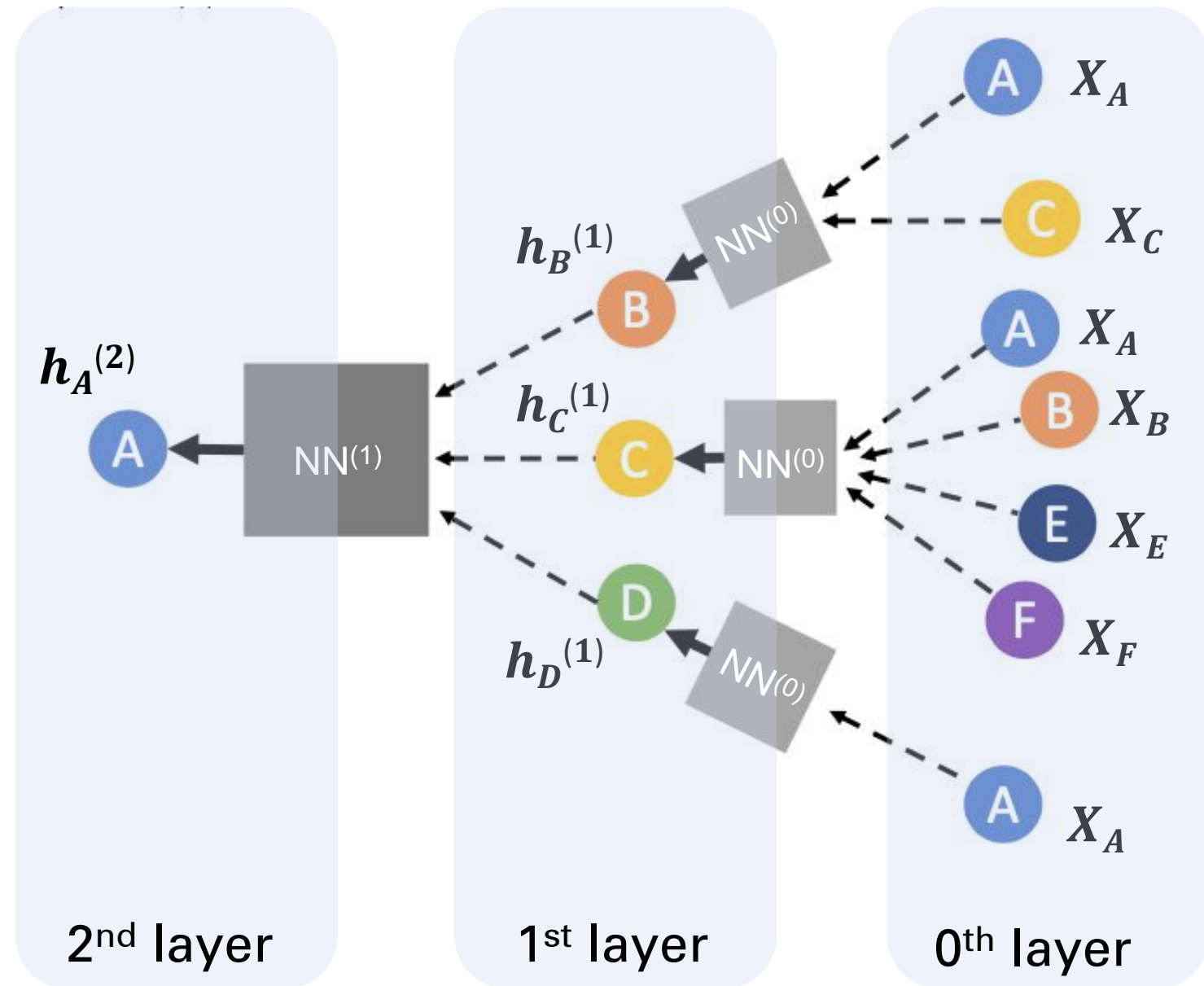
## Graph Convolutional Networks<sup>[1]</sup>

### 1. Aggregate messages

$$m_v^{(l)} = \frac{1}{|\mathcal{N}(v) + 1|} \sum_{u \in \mathcal{N}(v) \cup \{v\}} h_u^{(l)}$$

### 2. Transform messages

$$h_v^{(l+1)} = \sigma(W^{(l)} \circ m_v^{(l)})$$



[1] Kipf, Thomas N., et al. "Semi-supervised classification with graph convolutional networks."

# Graph Neural Networks (GNNs)

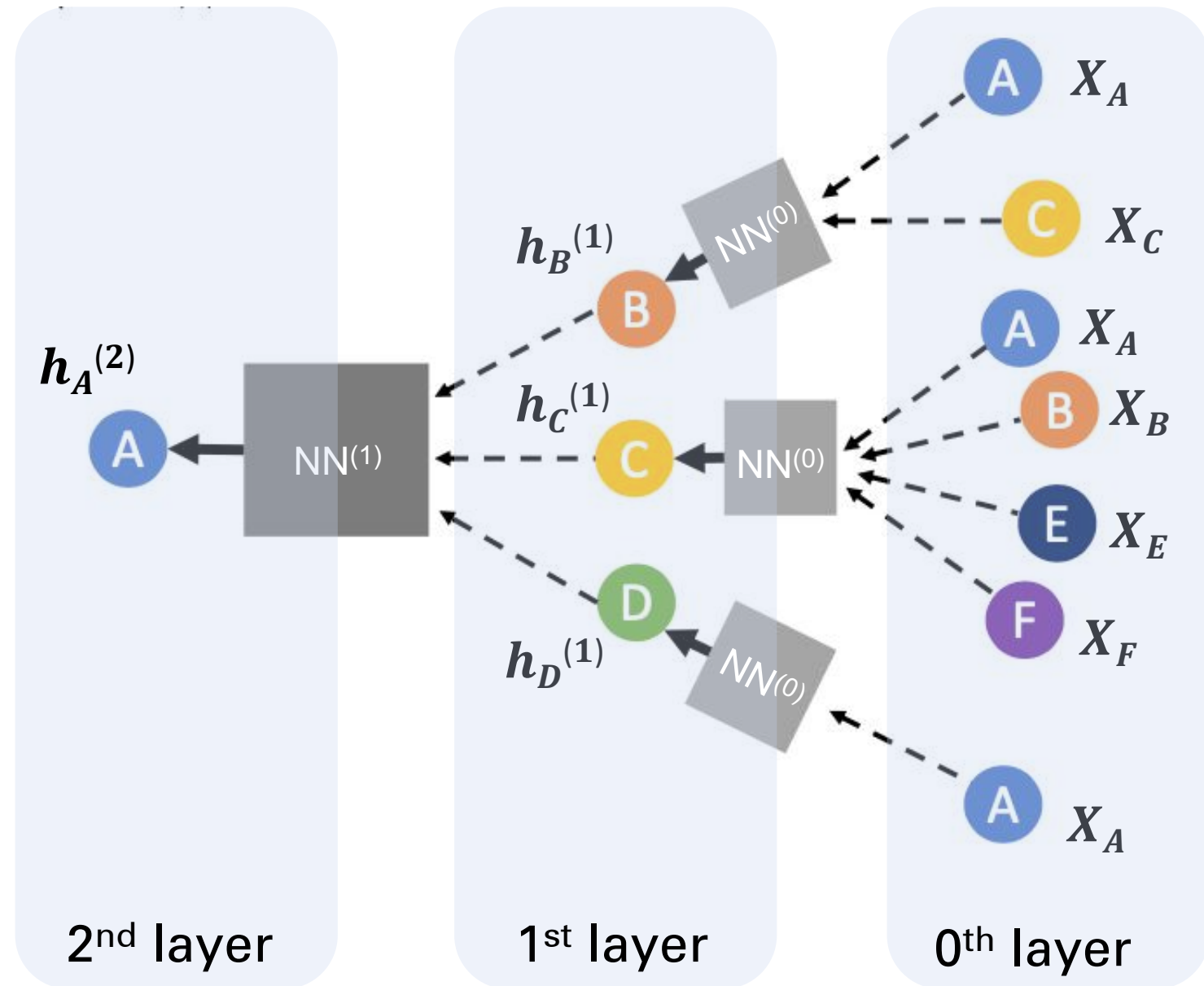
## Graph Isomorphism Networks<sup>[2]</sup>

### 1. Aggregate messages

$$m_v^{(l)} = \sum_{u \in \mathcal{N}(v) \cup \{v\}} h_u^{(l)}$$

### 2. Transform messages

$$h_v^{(l+1)} = \sigma(W^{(l)} \circ m_v^{(l)})$$



[2] Xu, Keyulu, et al. "How powerful are graph neural networks?."

# Graph Neural Networks (GNNs)

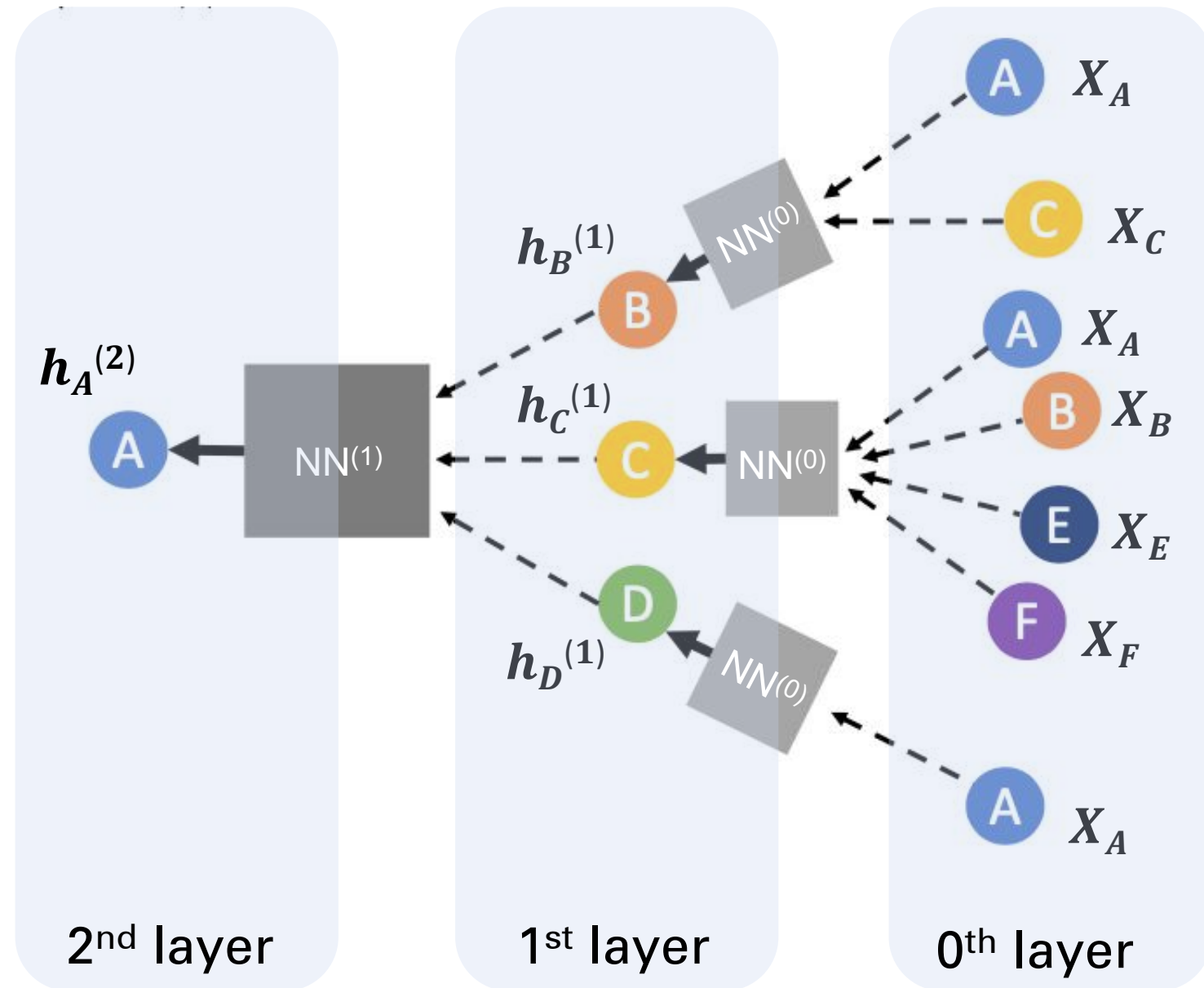
## Simplified Graph Convolutional Networks<sup>[2]</sup>

### 1. Aggregate messages

$$m_v^{(l)} = \frac{1}{|\mathcal{N}(v) + 1|} \sum_{u \in \mathcal{N}(v) \cup \{v\}} h_u^{(l)}$$

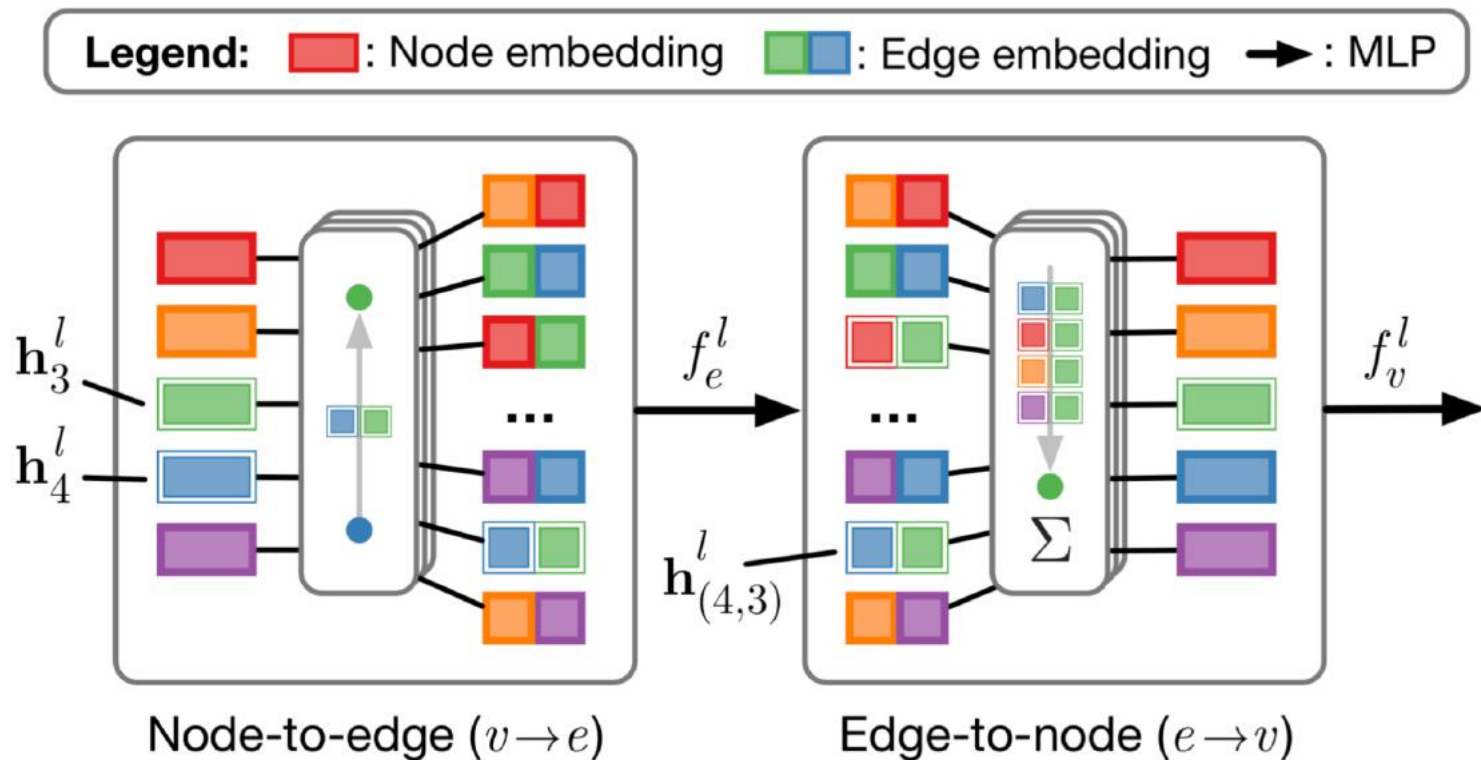
### 2. Transform messages

$$h_v^{(l+1)} = \mathbf{W}^{(l)} \circ m_v^{(l)}$$



[3] Wu, Felix, et al. "Simplifying graph convolutional networks."

# GCNs with edge embeddings

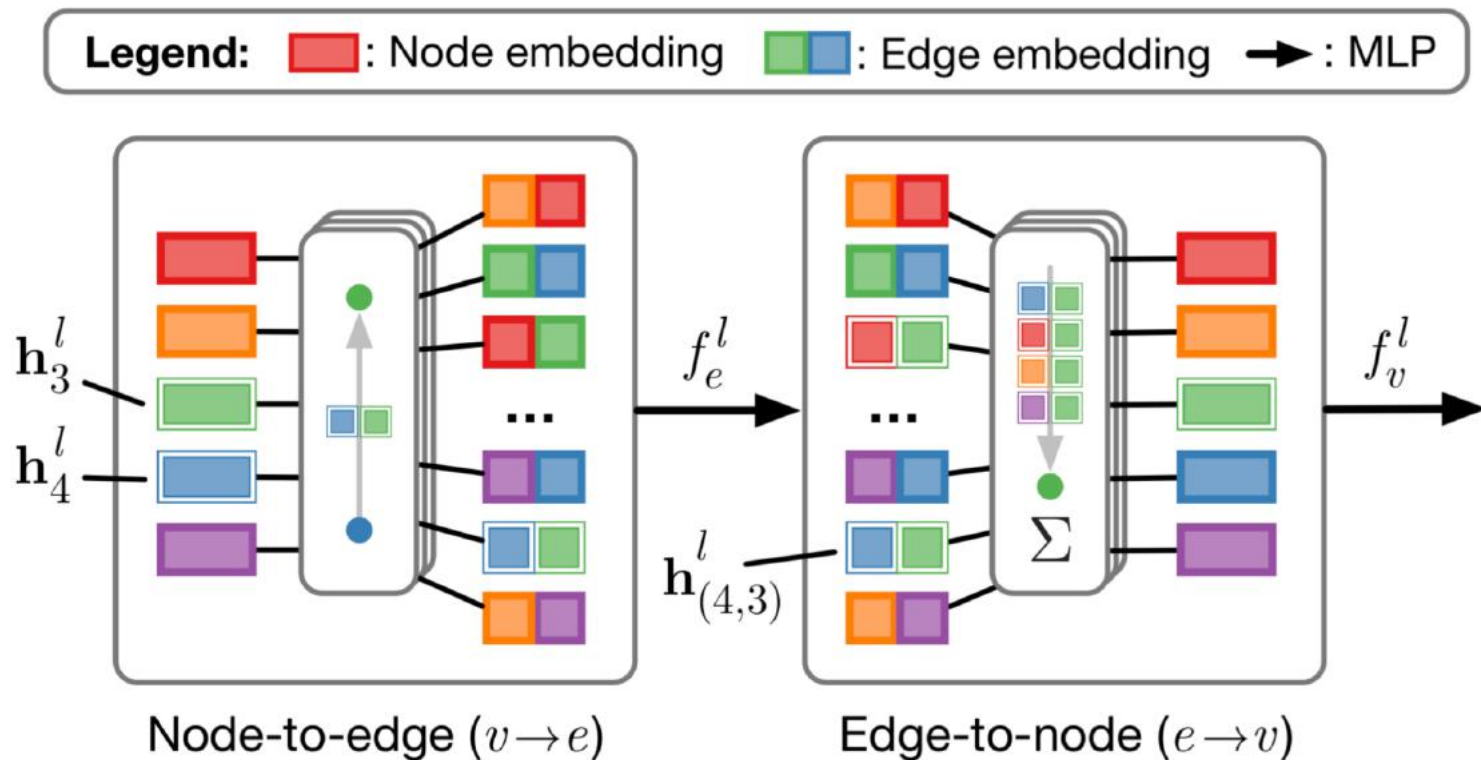


**Formally:**  $v \rightarrow e : \mathbf{h}_{(i,j)}^l = f_e^l \left( [\mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{x}_{(i,j)}] \right)$

$$e \rightarrow v : \mathbf{h}_j^{l+1} = f_v^l \left( \left[ \sum_{i \in \mathcal{N}_j} \mathbf{h}_{(i,j)}^l, \mathbf{x}_j \right] \right)$$



# GCNs with edge embeddings



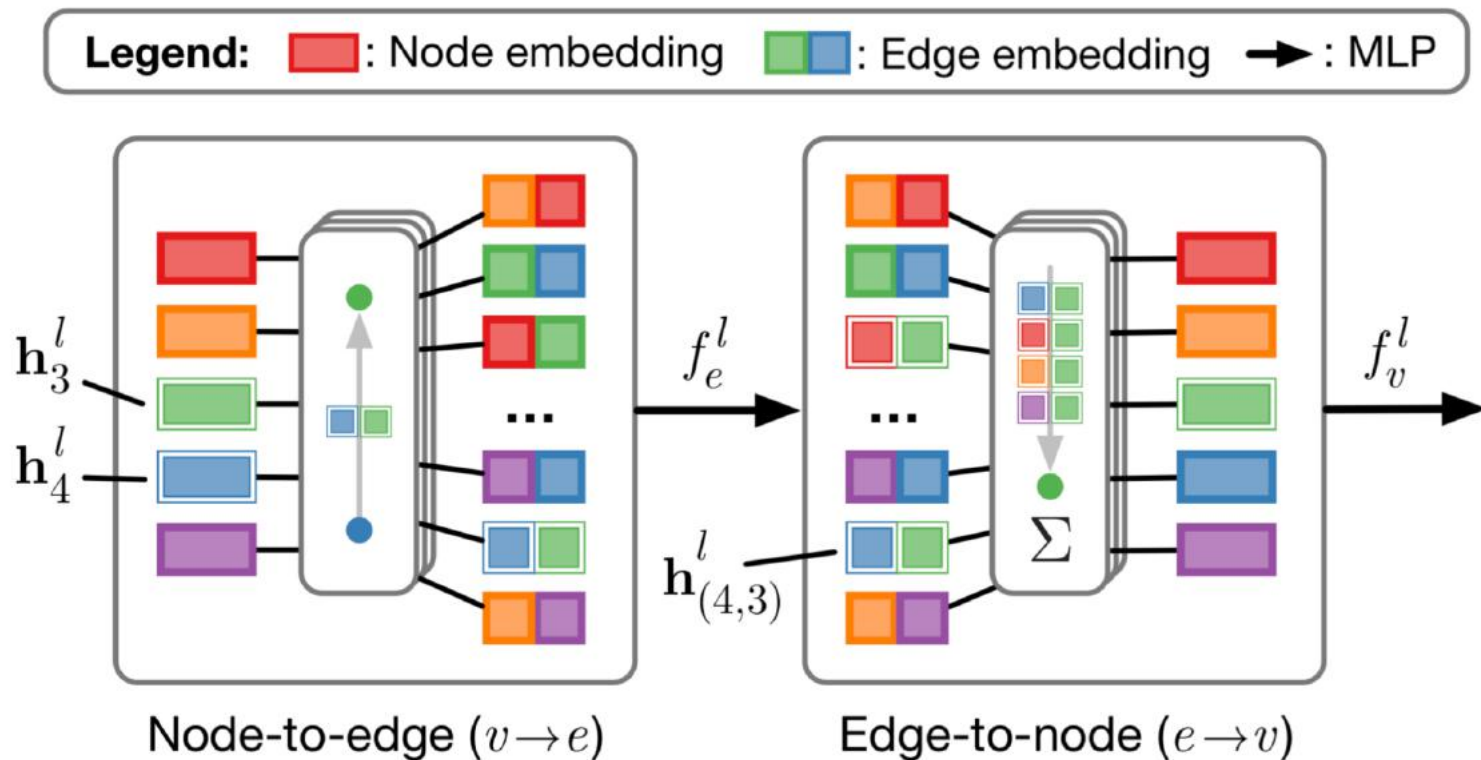
Pros:

- Supports edge features
- More expressive than GCN
- As general as it gets (?)
- Supports sparse matrix ops

**Formally:**  $v \rightarrow e : \mathbf{h}_{(i,j)}^l = f_e^l \left( \left[ \mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{x}_{(i,j)} \right] \right)$

$$e \rightarrow v : \mathbf{h}_j^{l+1} = f_v^l \left( \left[ \sum_{i \in \mathcal{N}_j} \mathbf{h}_{(i,j)}^l, \mathbf{x}_j \right] \right)$$

# GCNs with edge embeddings



## Pros:

- Supports edge features
- More expressive than GCN
- As general as it gets (?)
- Supports sparse matrix ops

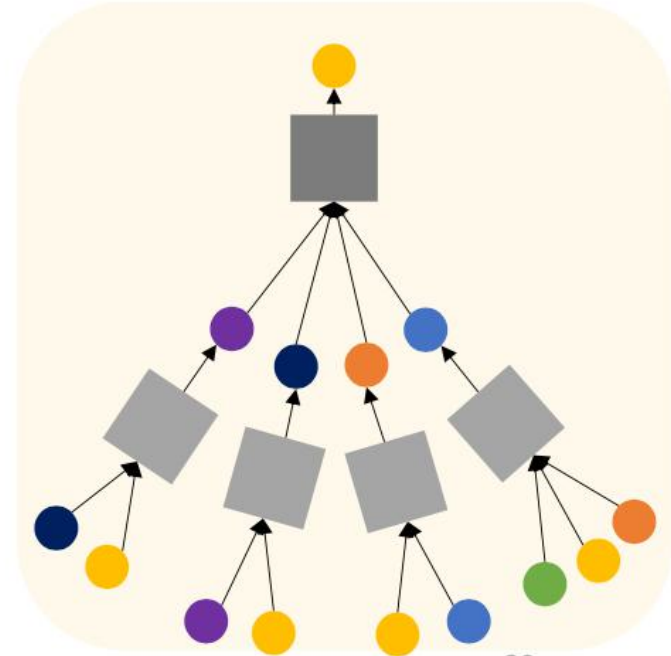
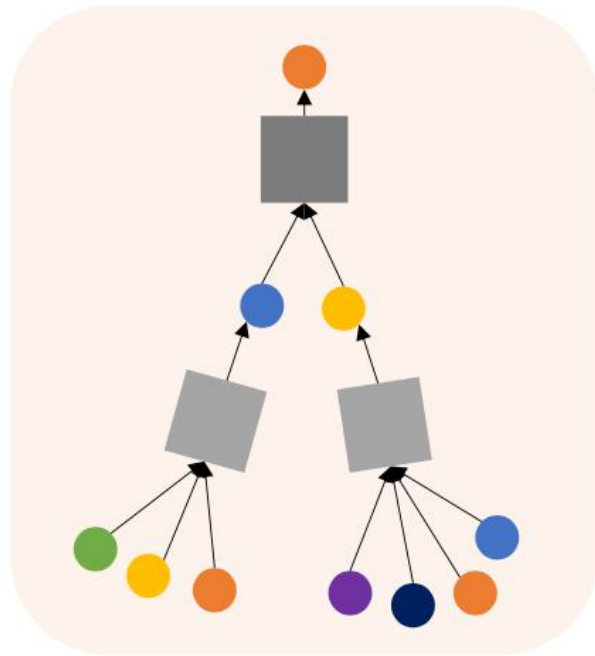
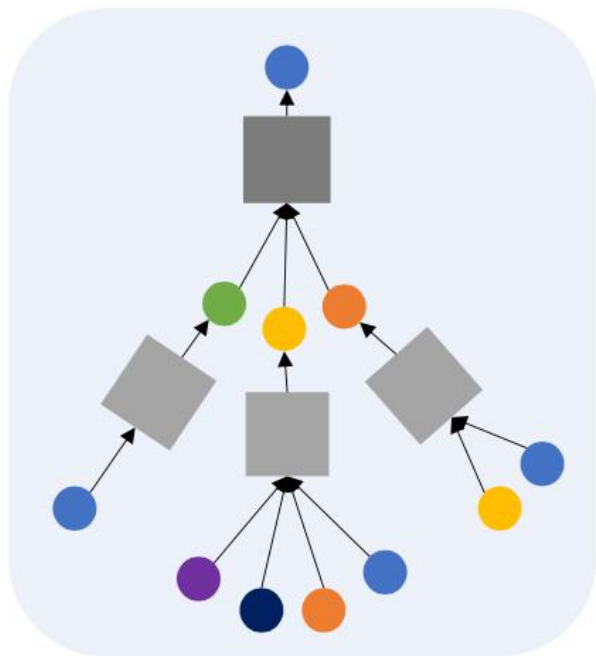
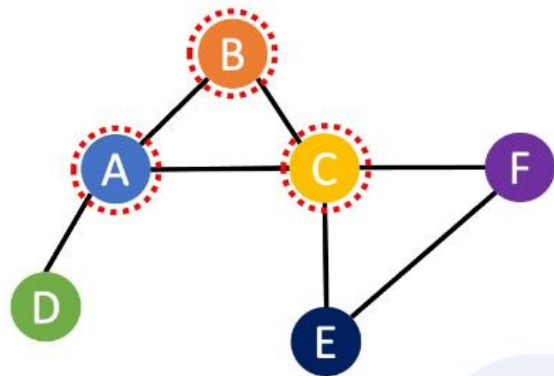
## Cons:

- Need to store intermediate edge-based activations
- Difficult to implement with subsampling
- In practice limited to small graphs

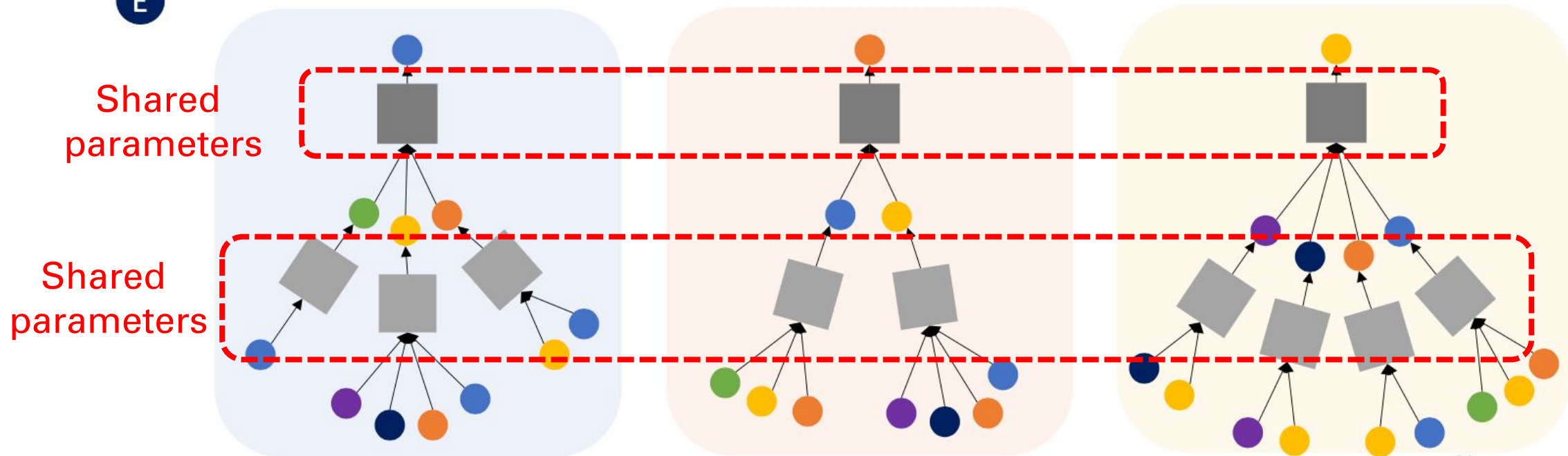
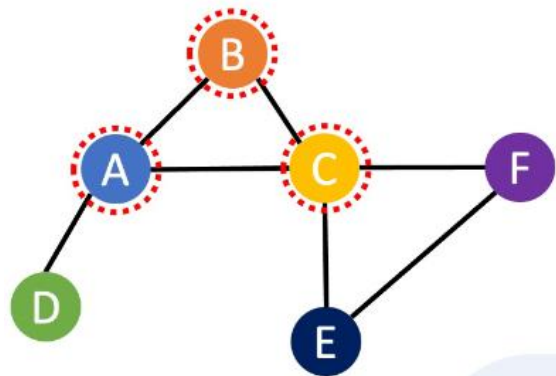
**Formally:**  $v \rightarrow e : \mathbf{h}_{(i,j)}^l = f_e^l \left( \left[ \mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{x}_{(i,j)} \right] \right)$

$$e \rightarrow v : \mathbf{h}_j^{l+1} = f_v^l \left( \left[ \sum_{i \in \mathcal{N}_j} \mathbf{h}_{(i,j)}^l, \mathbf{x}_j \right] \right)$$

# Computation graphs

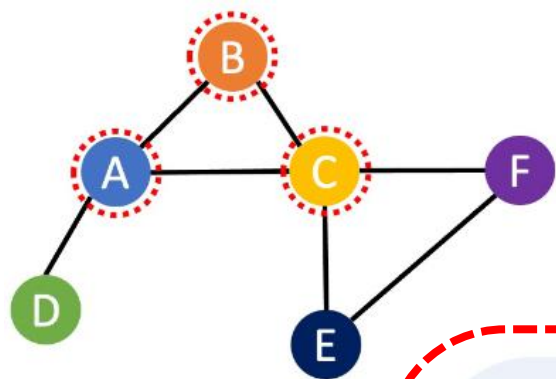


# Computation graphs

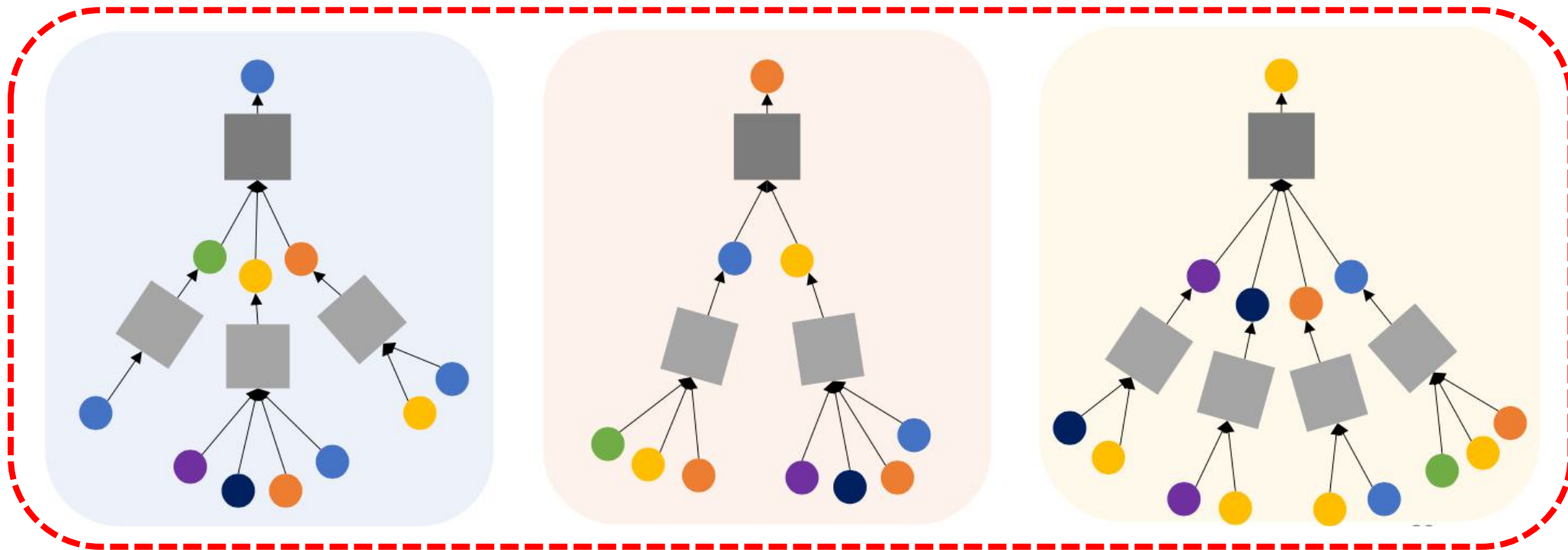


# Batch execution

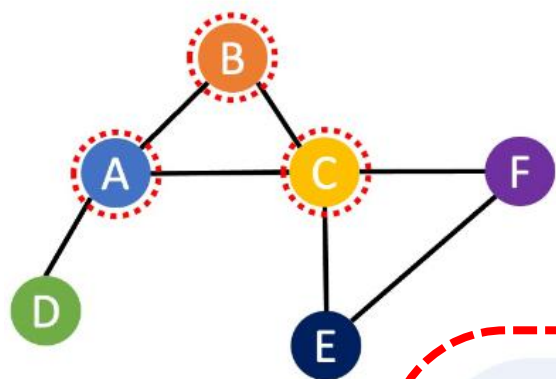
$$h_v^{(l)} = \sigma(W^{(l)} \circ (\frac{1}{|\mathcal{N}(v) \cup \{v\}|} \sum_{u \in \mathcal{N}(v) \cup \{v\}} h_u^{(l-1)}))$$



Batch size = 3



# Batch execution

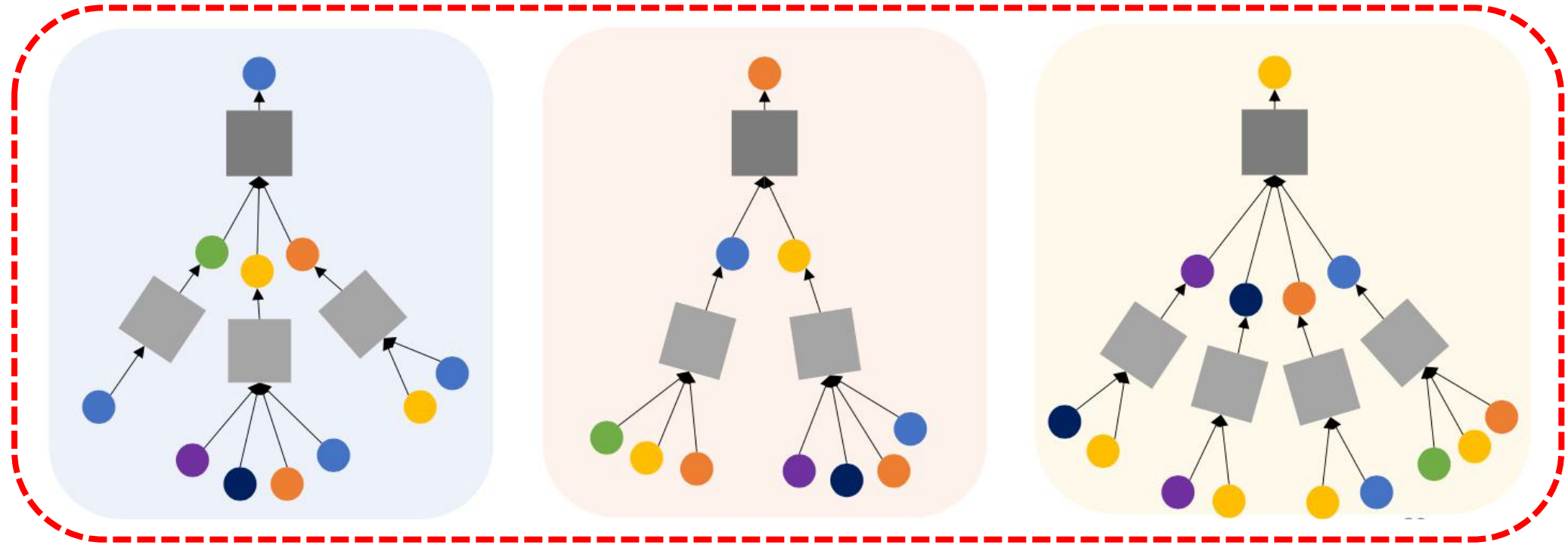


$$h_v^{(l)} = \sigma(W^{(l)} \circ (\frac{1}{|\mathcal{N}(v)+1|} \sum_{u \in \mathcal{N}(v) \cup \{v\}} h_u^{(l-1)}))$$
$$H^{(l)} = \sigma(\widetilde{A + I}) H^{(l-1)} W^{(l)}$$

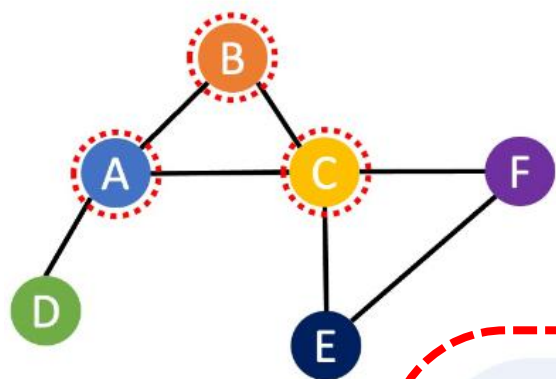
Node embedding matrix

(row-normalized) Adjacency matrix

Batch size = 3



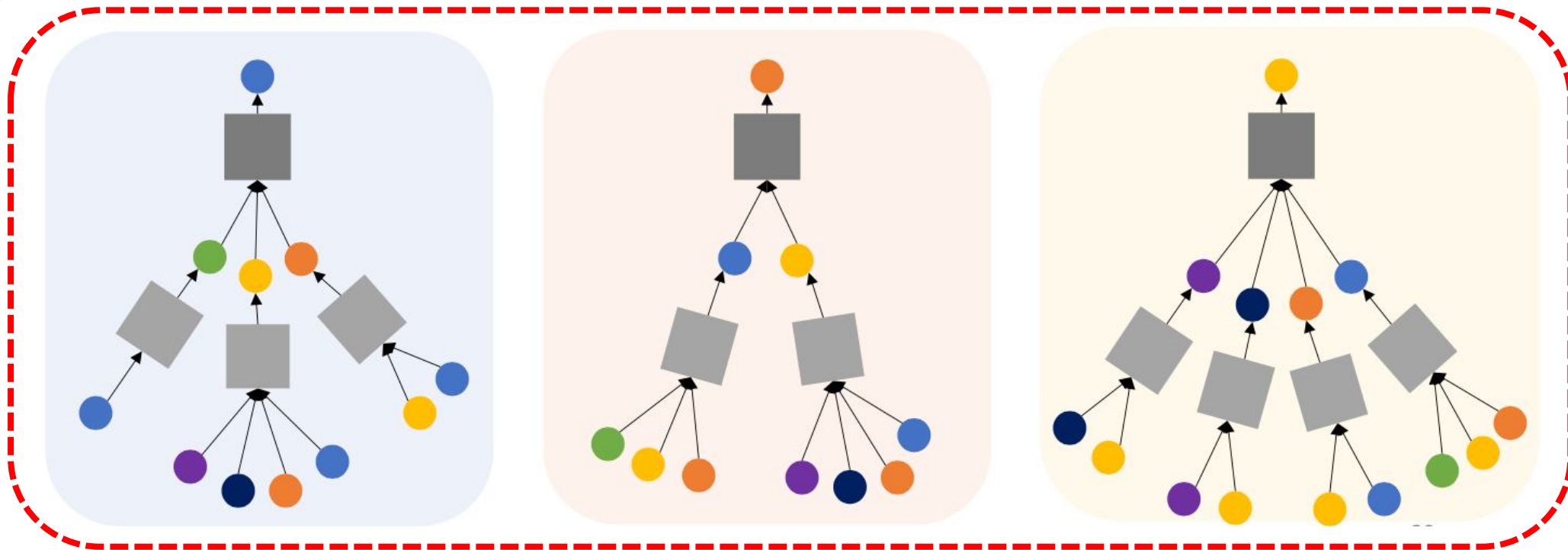
# Batch execution



$$h_v^{(l)} = \sigma(W^{(l)} \circ (\frac{1}{|\mathcal{N}(v)+1|} \sum_{u \in \mathcal{N}(v) \cup \{v\}} h_u^{(l-1)}))$$

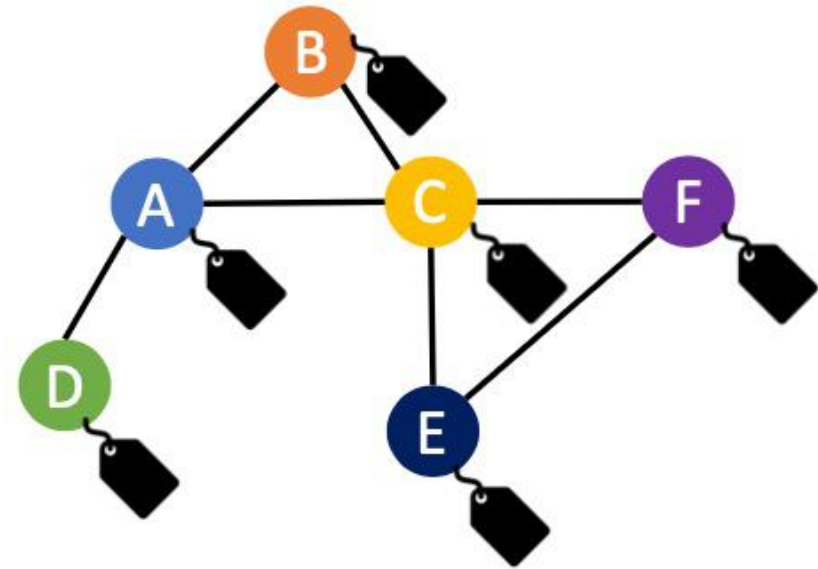
$$H^{(l)} = \sigma(\underbrace{(A + I)}_{\text{Fixed}} \underbrace{H^{(l-1)}}_{\text{Trainable}} \underbrace{W^{(l)}}_{\text{Trainable}})$$

Batch size = 3



# Downstream tasks

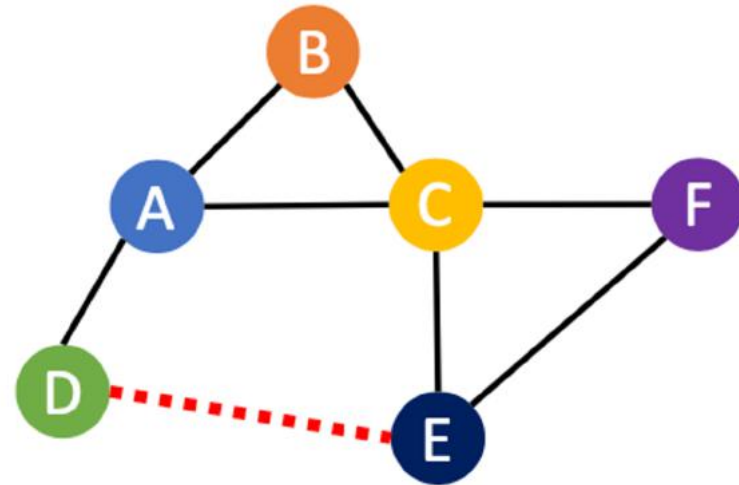
- Node-level prediction





# Downstream tasks

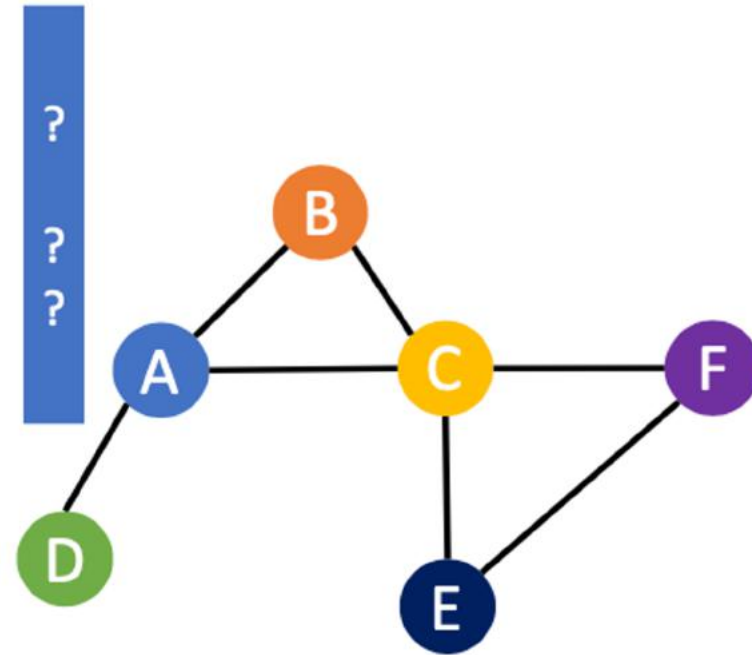
- Node-level prediction
- Edge-level prediction



**D and E are related enough  
to be connected?**

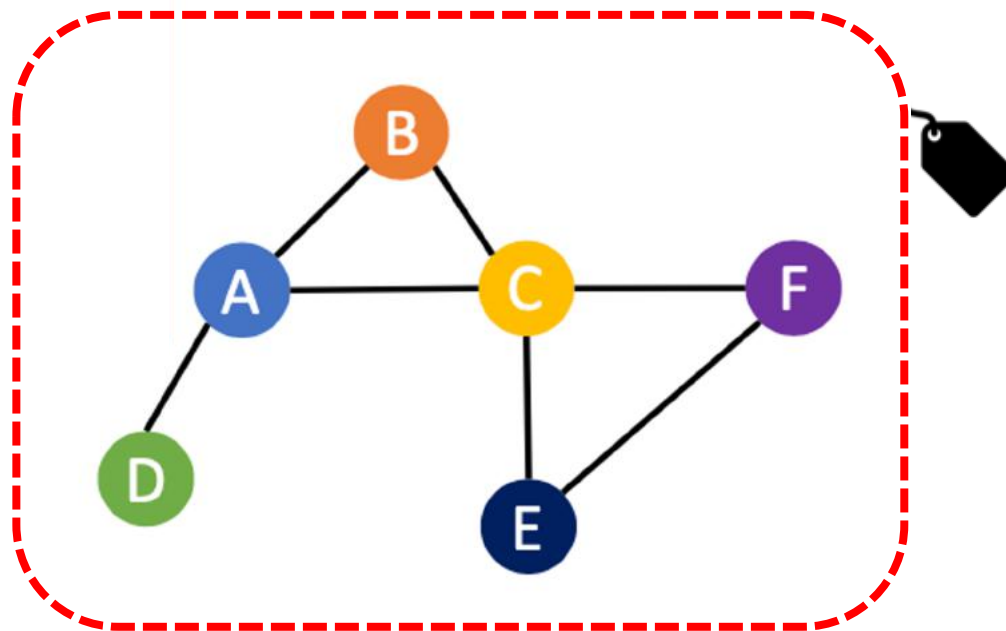
# Downstream tasks

- Node-level prediction
- Edge-level prediction
- Attribute-level prediction



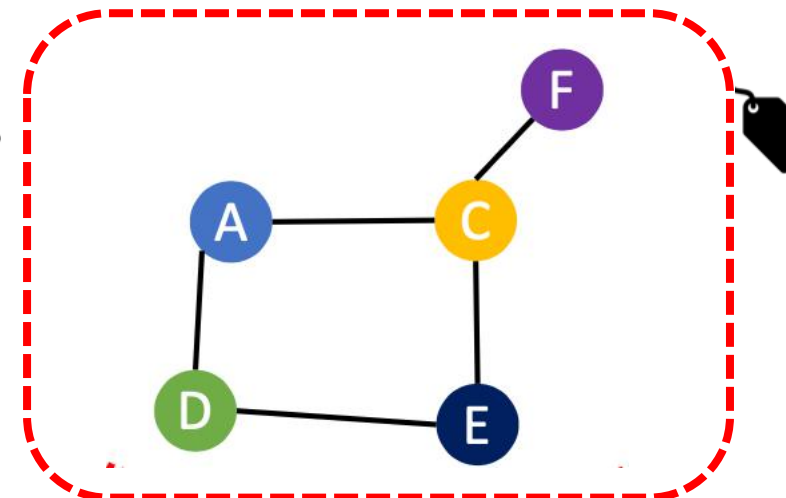
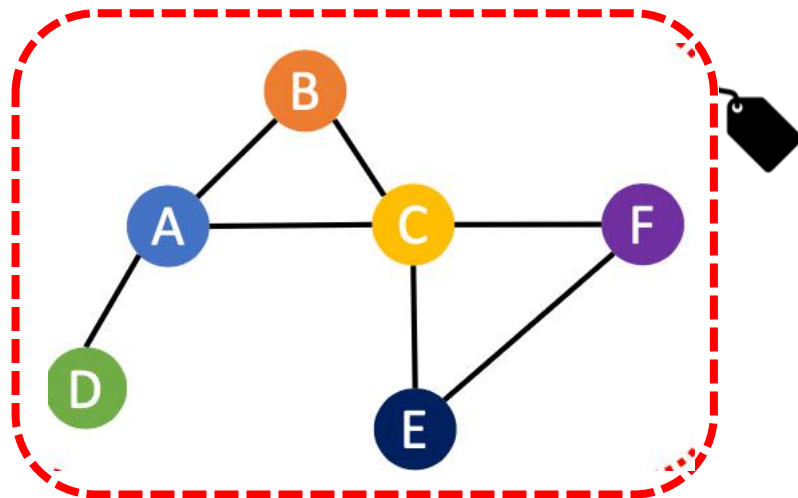
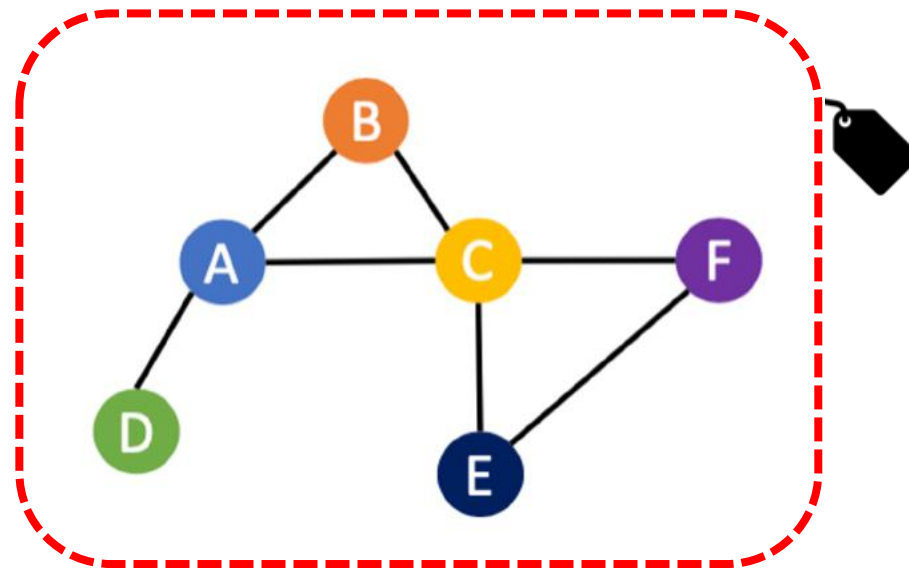
# Downstream tasks

- Node-level prediction
- Edge-level prediction
- Attribute-level prediction
- Graph-level prediction



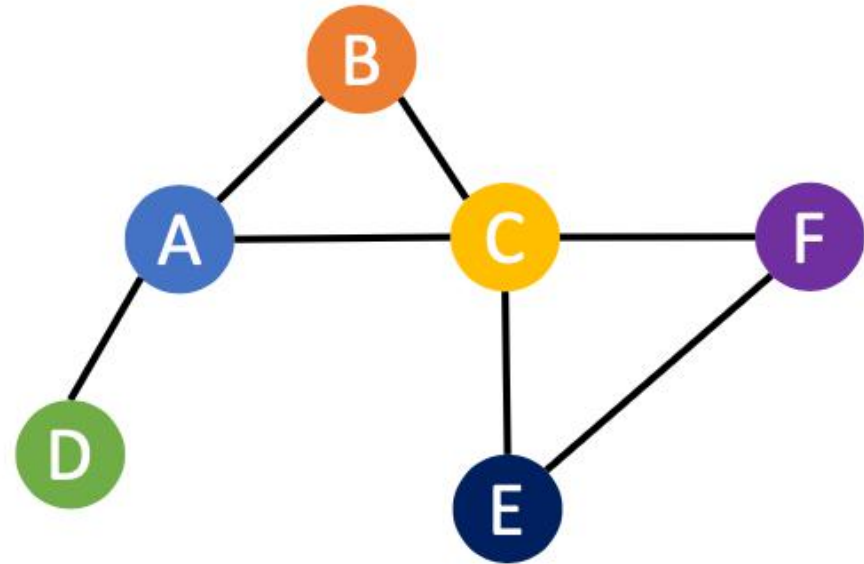
# Downstream tasks

- Node-level prediction
- Edge-level prediction
- Attribute-level prediction
- Graph-level prediction



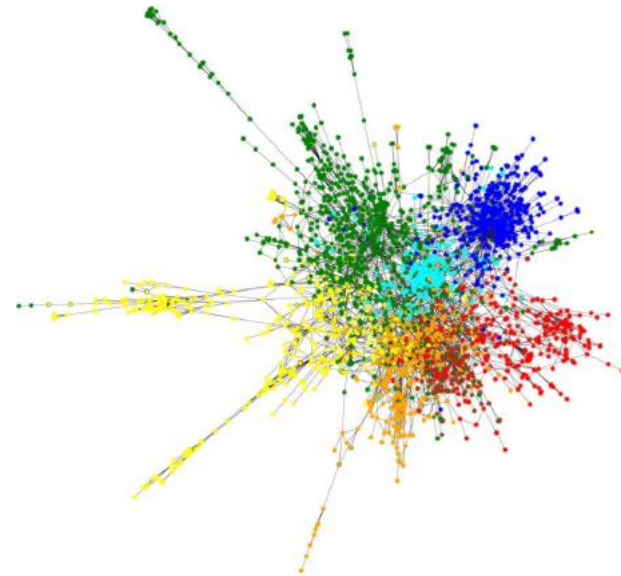
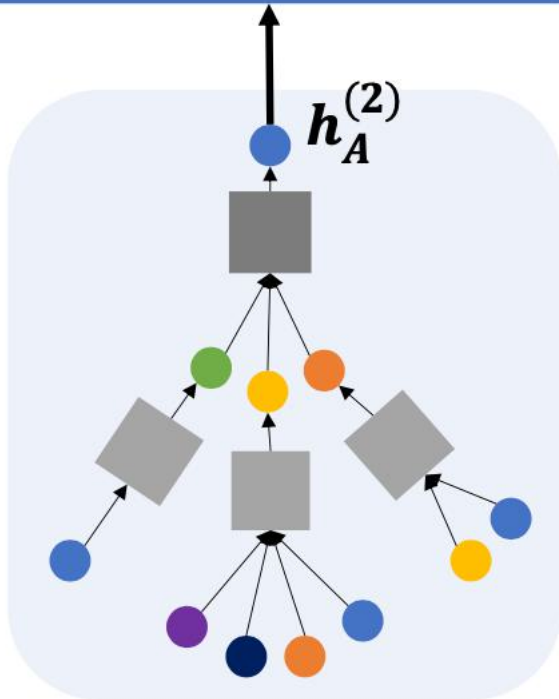
# Downstream tasks

- **Node-level prediction**
- Edge-level prediction
- Attribute-level prediction
- **Graph-level prediction**



# Node-level prediction tasks

## Node classification



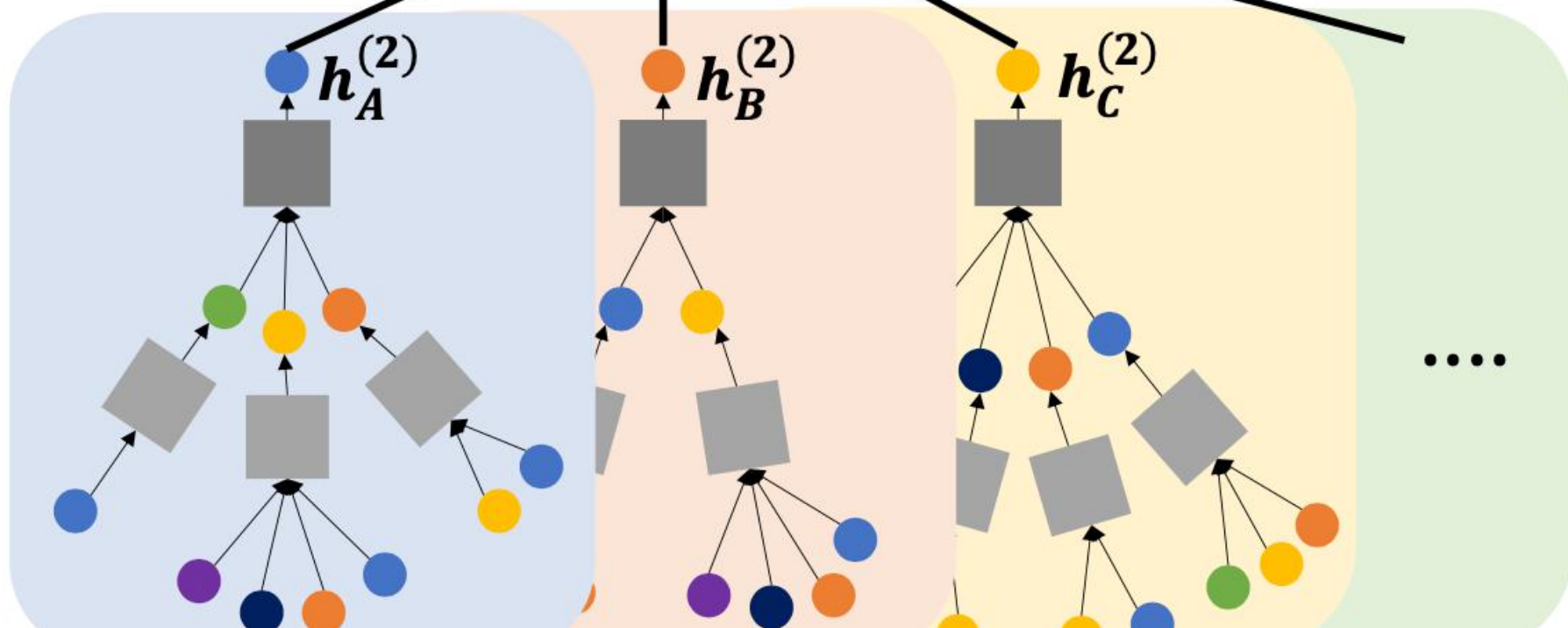
- Classify **papers** into topics on **citation networks**
- Cluster **posts** into subgroups on **Reddit networks**
- Classify **products** into categories on **Amazon co-purchase graphs**

# Graph-level prediction tasks

## Graph classification

$$h_G = \text{READOUT}(h_A^{(2)}, h_C^{(2)}, \dots, h_F^{(2)})$$

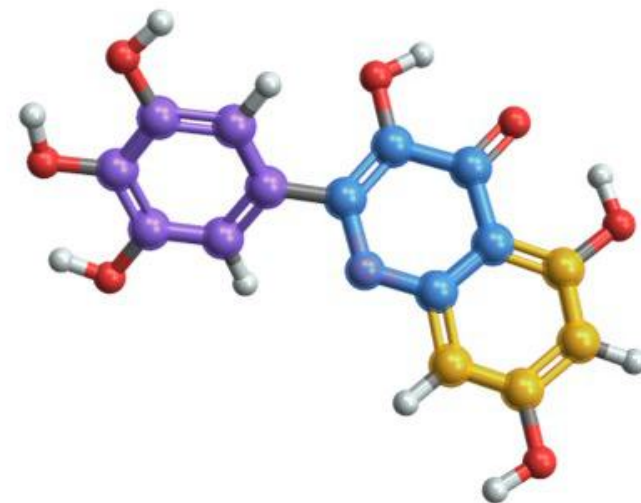
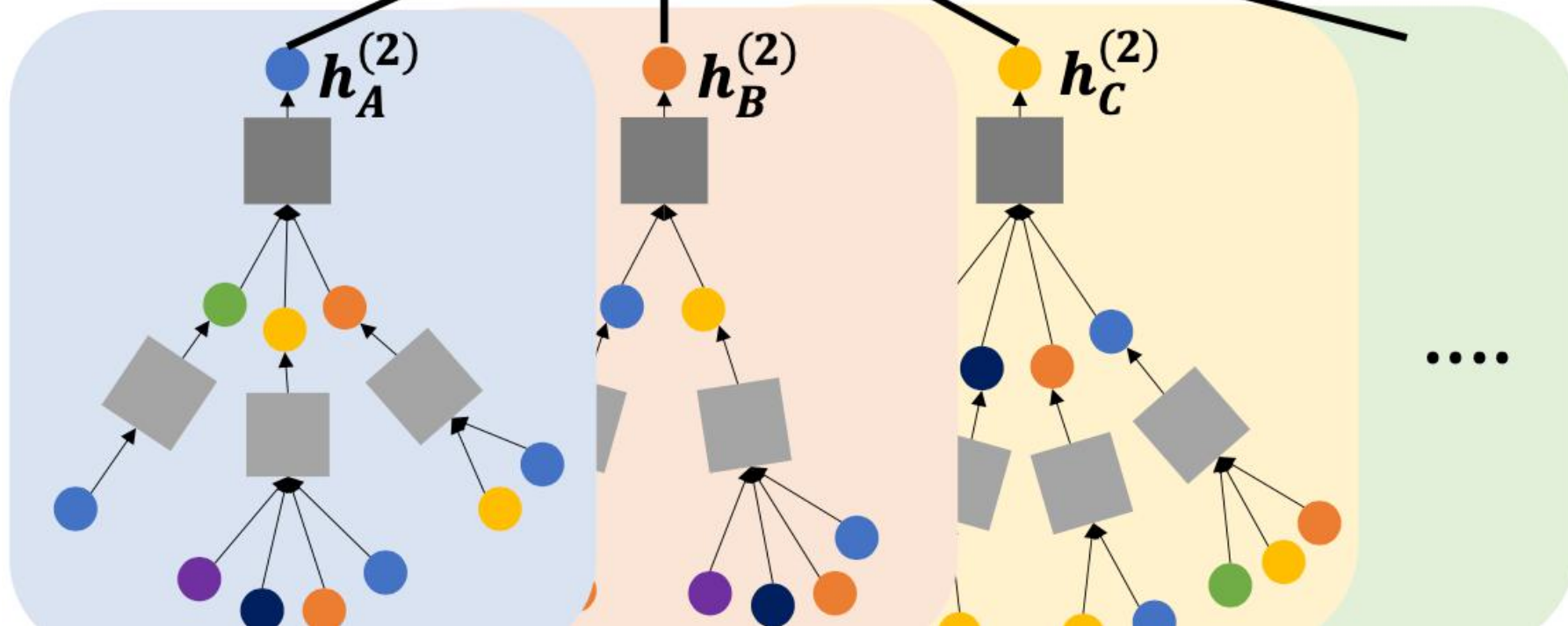
(ex) sum, average, min/max pooling of node embeddings



# Graph-level prediction tasks

Graph classification

$$h_G = \text{READOUT}(h_A^{(2)}, h_C^{(2)}, \dots, h_F^{(2)})$$

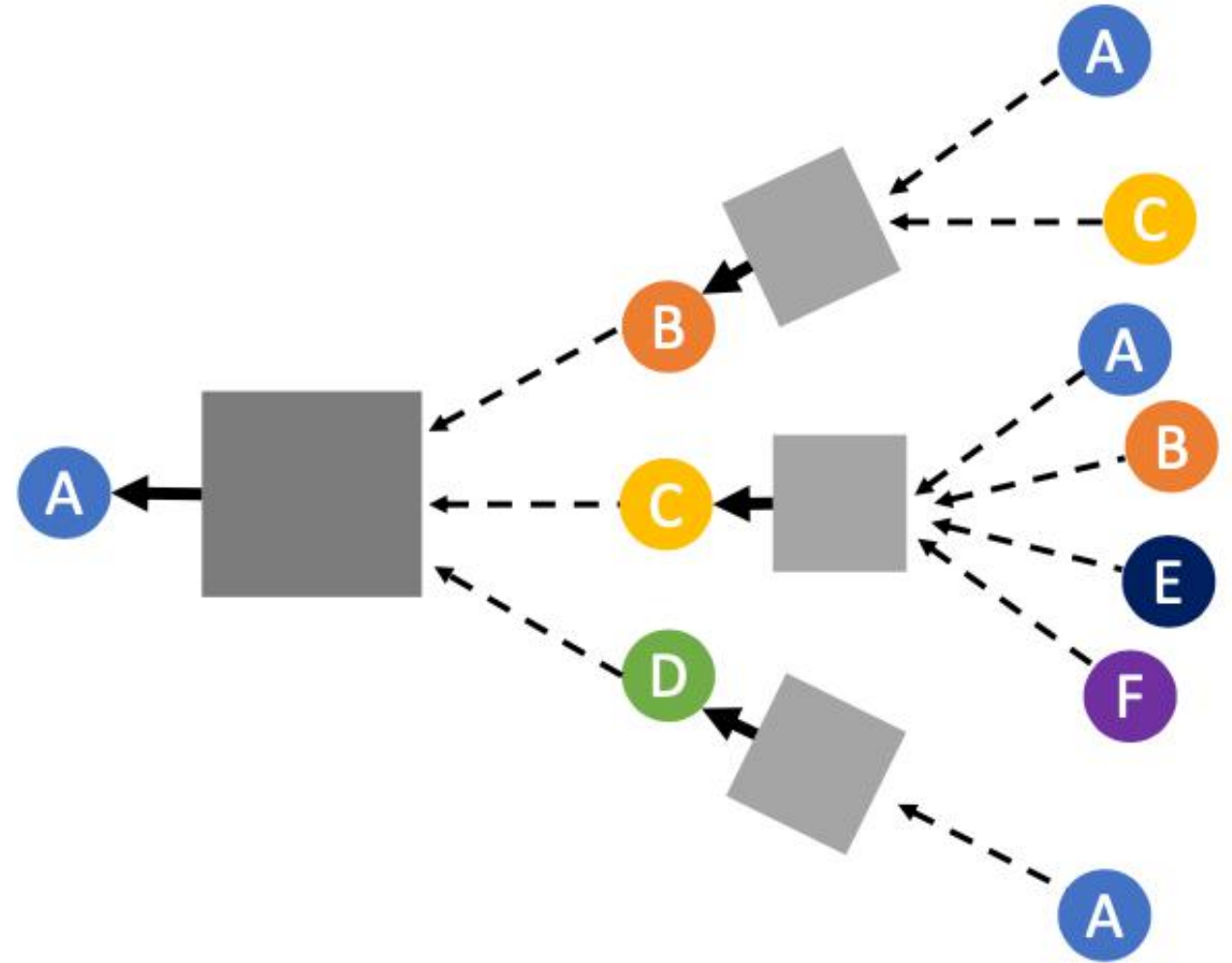
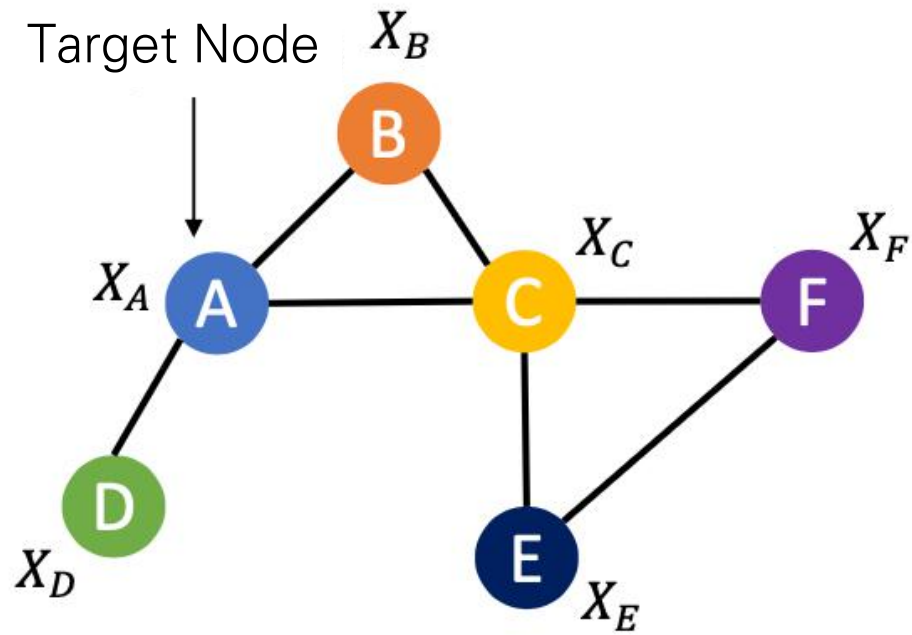


- Predict **properties of a molecule (graph)** where nodes are atoms and edges are chemical bonds

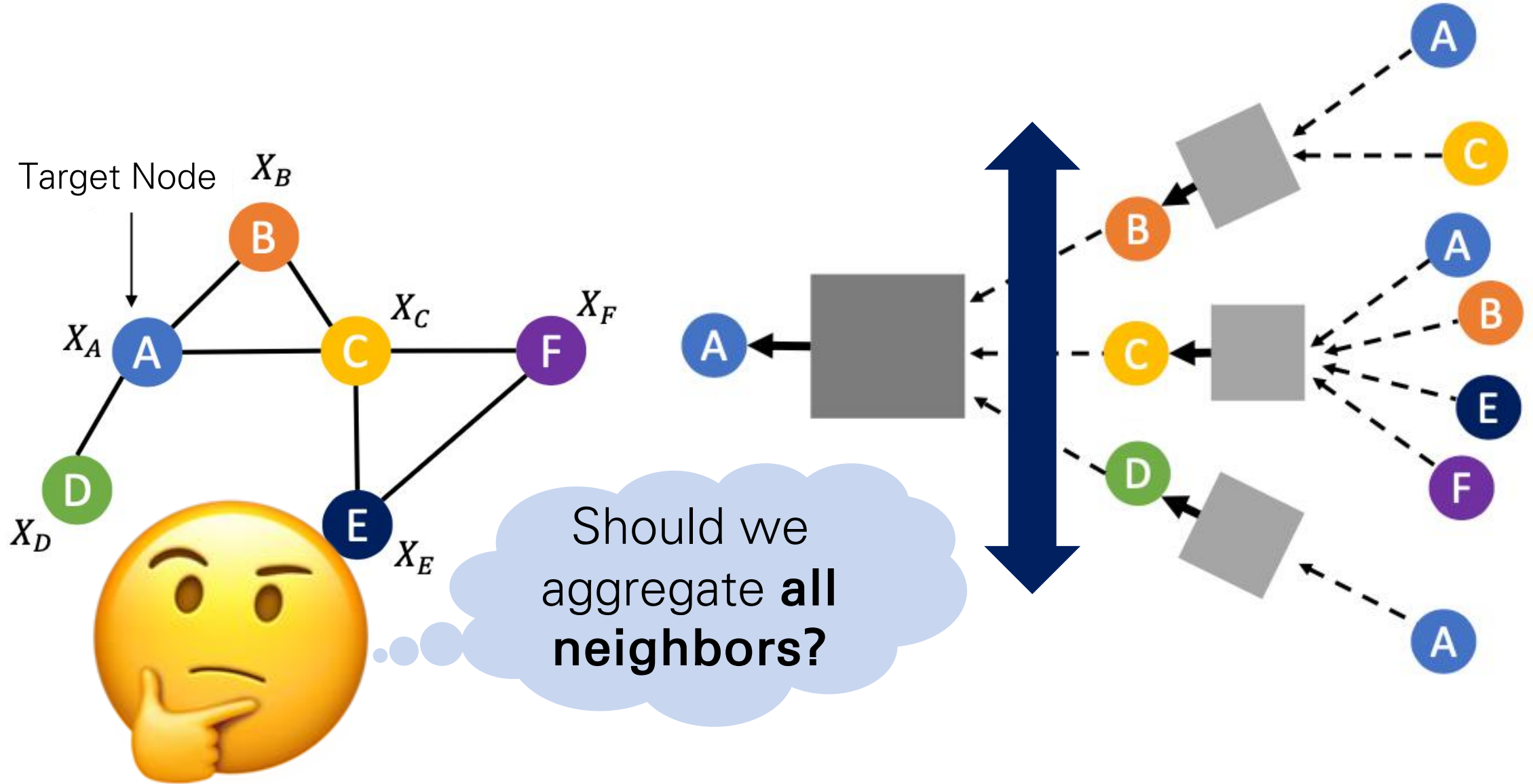


# More on aggregation and Transformation operations

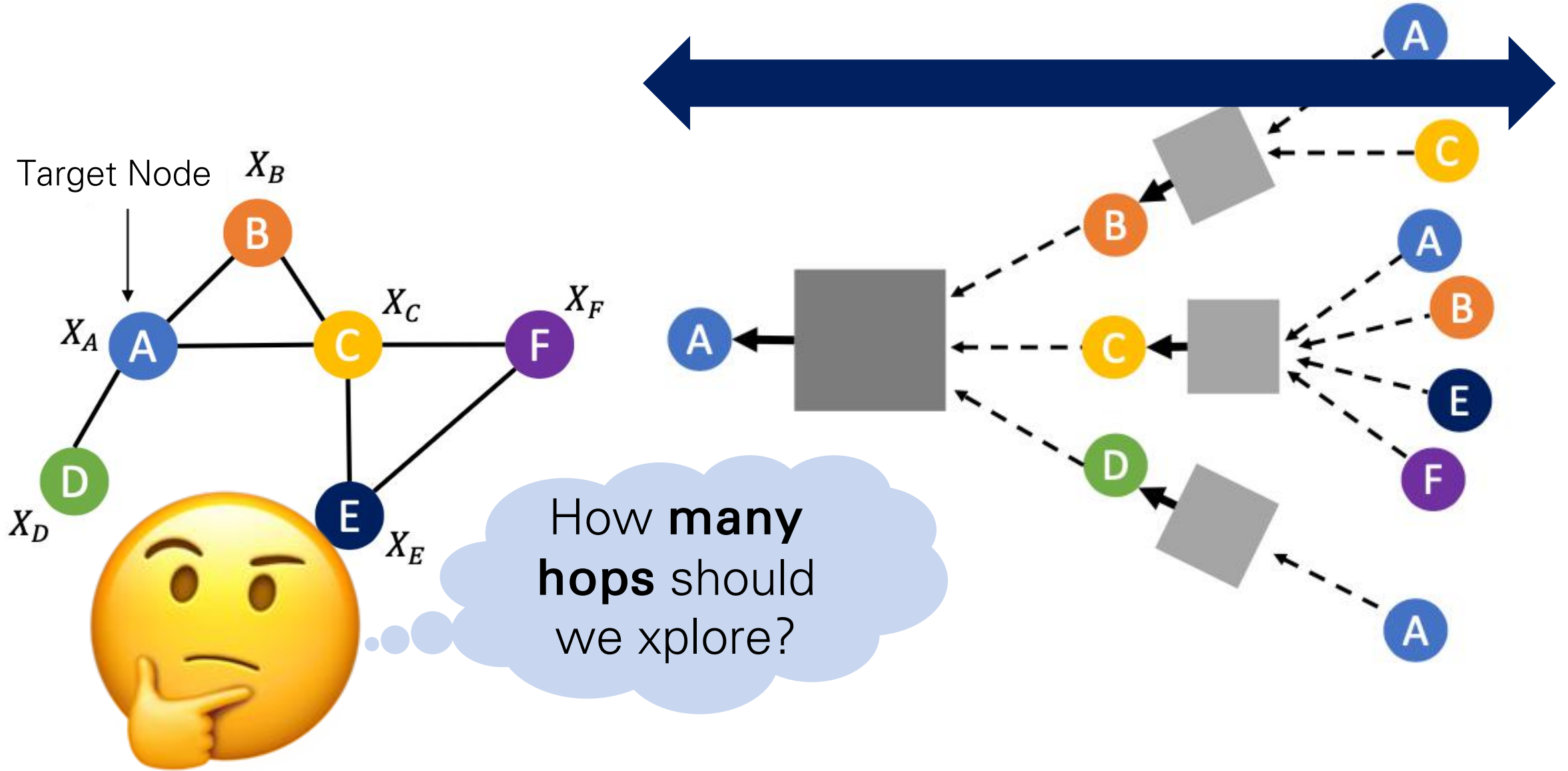
# Graph Neural Networks – Width



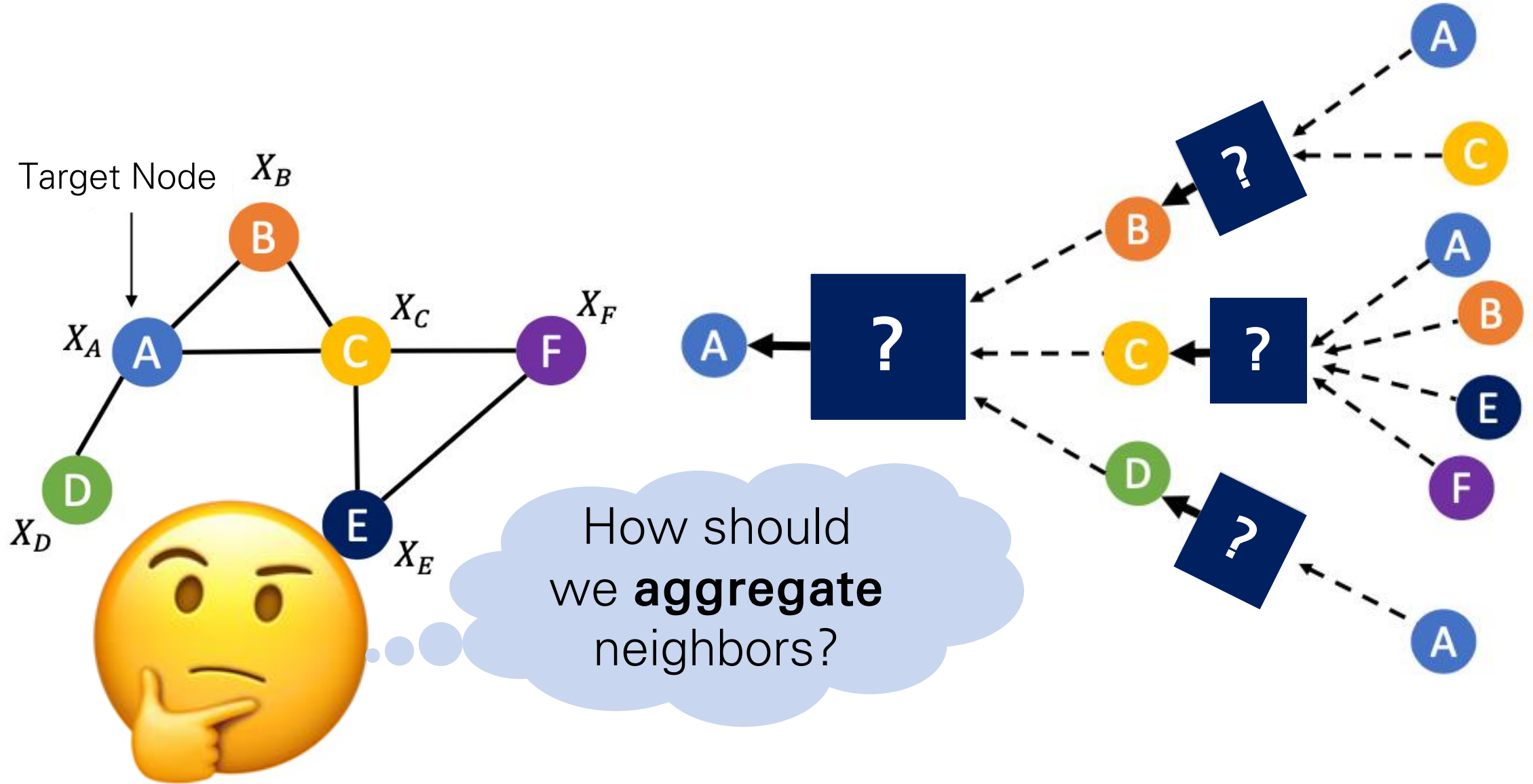
# Graph Neural Networks (GNNs)



# Graph Neural Networks (GNNs)

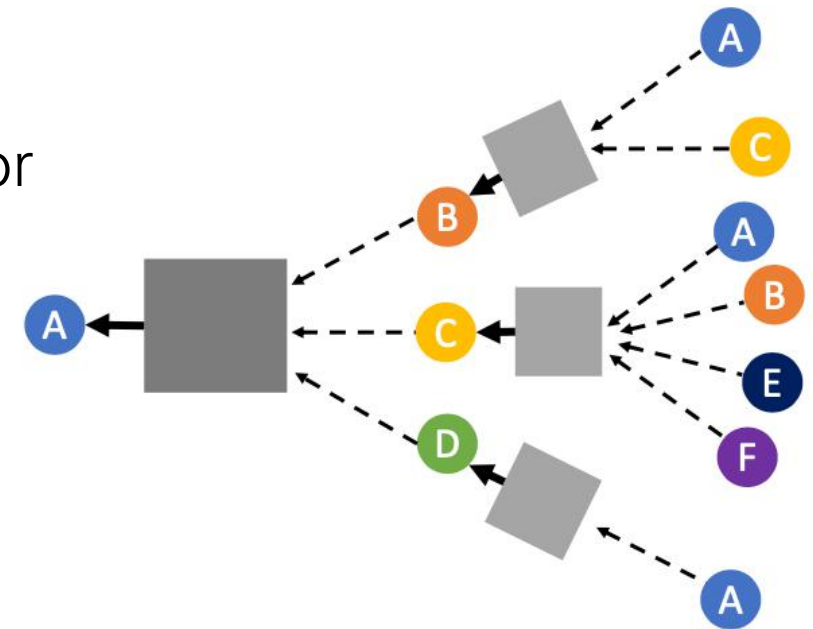


# Graph Neural Networks (GNNs)



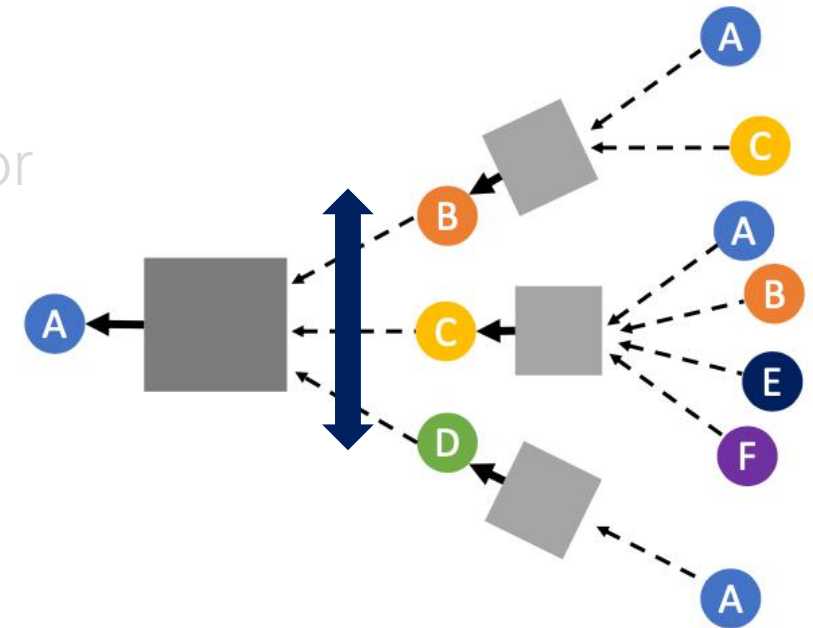
# Graph Neural Network Architectures

- Width
  - Which neighbors should we aggregate messages from?
- Depth
  - How many hops should we check?
- Aggregation
  - How should we aggregate messages from neighbor



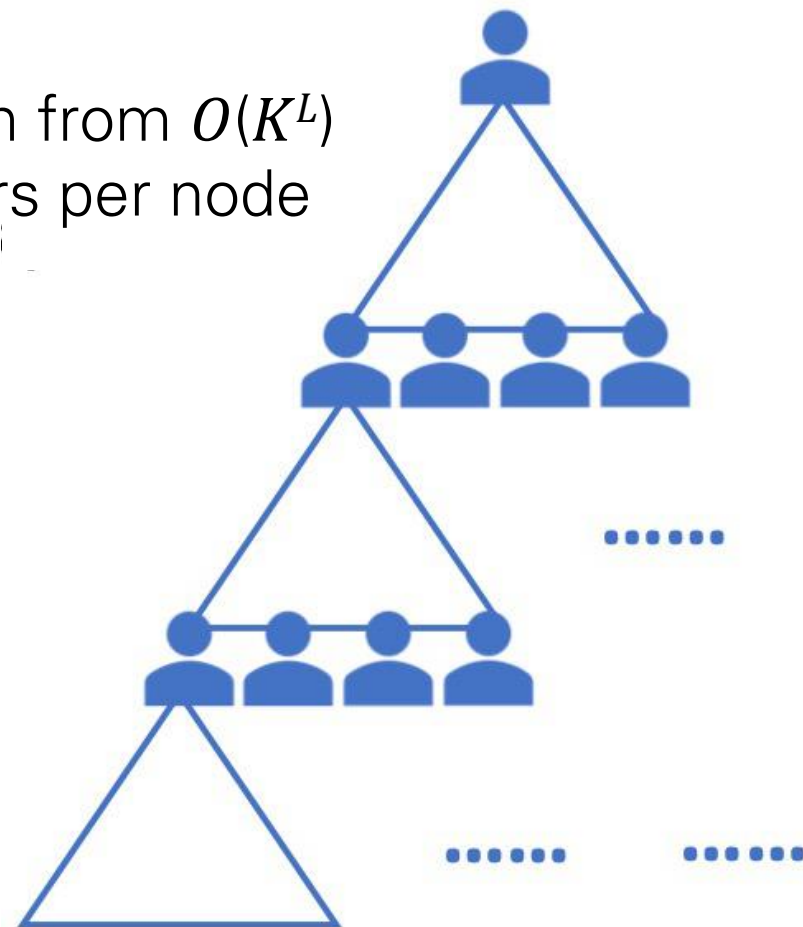
# Graph Neural Network Architectures

- Width
  - Which neighbors should we aggregate messages from?
- Depth
  - How many hops should we check?
- Aggregation
  - How should we aggregate messages from neighbor



# Aggregation Width in GNNs

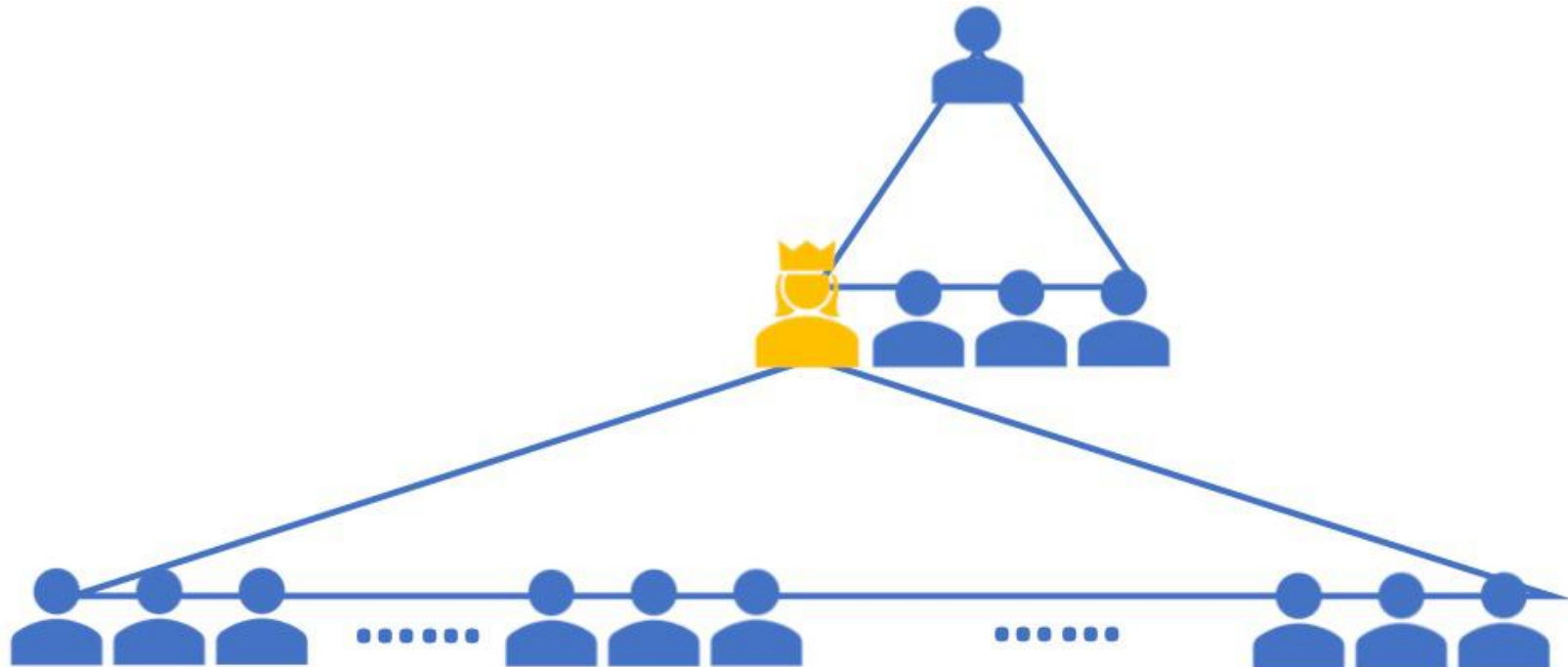
- If we aggregate all neighbors, GNNs have scalability issues
- Neighbor explosion
  - In  $L$ -layer GNNs, one node aggregates information from  $O(K^L)$  nodes where  $K$  is the average number of neighbors per node





# Aggregation Width in GNNs

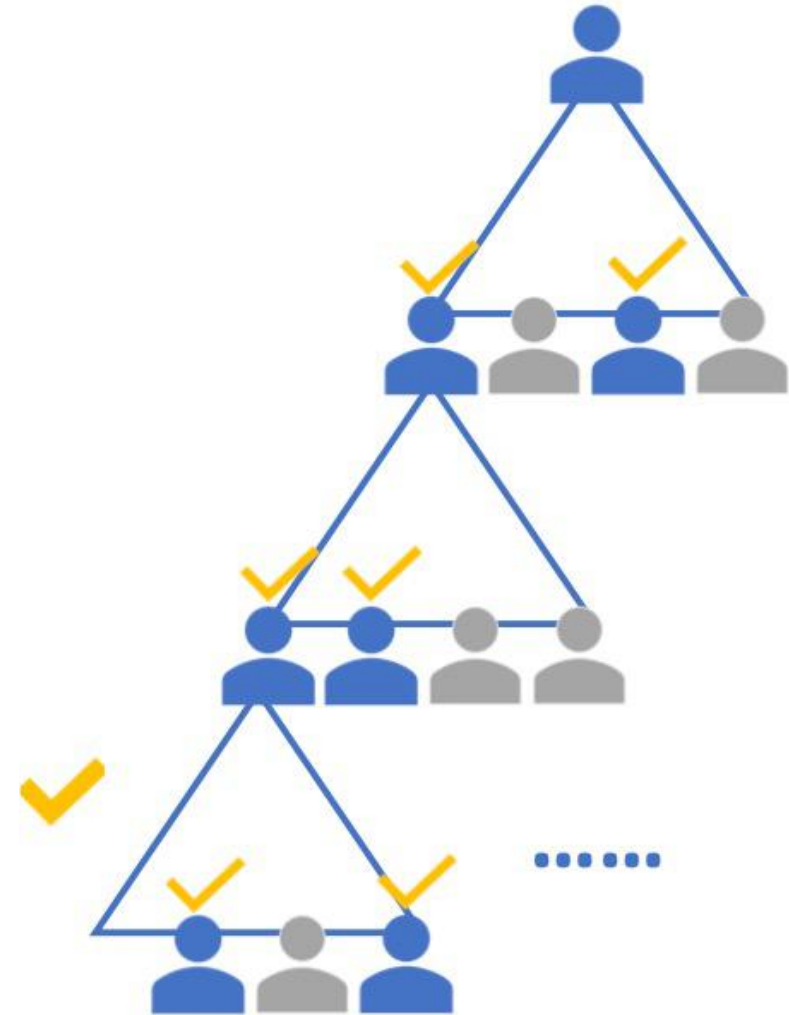
- If we aggregate all neighbors, GNNs have scalability issues
- Neighbor explosion
  - Hub nodes who are connected to a huge number of nodes



# Aggregation Width in GNNs

- Limit the neighborhood expansion by **sampling** a fixed number of neighbors

Sample the  
neighbors



# Aggregation Width in GNNs

- Random sampling
  - Assign **same** sampling probabilities to all neighbors
  - GraphSage<sup>[4]</sup>
- Importance sampling
  - Assign **different** sampling probabilities to all neighbors
  - FastGCN<sup>[5]</sup>, LADIES<sup>[6]</sup>, AS-GCN<sup>[7]</sup>, GCN-BS<sup>[8]</sup>, PASS<sup>[9]</sup>

[4] Will Hamilton, et al. "Inductive representation learning on large graphs"

[5] Jie Chen, et al. "Fastgcn: fast learning with graph convolutional networks via importance sampling"

[6] Difan Zou, et al. "Layer-Dependent Importance Sampling for Training Deep and Large Graph Convolutional Networks"

[7] Wenbing Huang, et al. "Adaptive sampling towards fast graph representation learning"

[8] Ziqi Liu, et al. "Bandit Samplers for Training Graph Neural Networks"

[9] Minji Yoon, et al. "Performance-Adaptive Sampling Strategy Towards Fast and Accurate Graph Neural Networks"

# Aggregation Width in GNNs

Importance sampling

: assign higher sampling probabilities to neighbors who

- **Minimize variance in sampling**
  - FastGCN<sup>[5]</sup>, LADIES<sup>[6]</sup>, AS-GCN<sup>[7]</sup>, GCN-BS<sup>[8]</sup>
- **Maximize GNN performance**
  - PASS<sup>[9]</sup>

[4] Will Hamilton, et al. "Inductive representation learning on large graphs"

[5] Jie Chen, et al. "Fastgcn: fast learning with graph convolutional networks via importance sampling"

[6] Difan Zou, et al. "Layer-Dependent Importance Sampling for Training Deep and Large Graph Convolutional Networks"

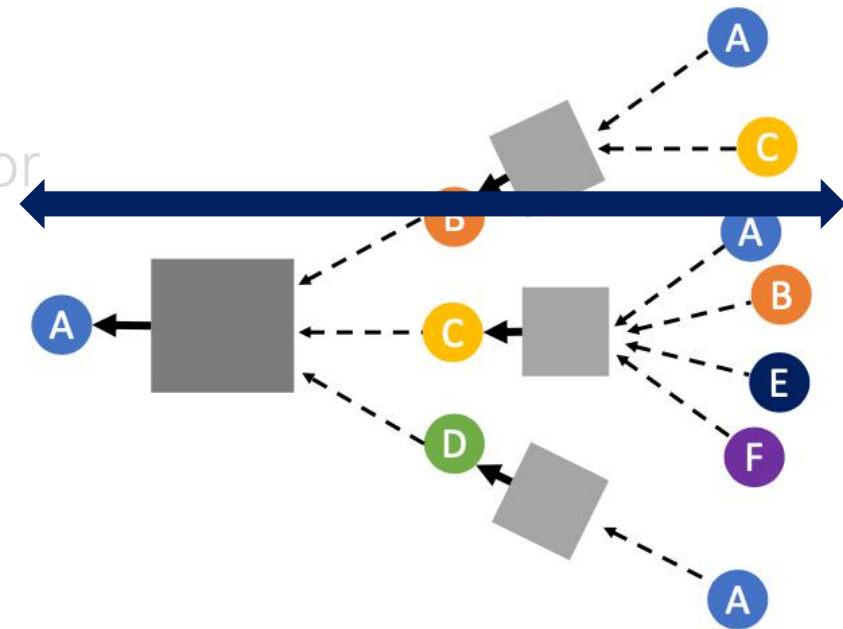
[7] Wenbing Huang, et al. "Adaptive sampling towards fast graph representation learning"

[8] Ziqi Liu, et al. "Bandit Samplers for Training Graph Neural Networks"

[9] Minji Yoon, et al. "Performance-Adaptive Sampling Strategy Towards Fast and Accurate Graph Neural Networks"

# Graph Neural Network Architectures

- Width
  - Which neighbors should we aggregate messages from?
- Depth
  - How many hops should we check?
- Aggregation
  - How should we aggregate messages from neighbors?







# Aggregation Depth in GNNs

- 2-layer or 3-layer GNNs are commonly used in real worlds

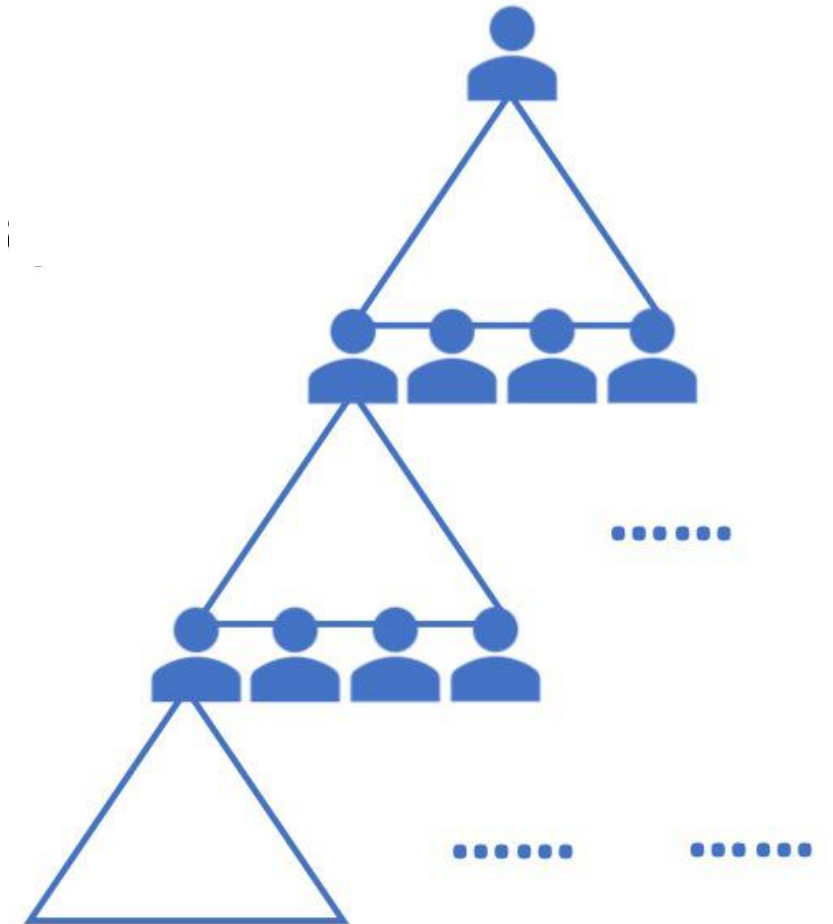


Wasn't it Deeeep Learning?



# Aggregation Depth in GNNs

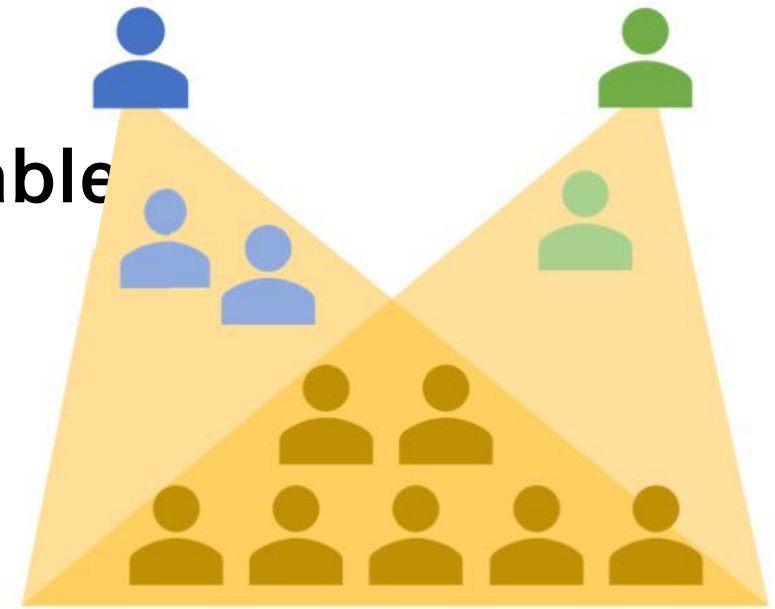
- When we increase the depth  $L$  more than this, GNNs face neighbor explosion  $O(K^L)$ 
  - **Over-smoothing**
  - **Over-squashing**



# Aggregation Depth in GNNs

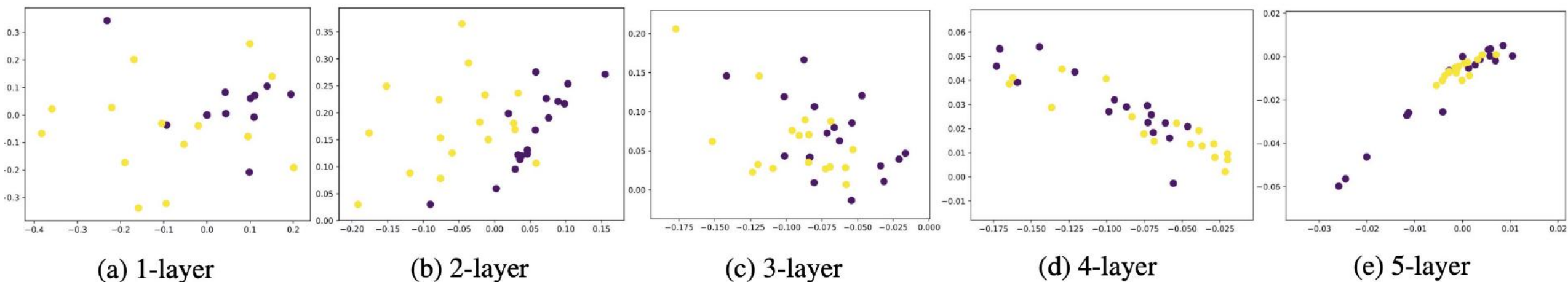
## Over-smoothing<sup>[10]</sup>

- When GNNs become deep, nodes share many neighbors
- Node embeddings become **indistinguishable**



# Aggregation Depth in GNNs

- Over-smoothing<sup>[10]</sup>
- Node embeddings of Zachary's karate club network with GNNs



# Aggregation Depth in GNNs

## Mitigate over-smoothing

PairNorm<sup>[11]</sup>

- Keep total pairwise squared distance (TPSD) **constant** across layers
- Push away pairs that are not connected

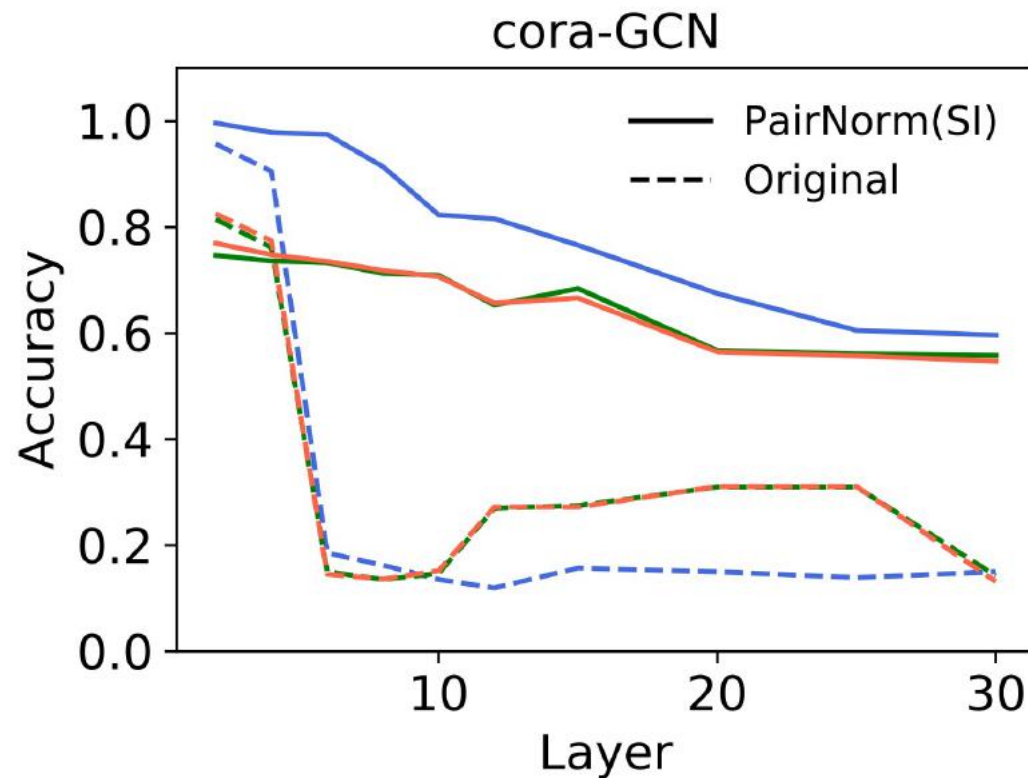
$$\text{TPSD}(\dot{X}) = \sum_{(i,j) \in \mathcal{E}} \|\dot{x}_i - \dot{x}_j\|_2^2 + \sum_{(i,j) \notin \mathcal{E}} \|\dot{x}_i - \dot{x}_j\|_2^2 = \mathbf{C}$$

The equation is annotated with a red dashed box around the first sum labeled "Connected pairs" and a blue dashed box around the second sum labeled "Disconnected pairs".

# Aggregation Depth in GNNs

## Mitigate over-smoothing

PairNorm<sup>[11]</sup>

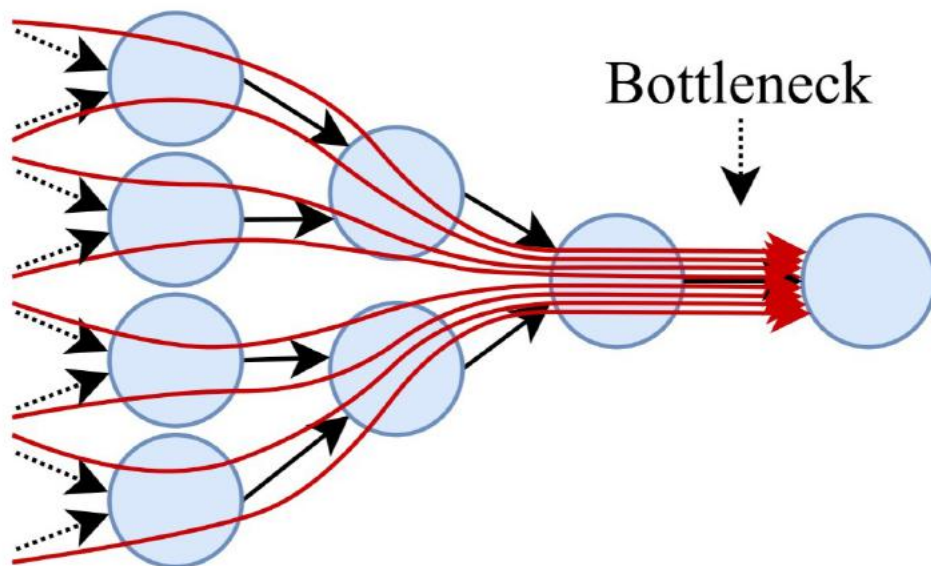


[11] Lingxiao Zhao, et al. "PairNorm: Tackling Oversmoothing in GNNs"

# Aggregation Depth in GNNs

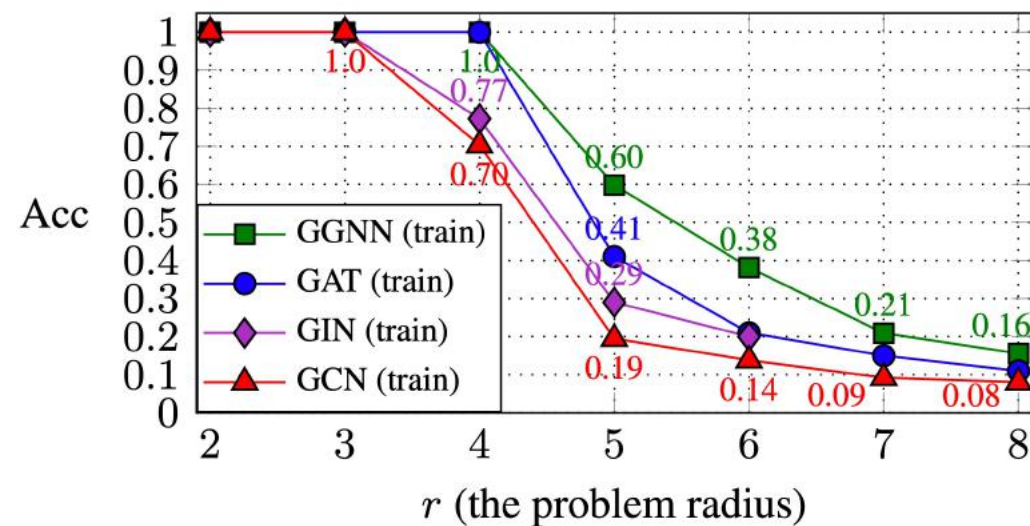
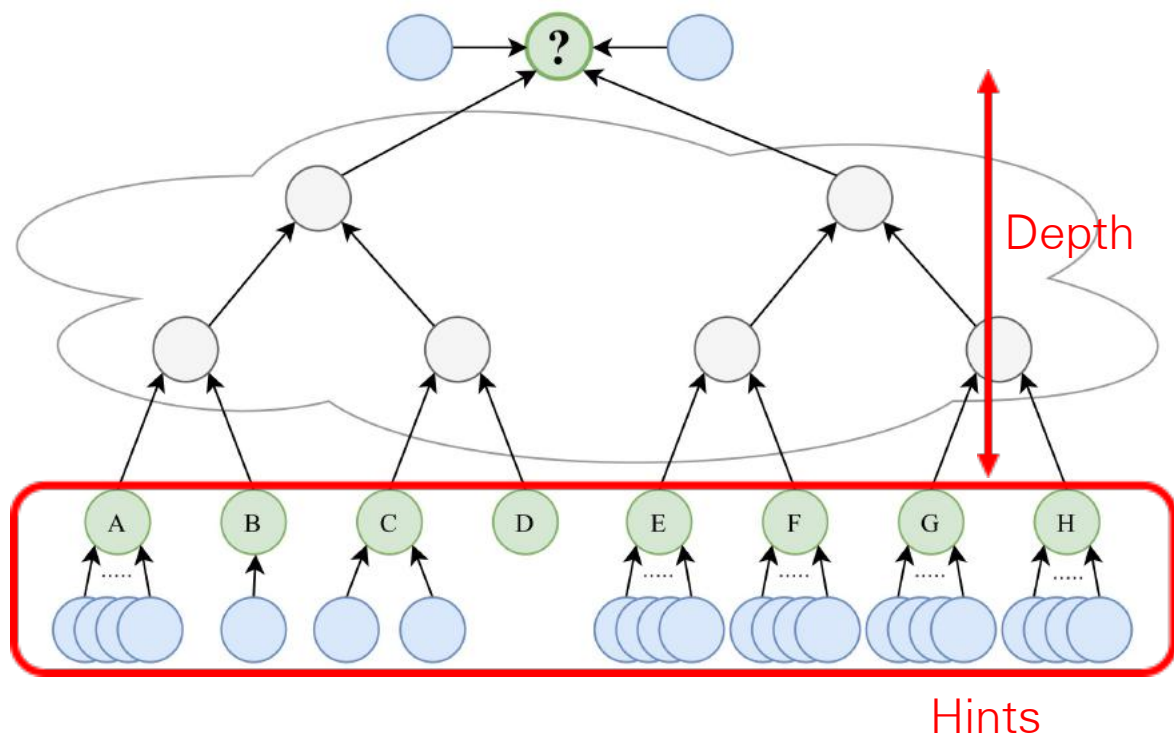
## Over-squashing<sup>[12]</sup>

- A node's exponentially-growing neighborhood is compressed into a fixed-size vector



# Aggregation Depth in GNNs

## Over-squashing<sup>[12]</sup>



# Aggregation Depth in GNNs

Decoupling the two concepts of depths in GNNs<sup>[13]</sup>

- **Depth-1**: neighborhood that each node aggregates information from
- **Depth-2**: number of layers in GNNs

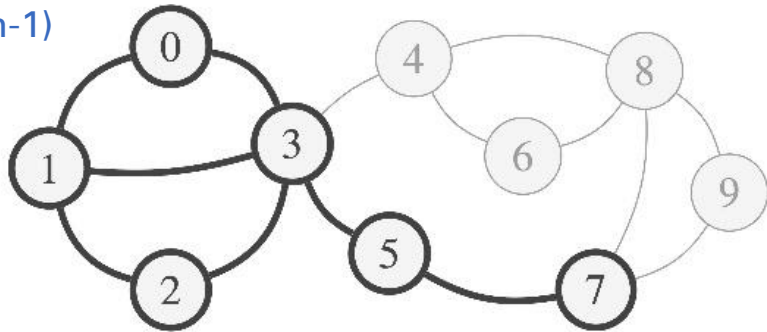


# Aggregation Depth in GNNs

Decoupling the two concepts of depths in GNNs<sup>[13]</sup>

- **Depth-1:** neighborhood that each node aggregates information from
- **Depth-2:** number of layers in GNNs

Depth of neighborhood  
(Depth-1)



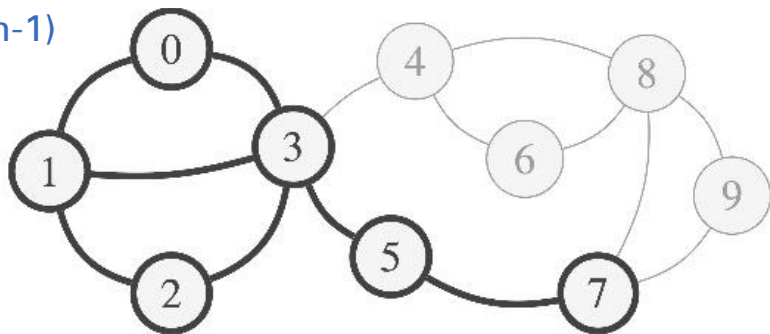
$$\mathcal{G}_s = \text{SAMPLE}(\mathcal{G})$$

# Aggregation Depth in GNNs

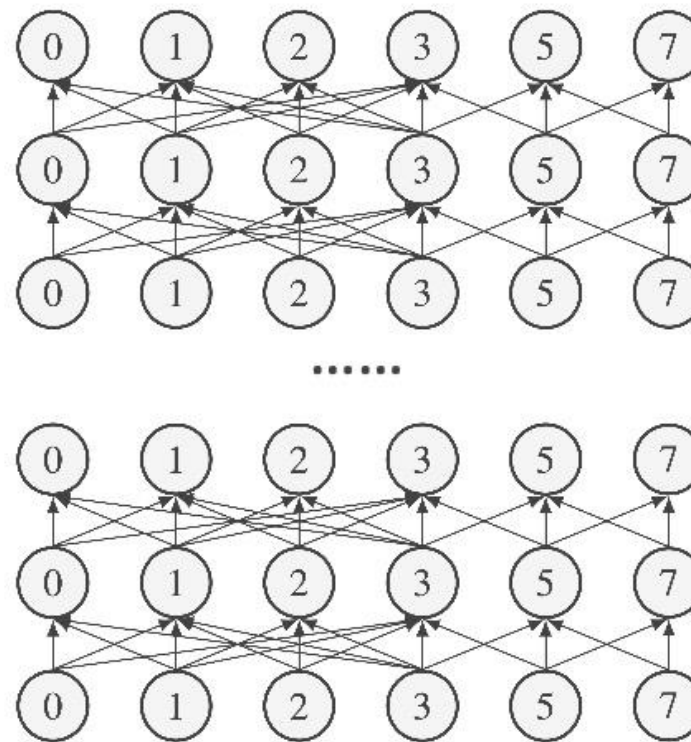
Decoupling the two concepts of depths in GNNs<sup>[13]</sup>

- **Depth-1**: neighborhood that each node aggregates information from
- **Depth-2**: number of layers in GNNs

Depth of neighborhood  
(Depth-1)



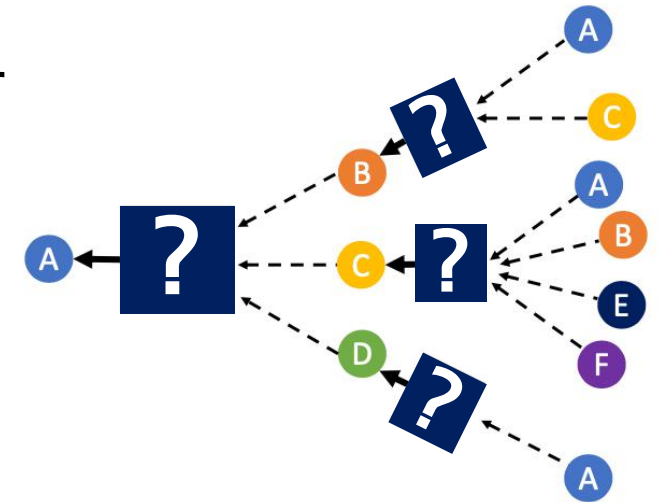
$$\mathcal{G}_s = \text{SAMPLE}(\mathcal{G})$$



Depth of GNN  
(Depth-2)

# Graph Neural Network Architectures

- Width
  - Which neighbors should we aggregate messages from?
- Depth
  - How many hops should we check?
- Aggregation
  - How should we aggregate messages from neighbor



# Aggregation strategy in GNNs

In each layer  $l$  :

**Aggregate** over neighbors

$$m_v^{(l-1)} = \mathbf{f}^{(l)}\left(h_v^{(l-1)}, \{h_u^{(l-1)} : u \in \mathcal{N}(v)\}\right)$$

**Transform** messages

$$h_v^{(l)} = \mathbf{g}^{(l)}(m_v^{(l-1)})$$

# Aggregation strategy in GNNs

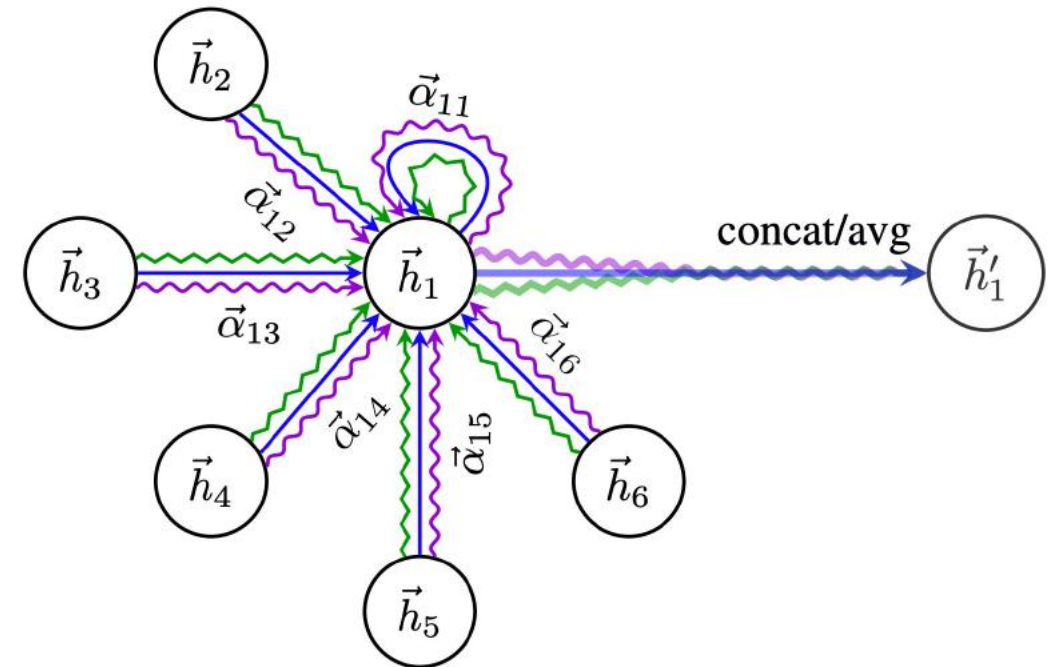
- GCN<sup>[1]</sup>
  - Average embeddings of neighboring nodes

[1] Kipf, Thomas N., et al. "Semi-supervised classification with graph convolutional networks."

# Aggregation strategy in GNNs

- GAT<sup>[14]</sup>
  - Different weights to different nodes in a neighborhood
  - Multi-head attention

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



# Aggregation strategy in GNNs

In each layer  $l$  :

**Aggregate** over neighbors

$$m_v^{(l-1)} = \mathbf{f}^{(l)} \left( h_v^{(l-1)}, \{h_u^{(l-1)} : u \in \mathcal{N}(v)\} \right)$$

Core part of GNNs

**Transform** messages

$$h_v^{(l)} = \mathbf{g}^{(l)}(m_v^{(l-1)})$$

Any neural network module can fit in  
1-layer MLP is commonly used

# Aggregation strategy in GNNs

Power of **GNNs**

=

Power of **aggregation strategies**



# Aggregation strategy in GNNs

- By measuring the power of GNNs, we can find the best aggregation strategy!!



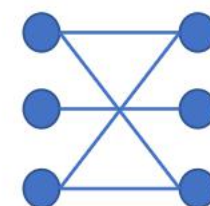
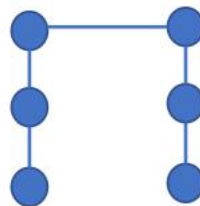
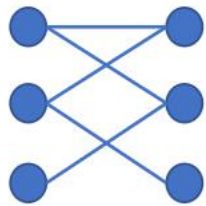
# Aggregation strategy in GNNs

- By measuring the power of GNNs, we can find the best aggregation strategy!!
- But.. what is the power of GNNs and how can we measure it?



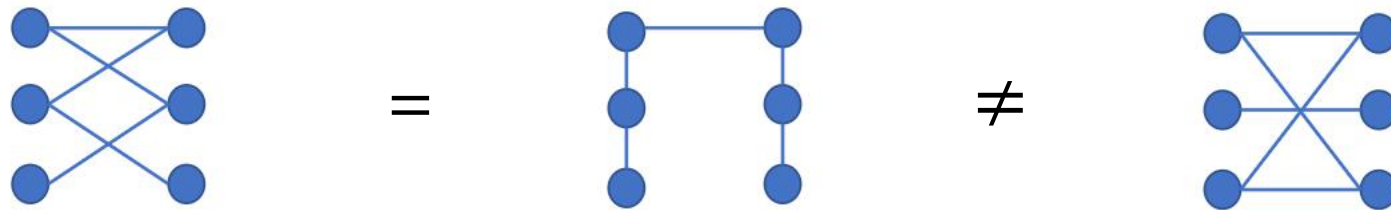
# Aggregation strategy in GNNs

- How powerful are Graph Neural Networks?<sup>[2]</sup>
- Metric
  - Graph-level prediction task
  - Can a GNN model distinguish two non-isomorphic graphs?




# Aggregation strategy in GNNs

- How powerful are Graph Neural Networks?<sup>[2]</sup>
- Metric
  - Graph-level prediction task
  - Can a GNN model distinguish two non-isomorphic graphs?



# Aggregation strategy in GNNs

- How powerful are Graph Neural Networks?<sup>[2]</sup>
  - Any aggregation-based GNN is at most as powerful as the **WL test**<sup>[15]</sup>
  - Maximum power = aggregation strategy is injective


$$f(x_1) = f(x_2) \Rightarrow x_1 = x_2$$

[2] Keyulu Xu., et al. "How Powerful are Graph Neural Networks?"

[15] Boris Weisfeiler and AA Leman. "A reduction of a graph to a canonical form and an algebra arising during this reduction"

# Aggregation strategy in GNNs

- How powerful are Graph Neural Networks?<sup>[2]</sup>
  - Any aggregation-based GNN is at most as powerful as the **WL test**<sup>[15]</sup>
  - Maximum power = aggregation strategy is injective
  - (ex) summation



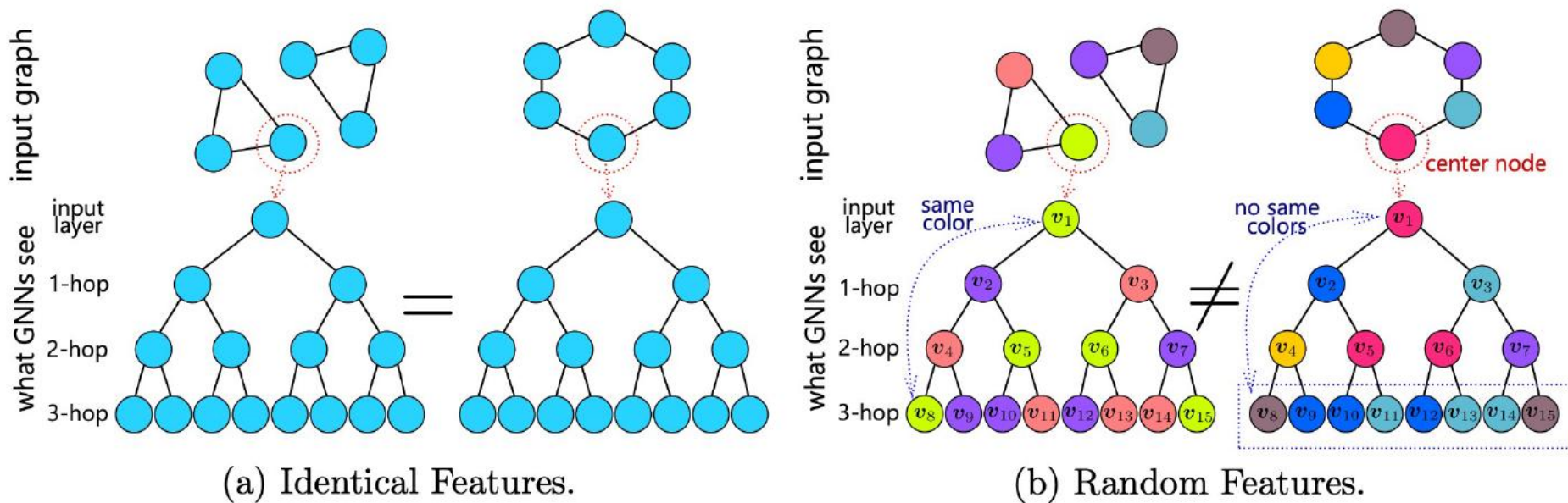
**Mean and Max both fail, while Sum can distinguish them!!**

# Aggregation strategy in GNNs

- Can we make more powerful GNNs?
  - Very active area, with many open problems

# Aggregation strategy in GNNs

- Can we make more powerful GNNs?
- Augment nodes with randomized/positional features<sup>[16]</sup>

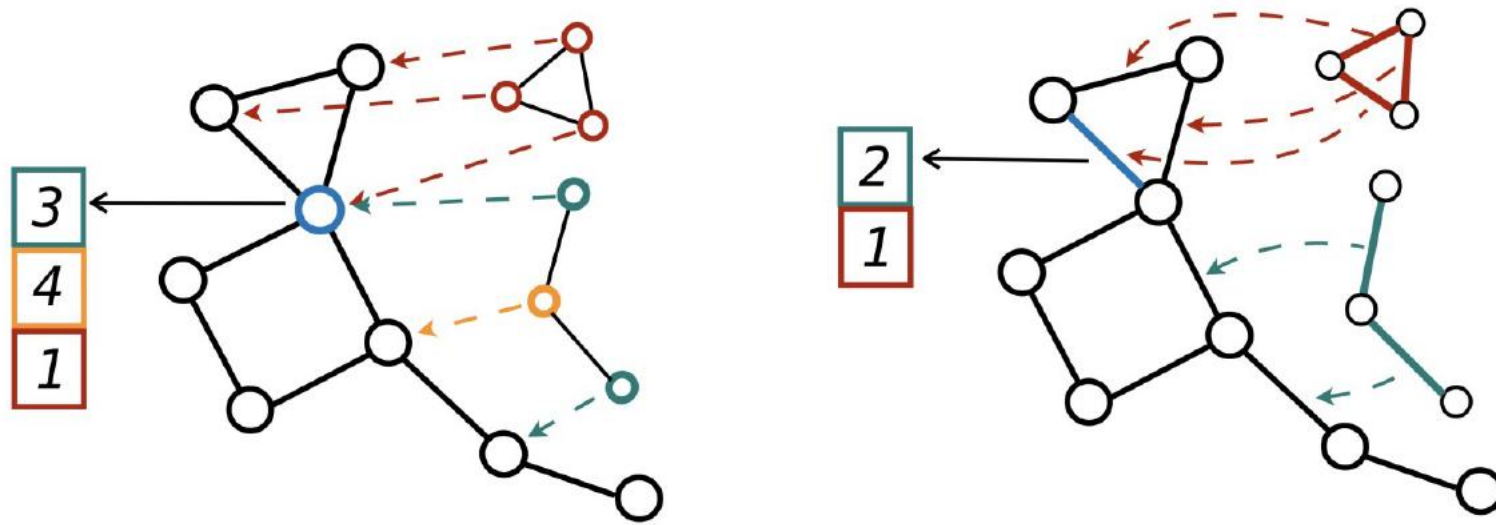


[16] Ryoma Sato, et al. "Random Features Strengthen Graph Neural Networks"



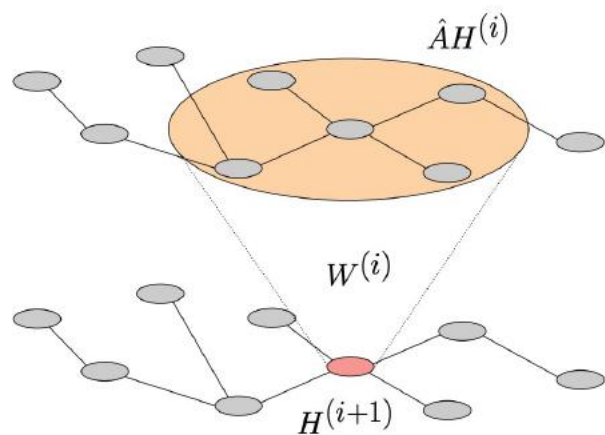
# Aggregation strategy in GNNs

- Can we make more powerful GNNs?
- Augment nodes with randomized/positional features<sup>[16]</sup>



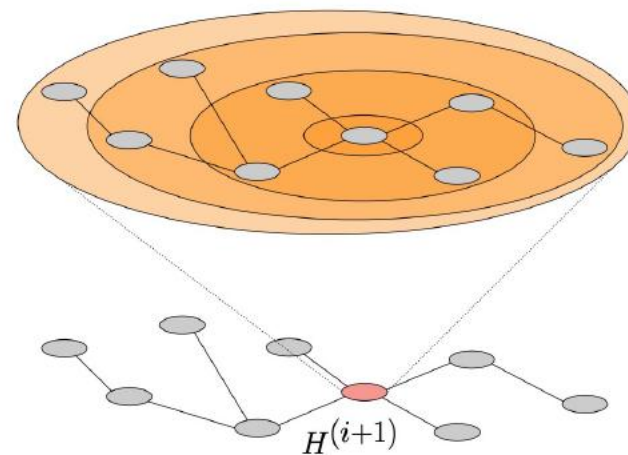
# Aggregation strategy in GNNs

- Can we make more powerful GNNs?
- Directly aggregates k-hop information by using adjacency matrix powers<sup>[18]</sup>



$$H^{(i+1)} = \sigma(\hat{A}H^{(i)}W^{(i)})$$

(a) Traditional graph convolution.

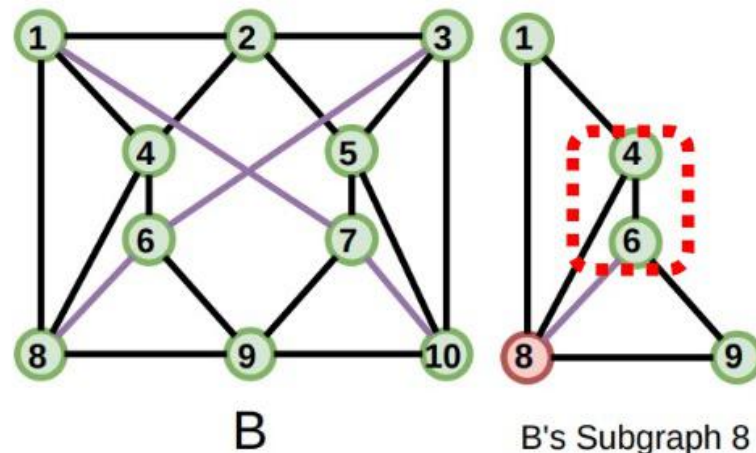
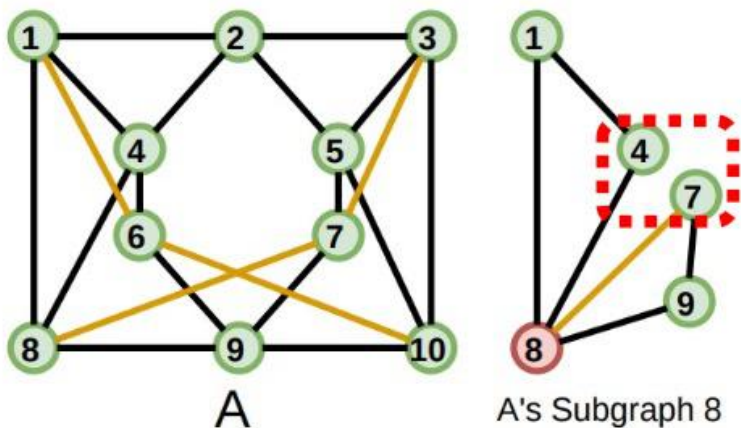


$$H^{(i+1)} = \sigma(\hat{A}^0 H^{(i)} W_0^{(i)} | \hat{A}^1 H^{(i)} W_1^{(i)} | \dots)$$

(b) Our mixed feature model.

# Aggregation strategy in GNNs

- Can we make more powerful GNNs?
- Extending local aggregation in GNNs from star patterns to general subgraph patterns<sup>[19]</sup>



# Aggregation strategy in GNNs

- [20] proves that there isn't a clear single "winner" aggregator

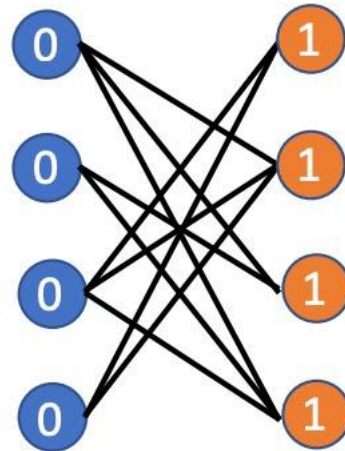
**Theorem 1** (Number of aggregators needed). *In order to discriminate between multisets of size  $n$  whose underlying set is  $\mathbb{R}$ , at least  $n$  aggregators are needed.*

# Aggregation strategy in GNNs

- Homophily assumption
  - Connected nodes are similar/related/informative

# Aggregation strategy in GNNs

- Homophily assumption
  - Connected nodes are similar/related/informative
- How can we deal with heterophilous networks?[21,22]
  - Connected nodes have different class labels and dissimilar features

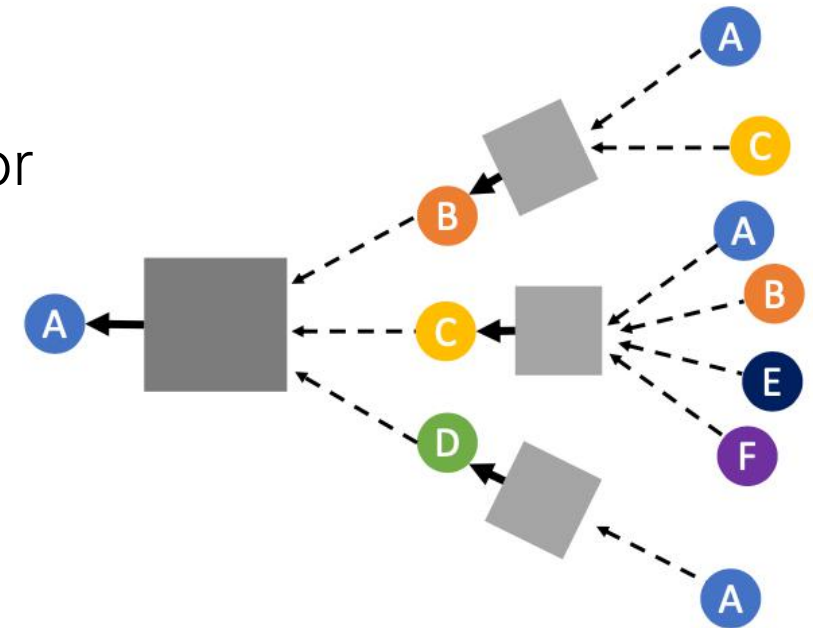


[21] Jiong Zhu., et al. "Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs"

[22] Yao Ma, et al. "IS HOMOPHILY A NECESSITY FOR GRAPH NEURAL NETWORKS?"

# Graph Neural Network Architectures

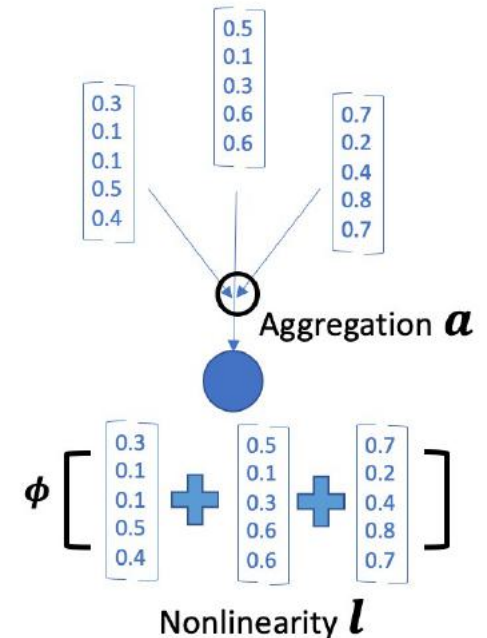
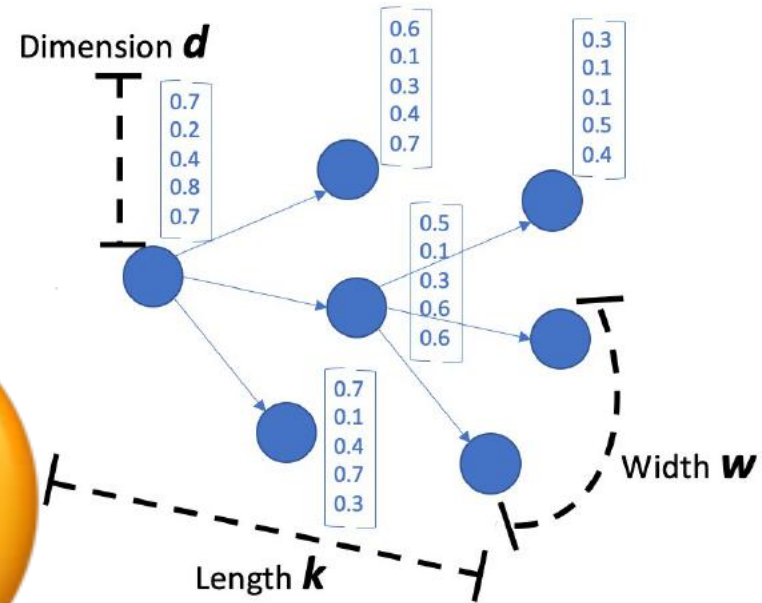
- Width
  - Which neighbors should we aggregate messages from?
- Depth
  - How many hops should we check?
- Aggregation
  - How should we aggregate messages from neighbor



# Neural Architecture Search for GNNs

- Which width, depth, and aggregation strategy are proper for a given graph and task?

Width?  
Depth?  
Aggregation?





# Neural Architecture Search for GNNs

- Finding proper width, depth, and aggregation strategy for a given graph and task **automatically**<sup>[1,2,3]</sup>

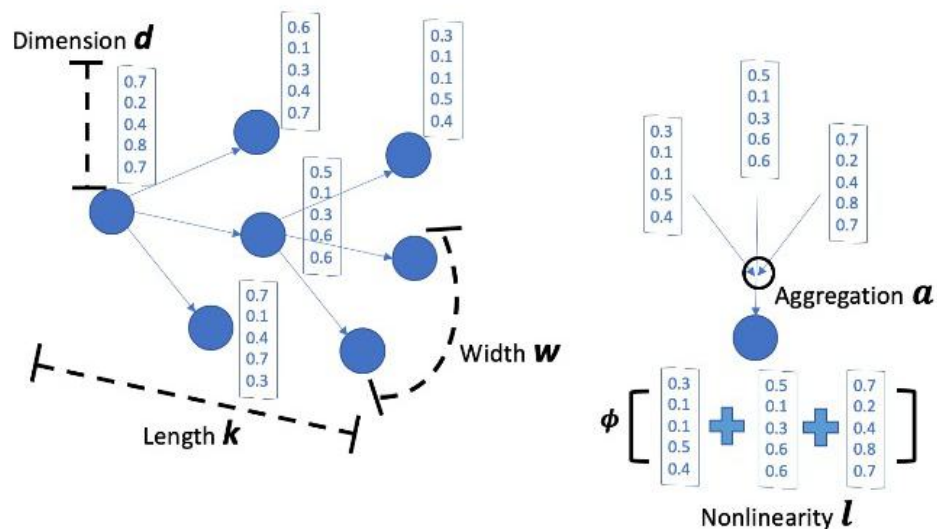
Here is the GNN you requested



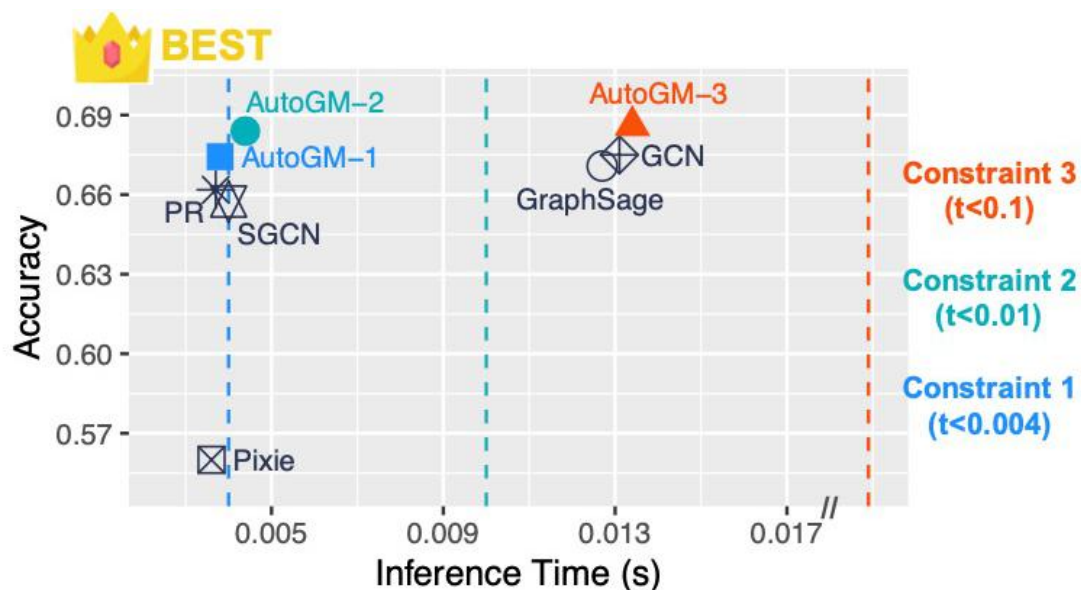
[23] Minji Yoon., et al. "Autonomous Graph Mining Algorithm Search with Best Speed/Accuracy Trade-off" [24] Kaixiong Zhou, et al. "Auto-GNN: Neural Architecture Search of Graph Neural Networks"  
[25] Yang Gao, et al. "GraphNAS: Graph Neural Architecture Search with Reinforcement Learning"

# Neural Architecture Search for GNNs

- AutoGM<sup>[23]</sup>



**Step 1: define a hyperparameter space**



**Step 2: explore the space efficiently**

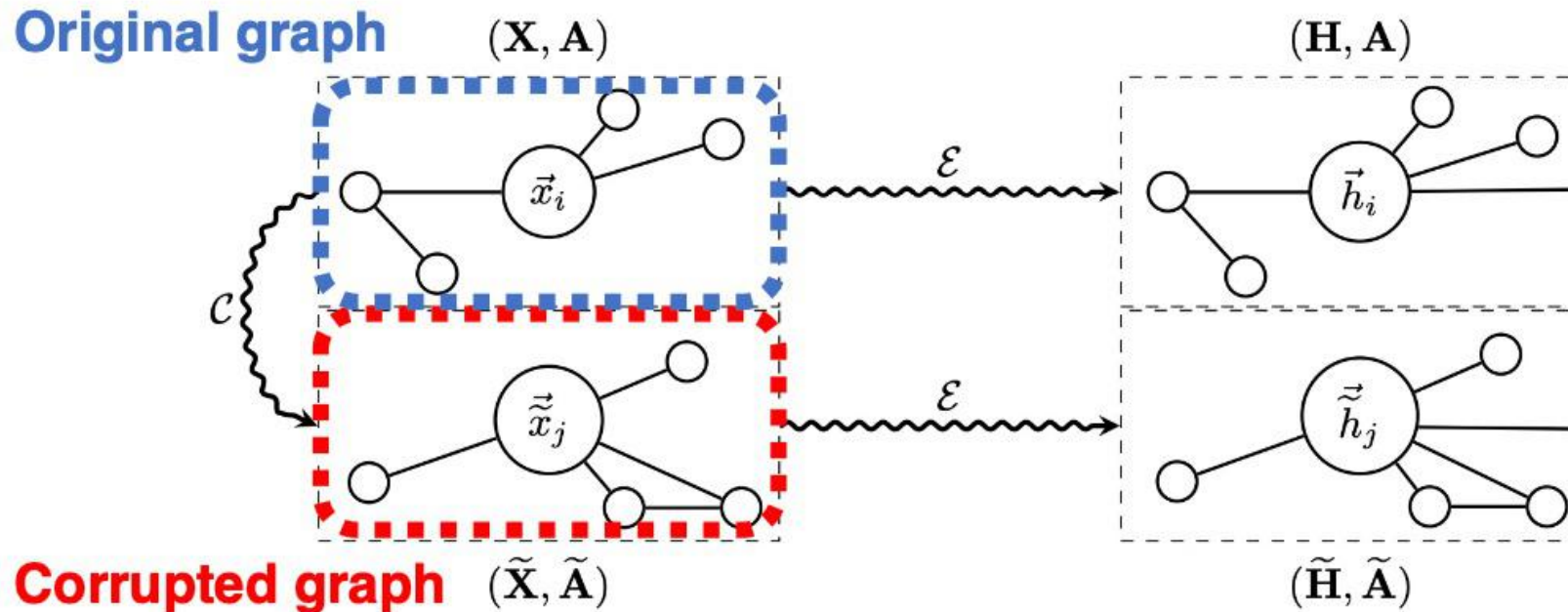
# How to train GNNs?

# How to train GNNs

- **Semi-supervised learning**
  - Input node features are given for all nodes in a graph
  - Only a subset of nodes have labels

# How to train GNNs

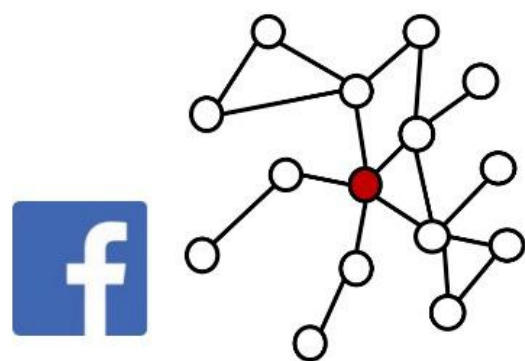
- Unsupervised learning<sup>[26]</sup>
  - Contrastive learning



# How to train GNNs

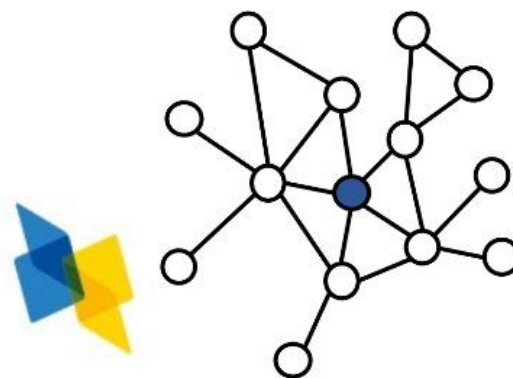
- Transfer learning

- Transfer a pre-trained GNN model between graphs[27]



**Facebook network**

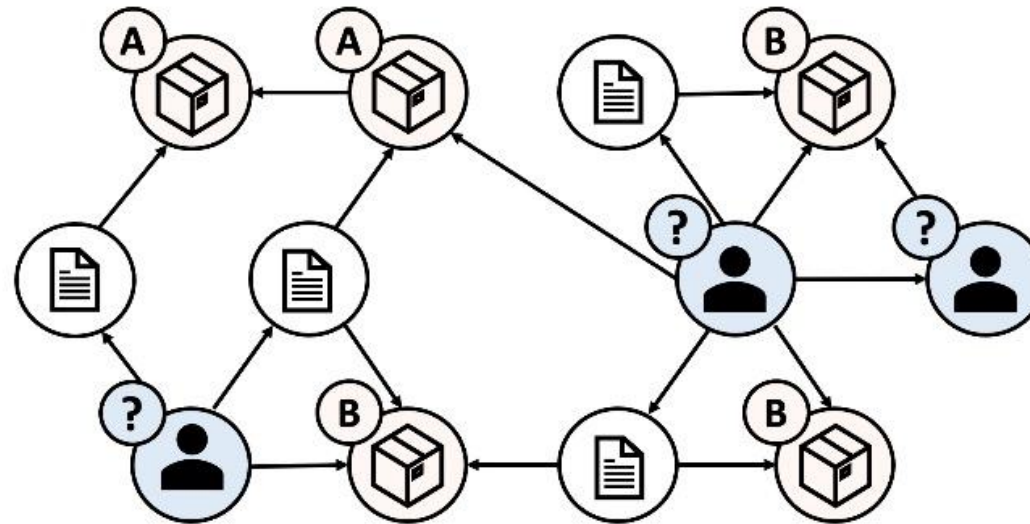
Pre-trained GNN  $f$



**DBLP co-authorship network**

# How to train GNNs

- Transfer learning
  - Transfer between different node types across a **heterogeneous graph**[28]

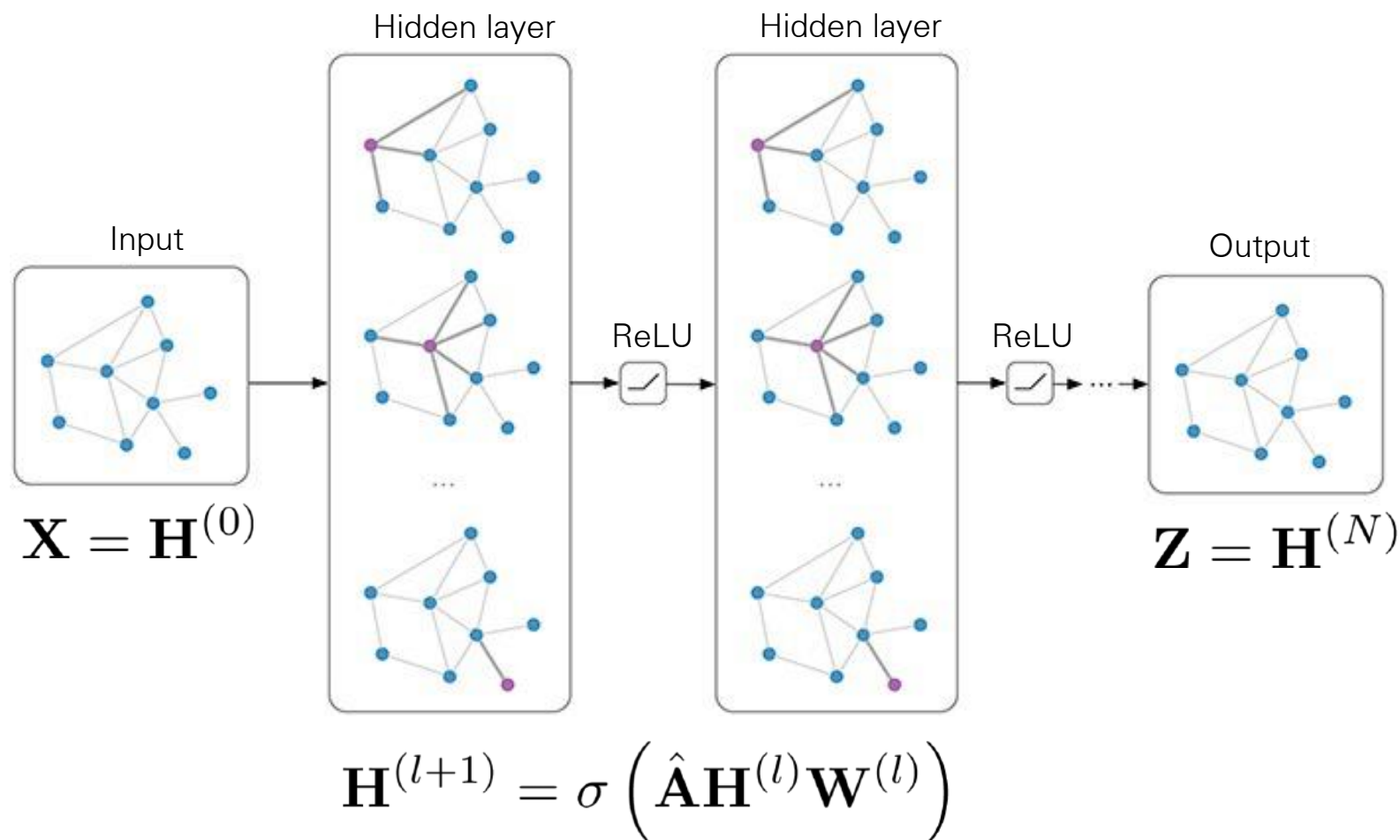


# Applications to "classical" network problems



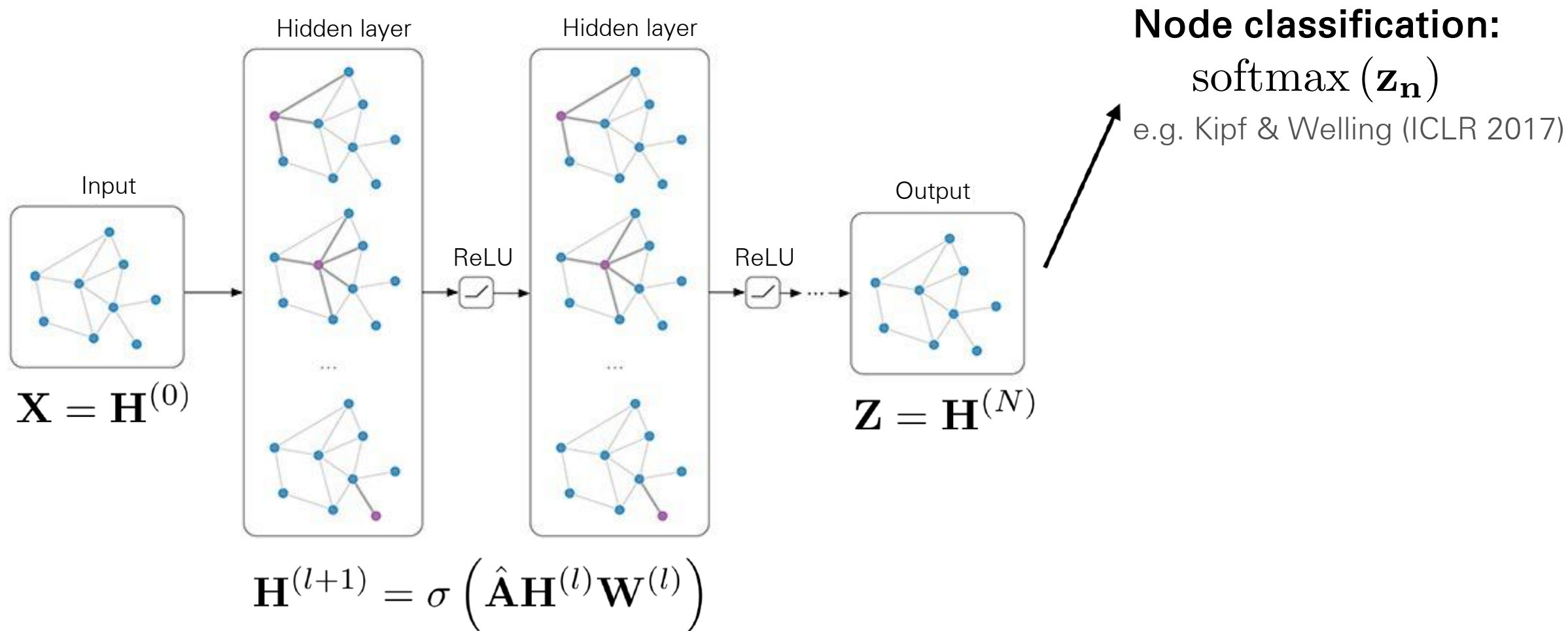
# One fits all: Classification and link prediction with GNNs/GCNs

Input: Feature matrix  $\mathbf{X} \in \mathbb{R}^{N \times E}$ , preprocessed adjacency matrix  $\hat{\mathbf{A}}$



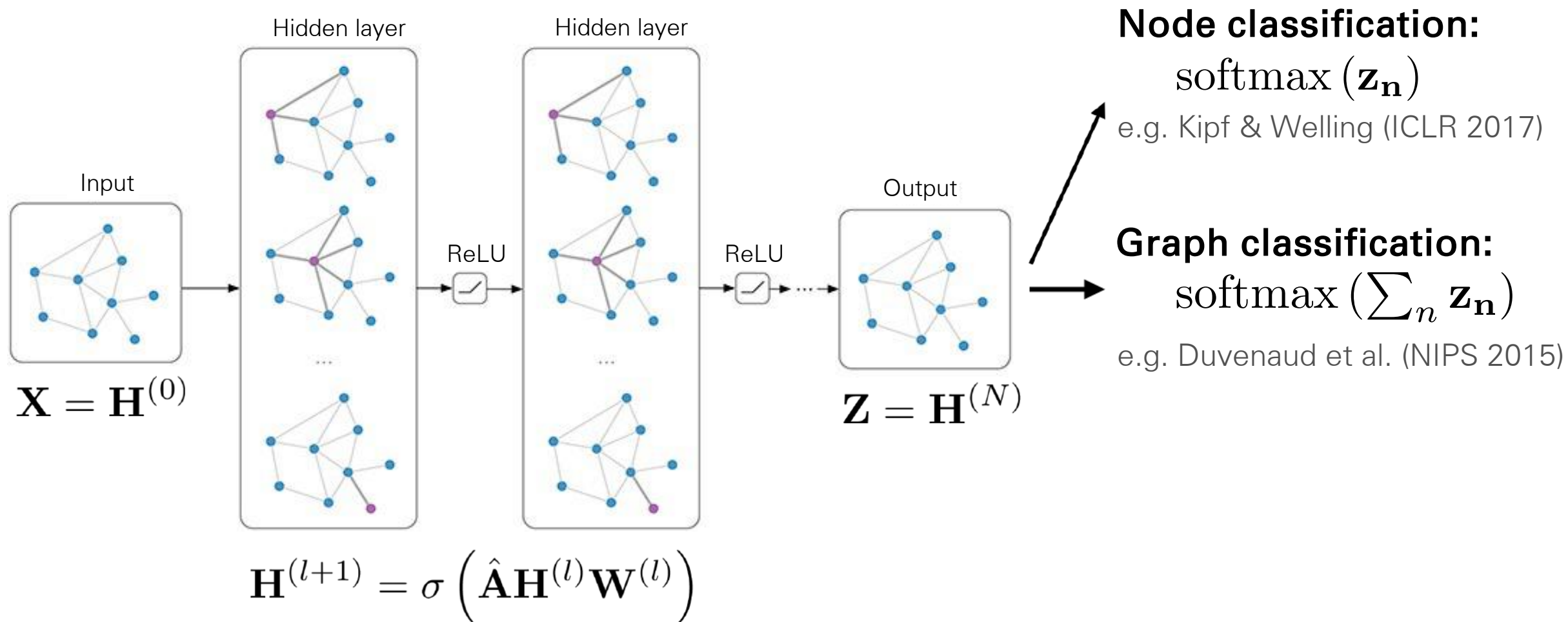
# One fits all: Classification and link prediction with GNNs/GCNs

Input: Feature matrix  $\mathbf{X} \in \mathbb{R}^{N \times E}$ , preprocessed adjacency matrix  $\hat{\mathbf{A}}$



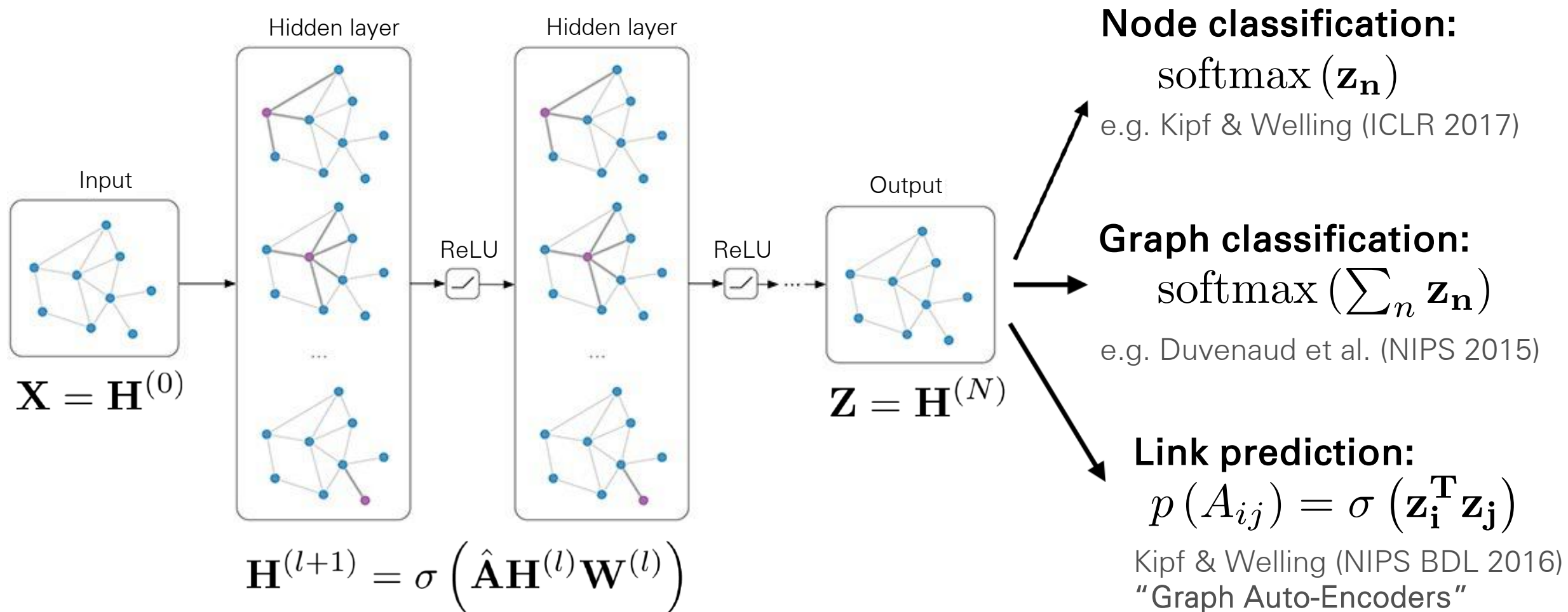
# One fits all: Classification and link prediction with GNNs/GCNs

Input: Feature matrix  $\mathbf{X} \in \mathbb{R}^{N \times E}$ , preprocessed adjacency matrix  $\hat{\mathbf{A}}$



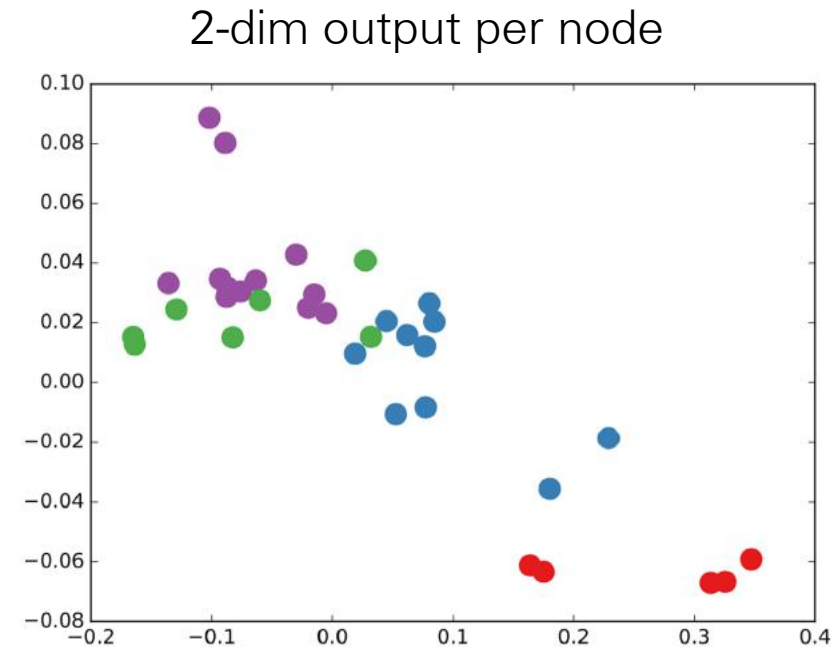
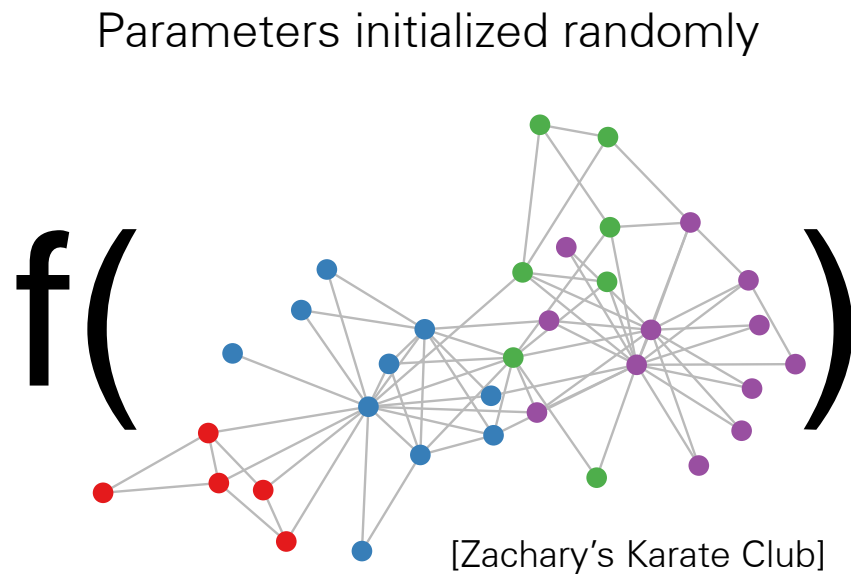
# One fits all: Classification and link prediction with GNNs/GCNs

Input: Feature matrix  $\mathbf{X} \in \mathbb{R}^{N \times E}$ , preprocessed adjacency matrix  $\hat{\mathbf{A}}$



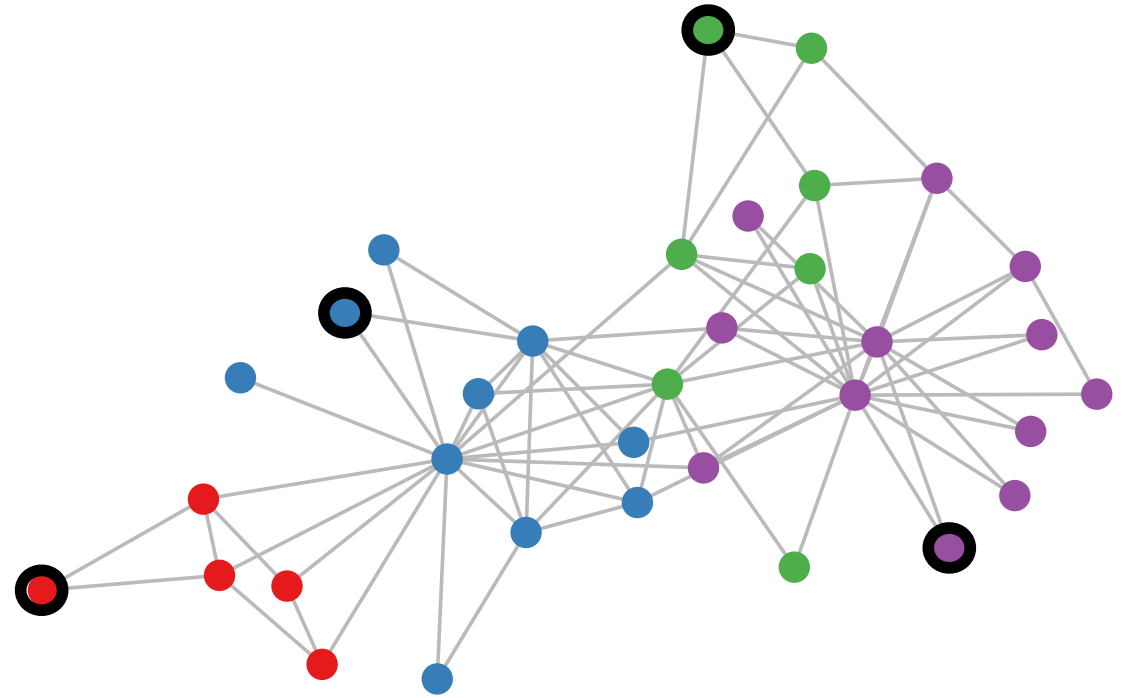
# What do learned representations look like?

Forward pass through untrained 3-layer GCN model



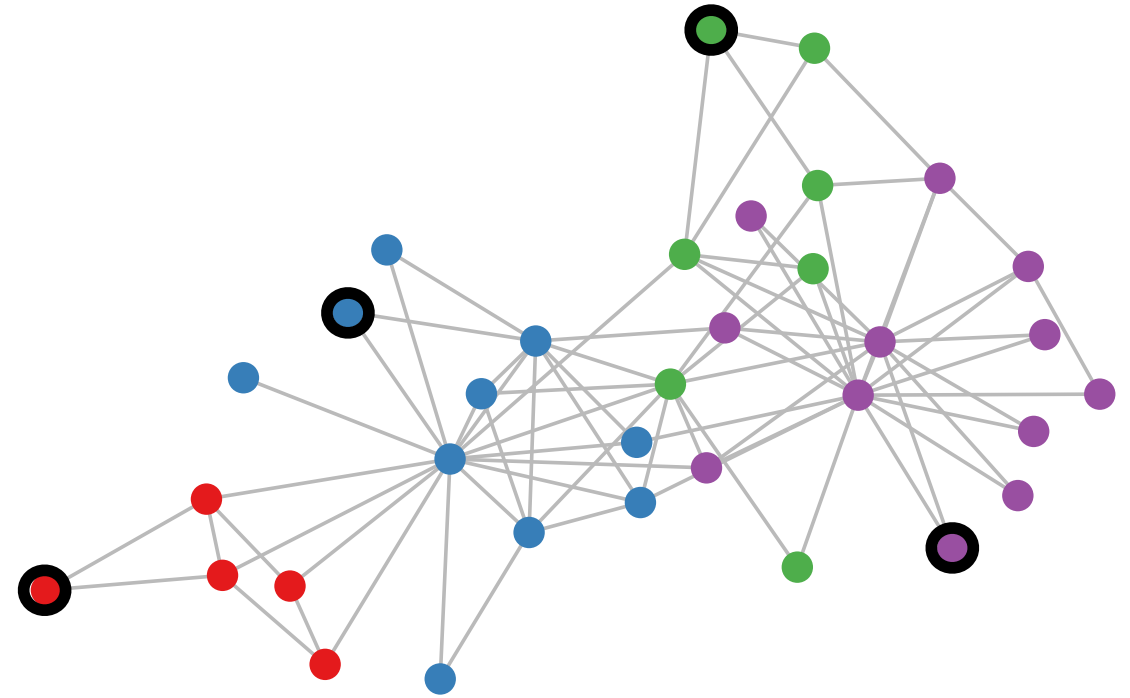
# Semi-supervised classification on graphs

- **Setting:**  
Some nodes are labeled (black circle)  
All other nodes are unlabeled
- **Task:**  
Predict node label of unlabeled nodes



# Semi-supervised classification on graphs

- **Setting:**  
Some nodes are labeled (black circle)  
All other nodes are unlabeled
- **Task:**  
Predict node label of unlabeled nodes



Evaluate loss on labeled nodes only:

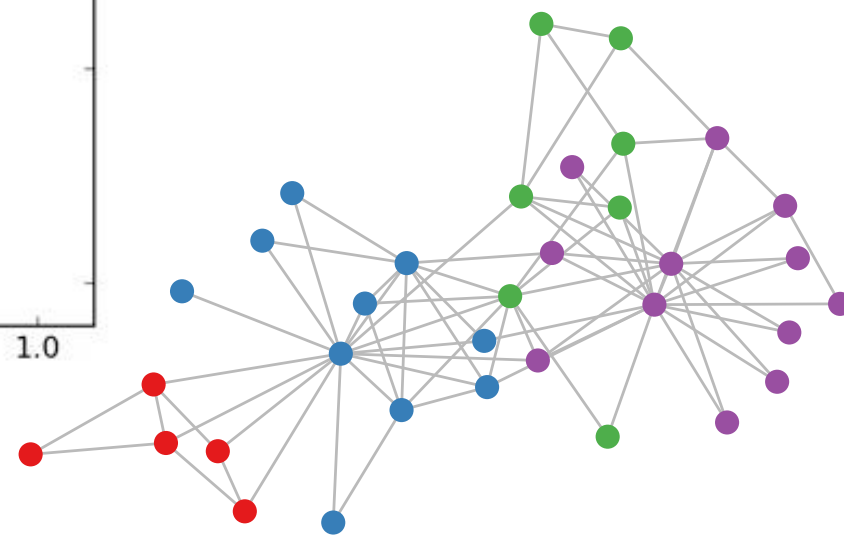
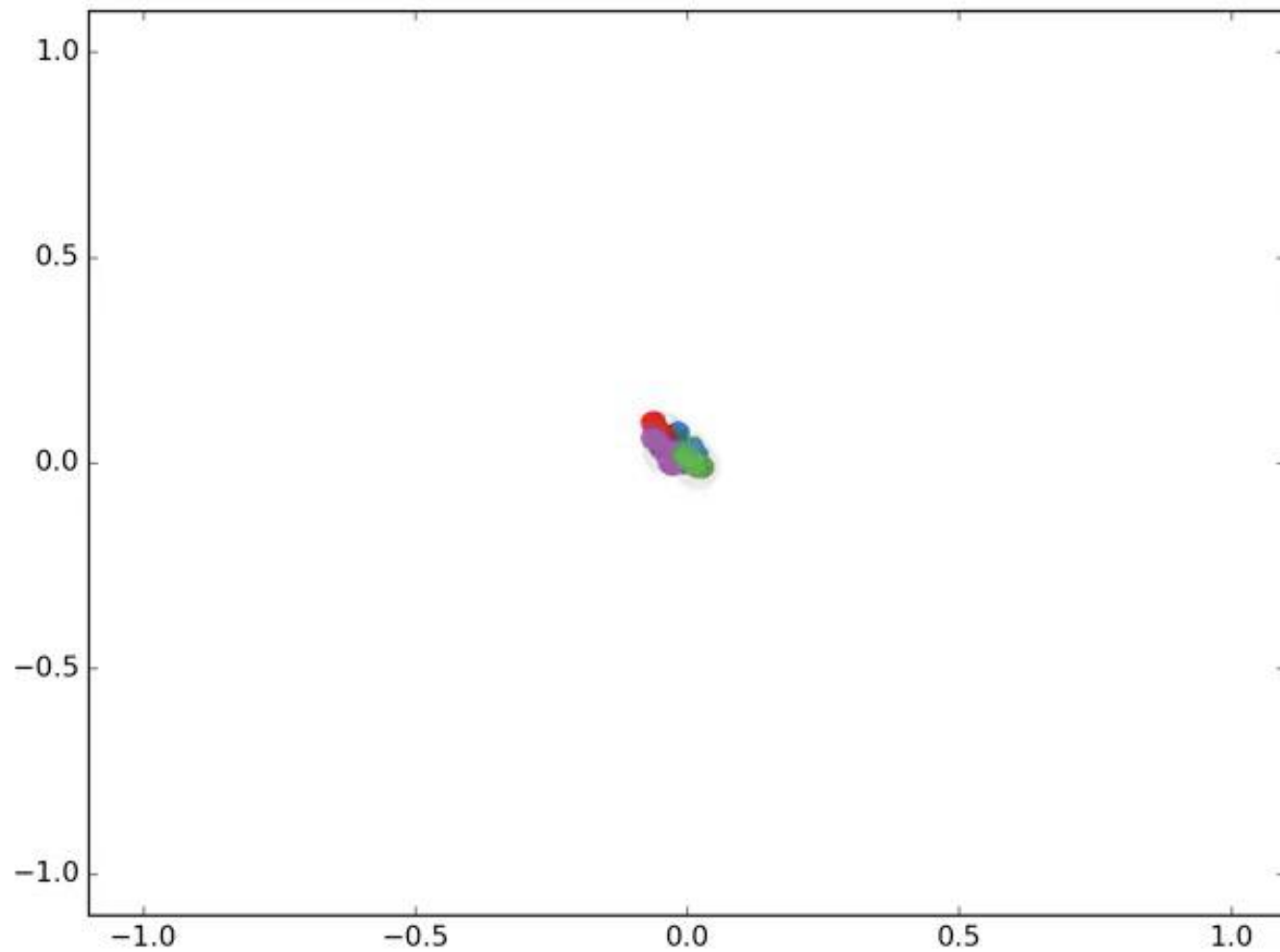
$$\mathcal{L} = - \sum_{l \in \mathcal{Y}_L} \sum_{f=1}^F Y_{lf} \ln Z_{lf}$$

$\mathcal{Y}_L$  set of labeled node indices

$\mathbf{Y}$  label matrix

$\mathbf{Z}$  GCN output (after softmax)

# Toy example (semi-supervised learning)





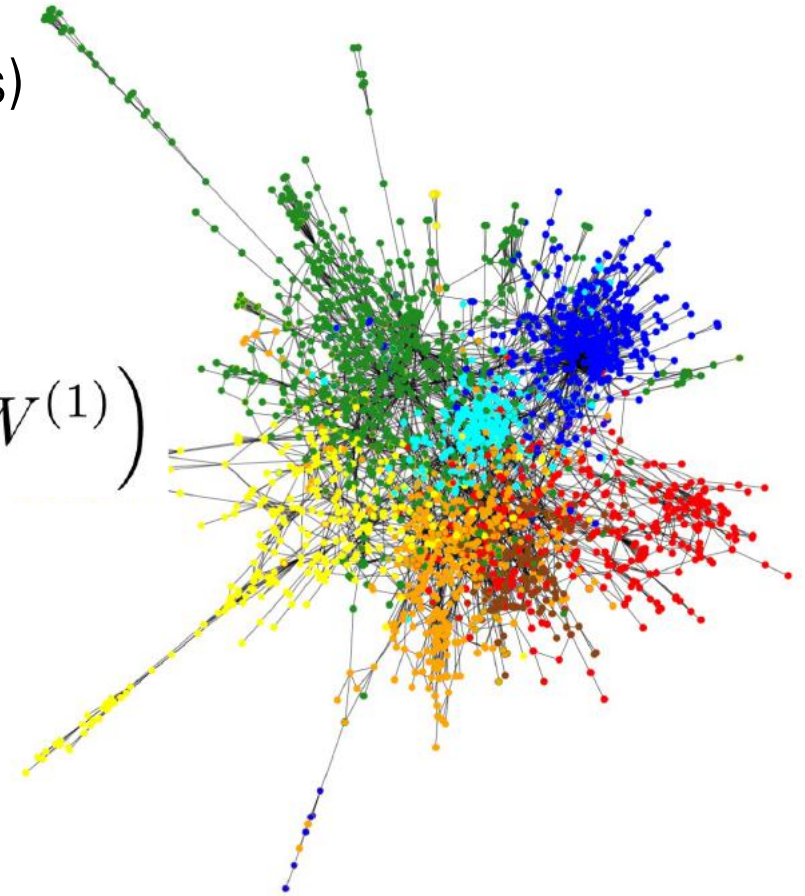
# Application: Classification on citation networks

Input: Citation networks (nodes are papers, edges are citation links, optionally bag-of-words features on nodes)

Target: Paper category (e.g. stat.ML, cs.LG, ...)

Model: 2-layer GCN

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



(Figure from: Bronstein, Bruna, LeCun, Szlam, Vandergheynst, 2016)

# Application: Classification on citation networks

Input: Citation networks (nodes are papers, edges are citation links, optionally bag-of-words features on nodes)

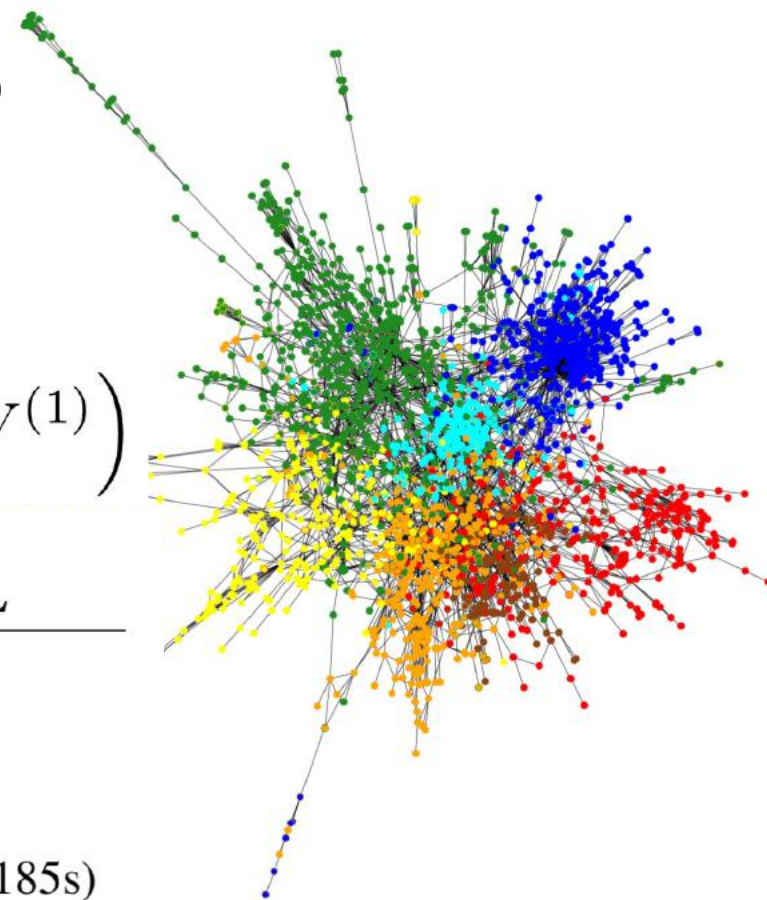
Target: Paper category (e.g. stat.ML, cs.LG, ...)

Model: 2-layer GCN

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

Classification results (accuracy)

	Method	Citeseer	Cora	Pubmed	NELL
	ManiReg [3]	60.1	59.5	70.7	21.8
	SemiEmb [24]	59.6	59.0	71.1	26.7
no input	LP [27]	45.3	68.0	63.0	26.5
features	DeepWalk [18]	43.2	67.2	65.3	58.1
	Planetoid* [25]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
	<b>GCN (this paper)</b>	<b>70.3 (7s)</b>	<b>81.5 (4s)</b>	<b>79.0 (38s)</b>	<b>66.0 (48s)</b>
	GCN (rand. splits)	67.9 ± 0.5	80.1 ± 0.5	78.9 ± 0.7	58.4 ± 1.7



(Figure from: Bronstein, Bruna, LeCun, Szlam, Vandergheynst, 2016)

# Still many open problems..

- And many more chances to do groundbreaking research
- ex) other graph formats
  - 3-dimensional graphs
  - Temporal graphs
  - ...

**Next Lecture:**  
Autoencoders and  
Autoregressive Models