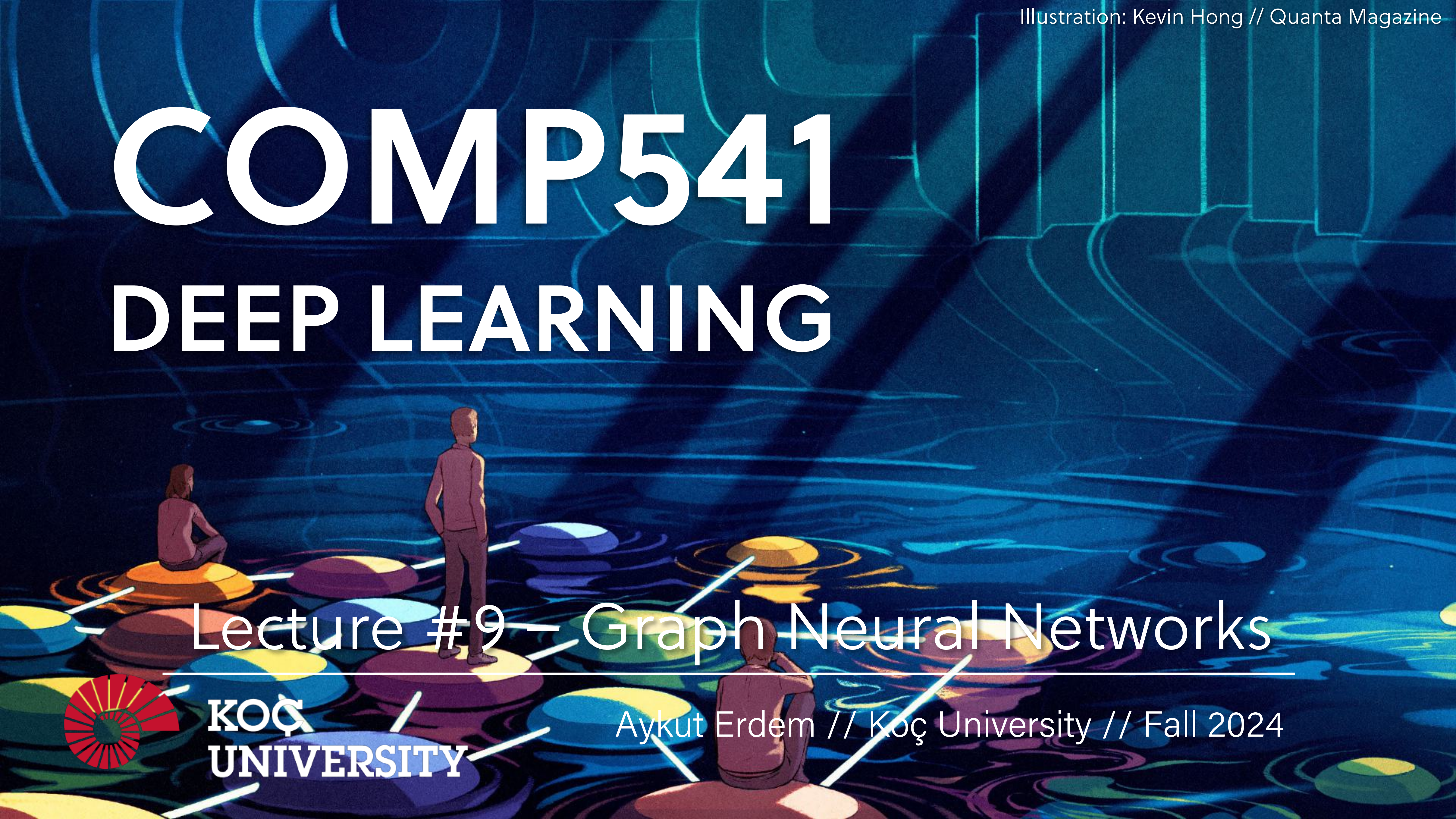


Illustration: Kevin Hong // Quanta Magazine

COMP541
DEEP LEARNING

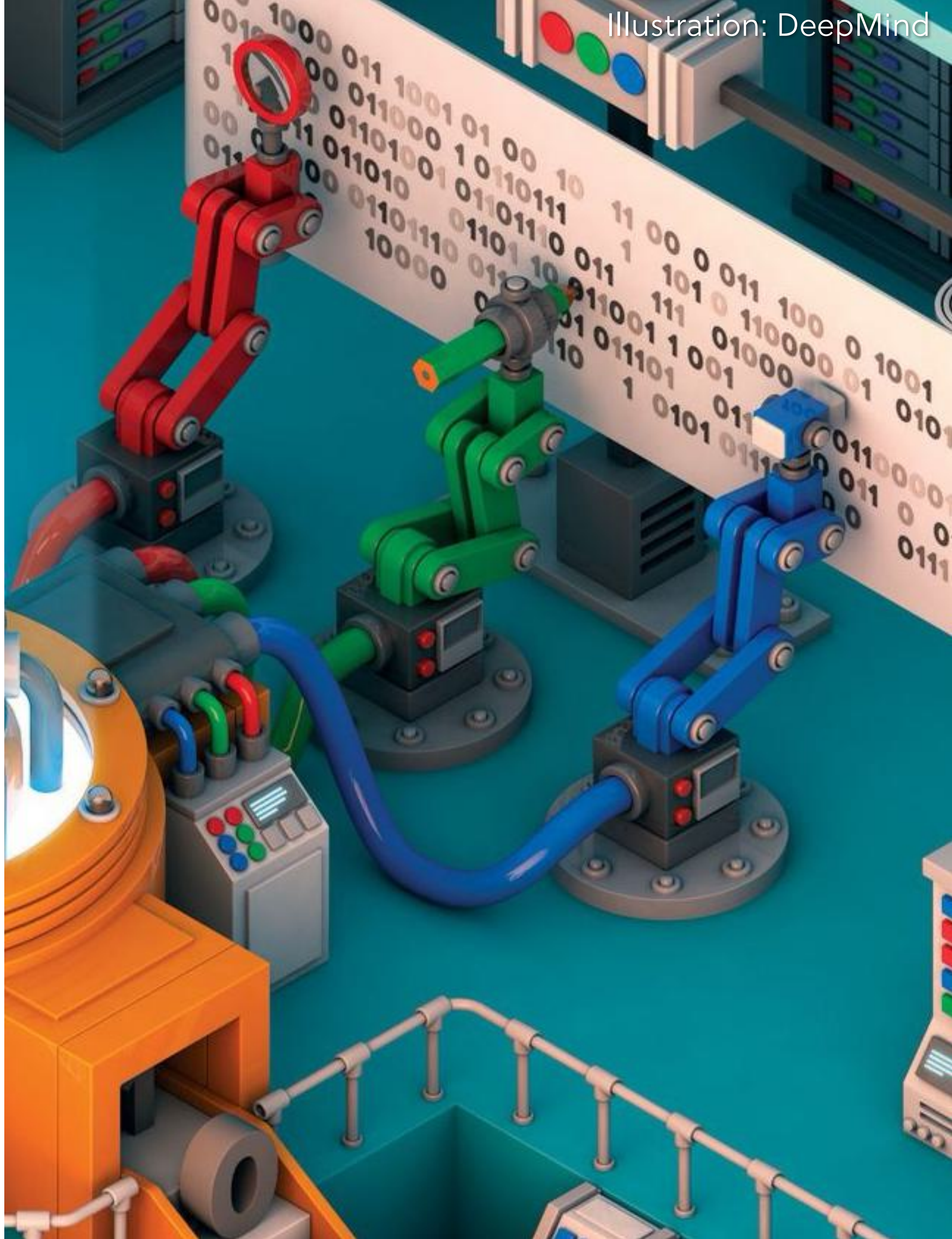
Lecture #9 – Graph Neural Networks

KOÇ UNIVERSITY

Aykut Erdem // Koç University // Fall 2024

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

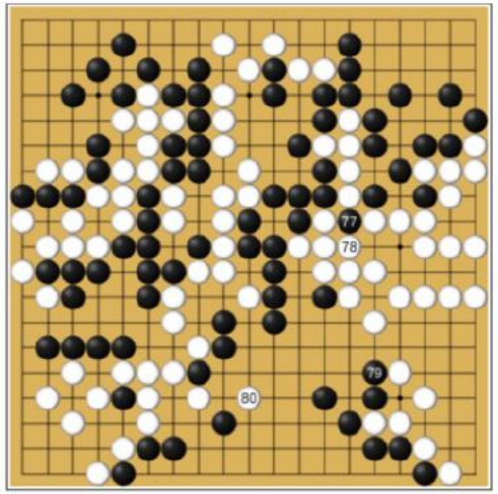
- **Disclaimer:** Much of the material and slides for this lecture were borrowed from
  —Yujia Li and Oriol Vinyals' tutorial on Graph Nets
  —Thomas Kipf’s talk on structured deep models: deep Learning on graphs and beyond
  —Minji Yoon’s CMU 10707 slides

Deep Learning

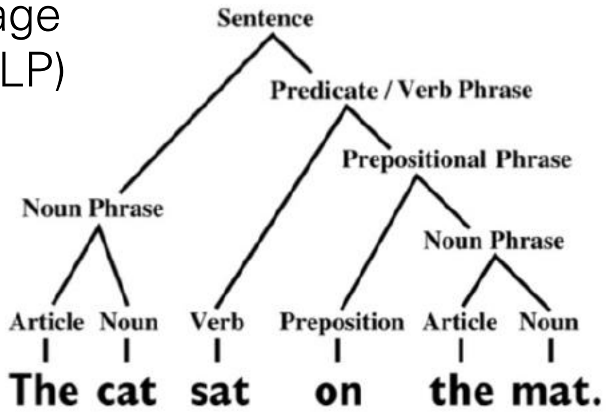
IMAGENET

Speech data

Grid games



Natural language processing (NLP)



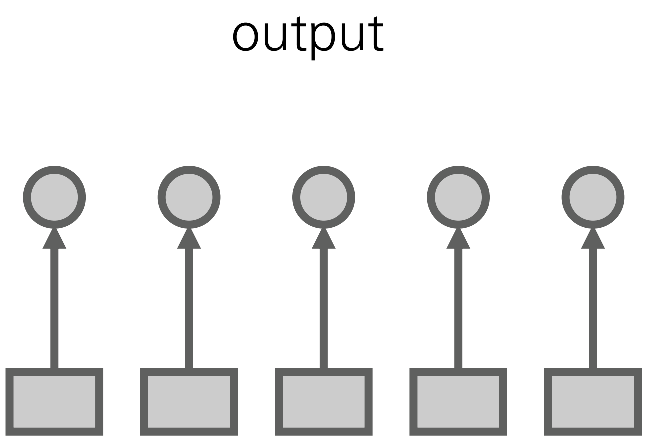
...

Deep neural nets that exploit:

- translation equivariance (weight sharing)
- hierarchical compositionality

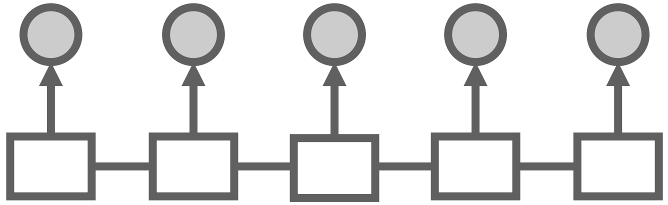
Modeling Structured Data

Unstructured Data

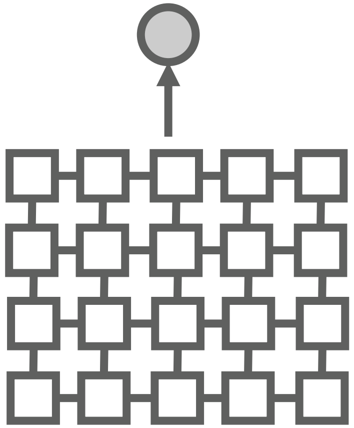


Data with Rigid Structure

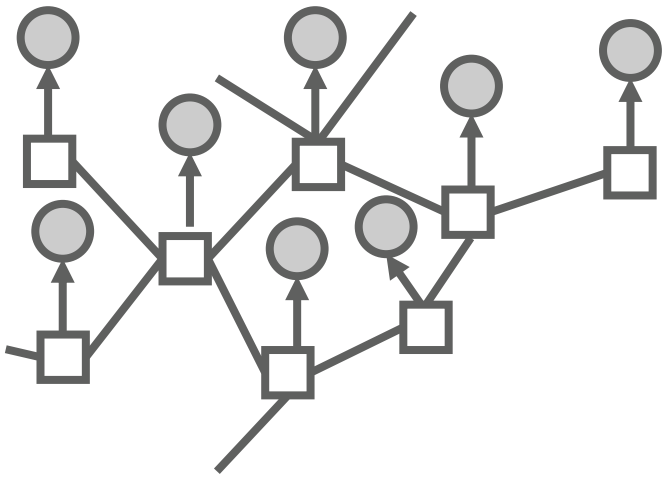
sequences



visual data



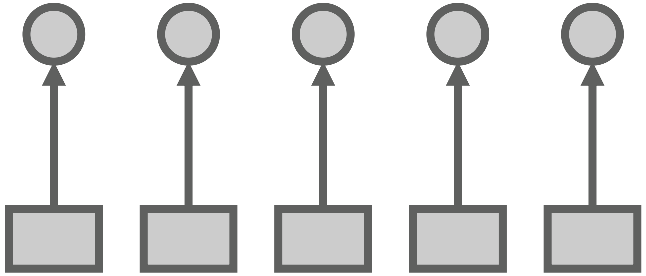
Graph Structured Data



Modeling Structured Data

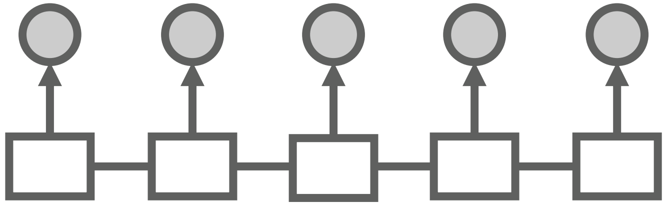
Unstructured Data

output

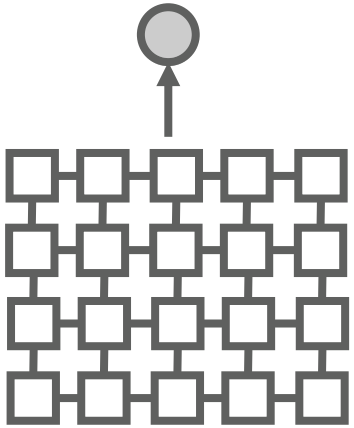


Data with Rigid Structure

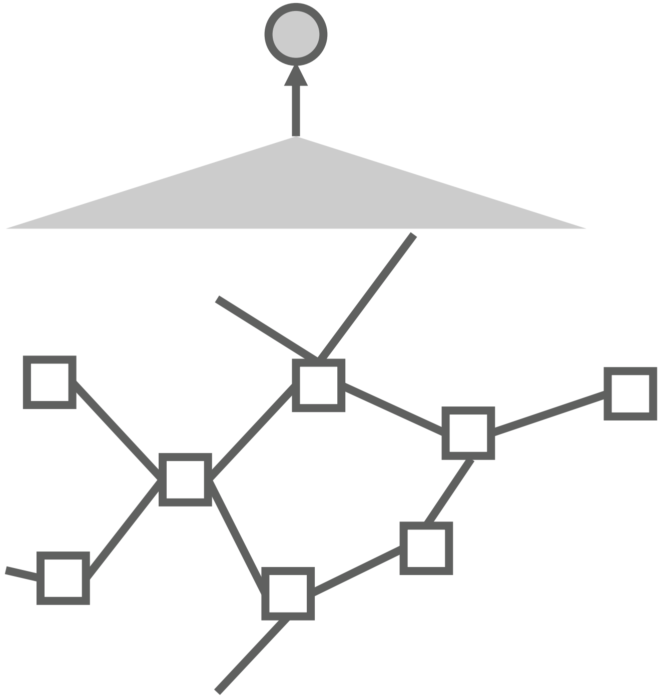
sequences



visual data



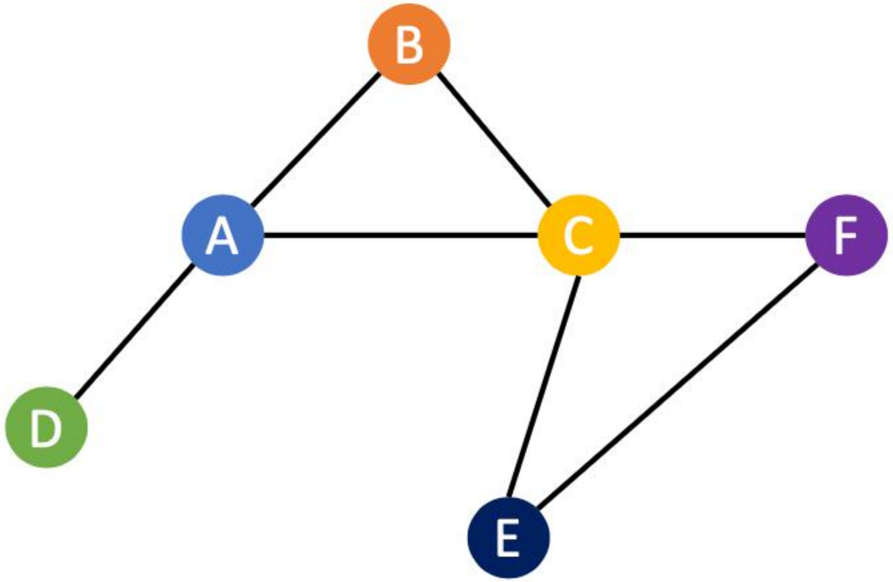
Graph Structured Data



What is a graph

- A graph is composed of
  - **Nodes** (also called vertices)
  - **Edges** connecting a pair of nodes

presented in an **adjacency matrix**

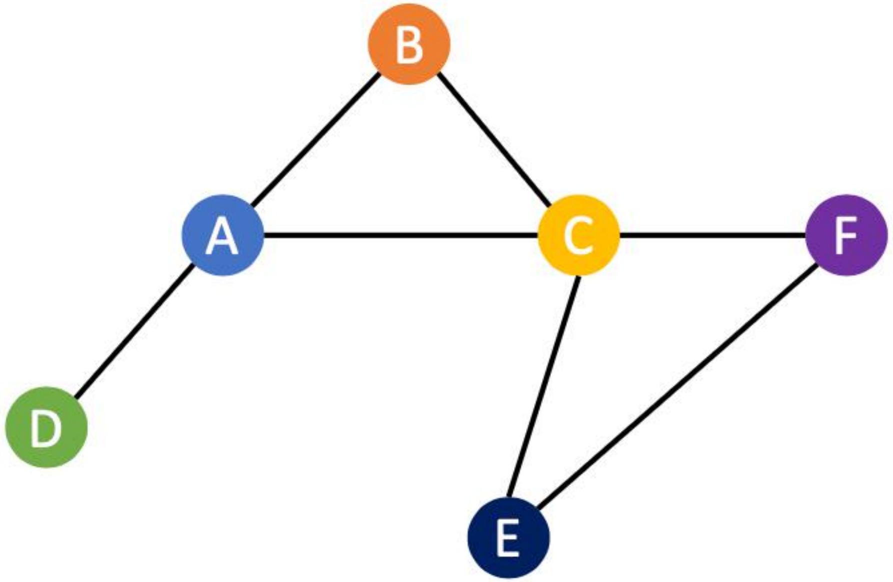


	A	B	C	D	E	F
A		1	1	1		
B	1		1			
C	1	1			1	1
D	1					
E			1			1
F			1		1	

What is a graph

- A graph is composed of
  - **Nodes** (also called vertices)
  - **Edges** connecting a pair of nodes

  presented in an **adjacency matrix**
- Nodes can have **feature vectors**

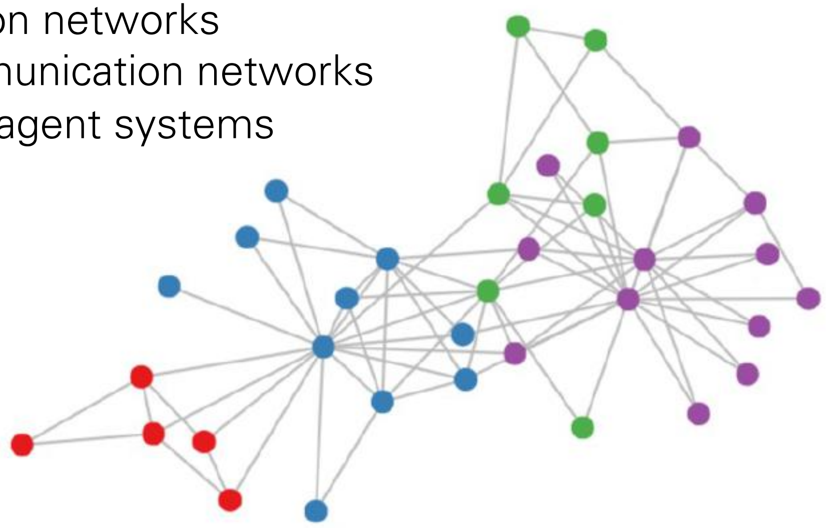


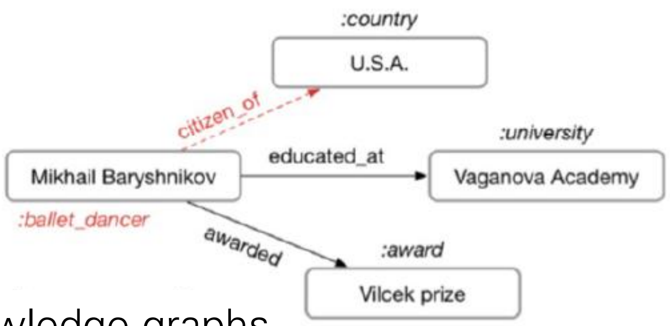
A	X_A
B	X_B
C	X_C
D	X_D
E	X_E
F	X_F

Graph structured data

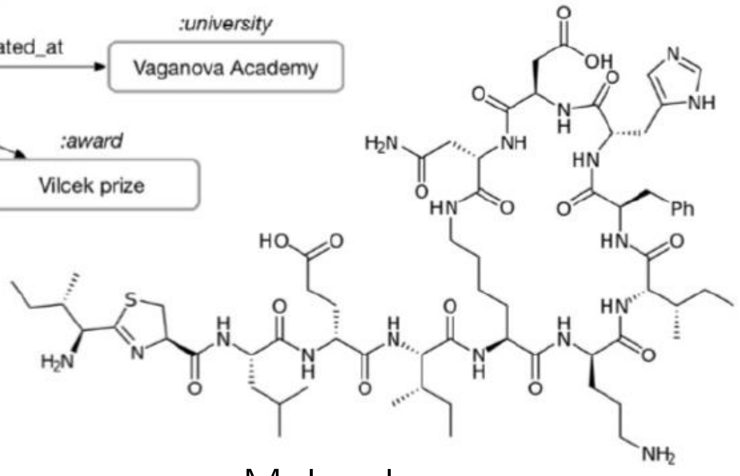
- A lot of real-world data does not “live” on grids

Social networks
Citation networks
Communication networks
Multi-agent systems

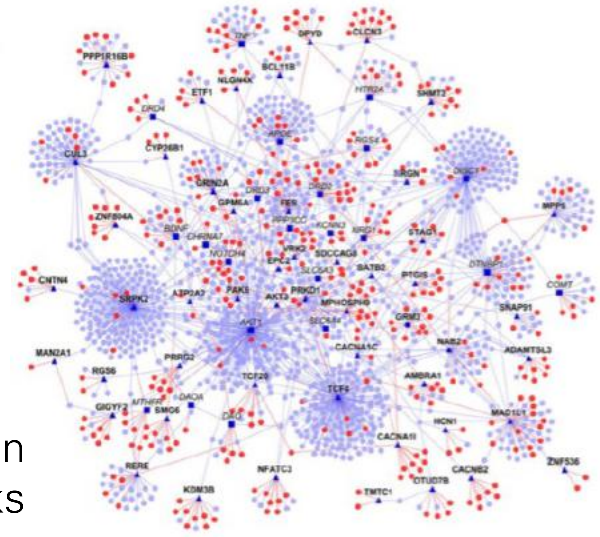




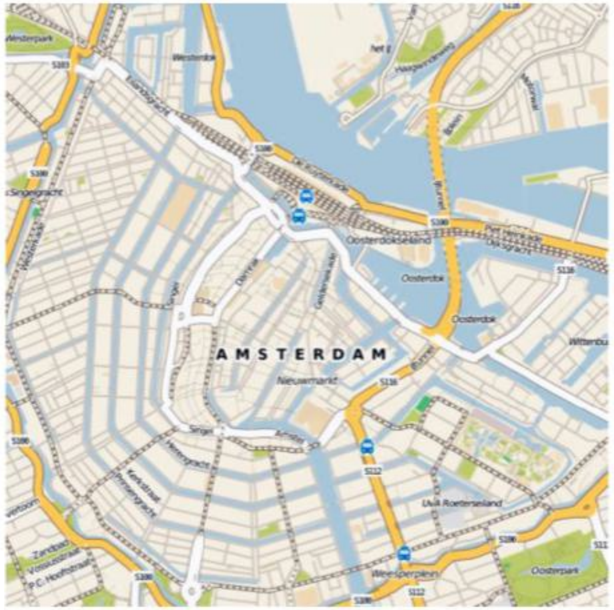
Knowledge graphs



Molecules



Protein interaction networks



Road maps

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

Graph Neural Networks have a large impact on…

DeepMind > Blog > Traffic prediction with advanced Graph Neural Networks

BLOG POST
RESEARCH
03 SEP 2020

Traffic prediction with advanced Graph Neural Networks

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
nking in cross-modal retrieval.

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

Ankit Jain, Isaac Liu, Ankur Sarda, and Piero Molino December 4, 2019
0

amazon | 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[1], Kuan Wang[1], Jiacheng Yang[1], Linxiao Shen[2], Nan Sun[2], Hae-Seung Lee[1], Song Han[1]
[1]Massachusetts Institute of Technology
[2]UT Austin

HAN LAB Hardware, AI and Neural-nets
TEXAS

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

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

September 24, 2020 | Zack Savitsky

Share Tweet Email

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.

nature

View all journals Search Login

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

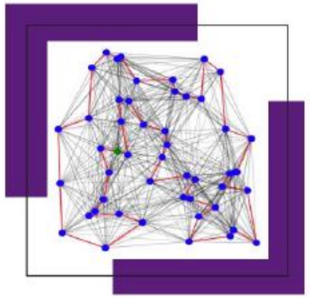
DeepMind's AI helps untangle the mathematics of knots

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

ipam institute for pure & applied mathematics

Deep Learning and Combinatorial Optimization

February 22 - 25, 2021



Patterns

CellPress OPEN ACCESS

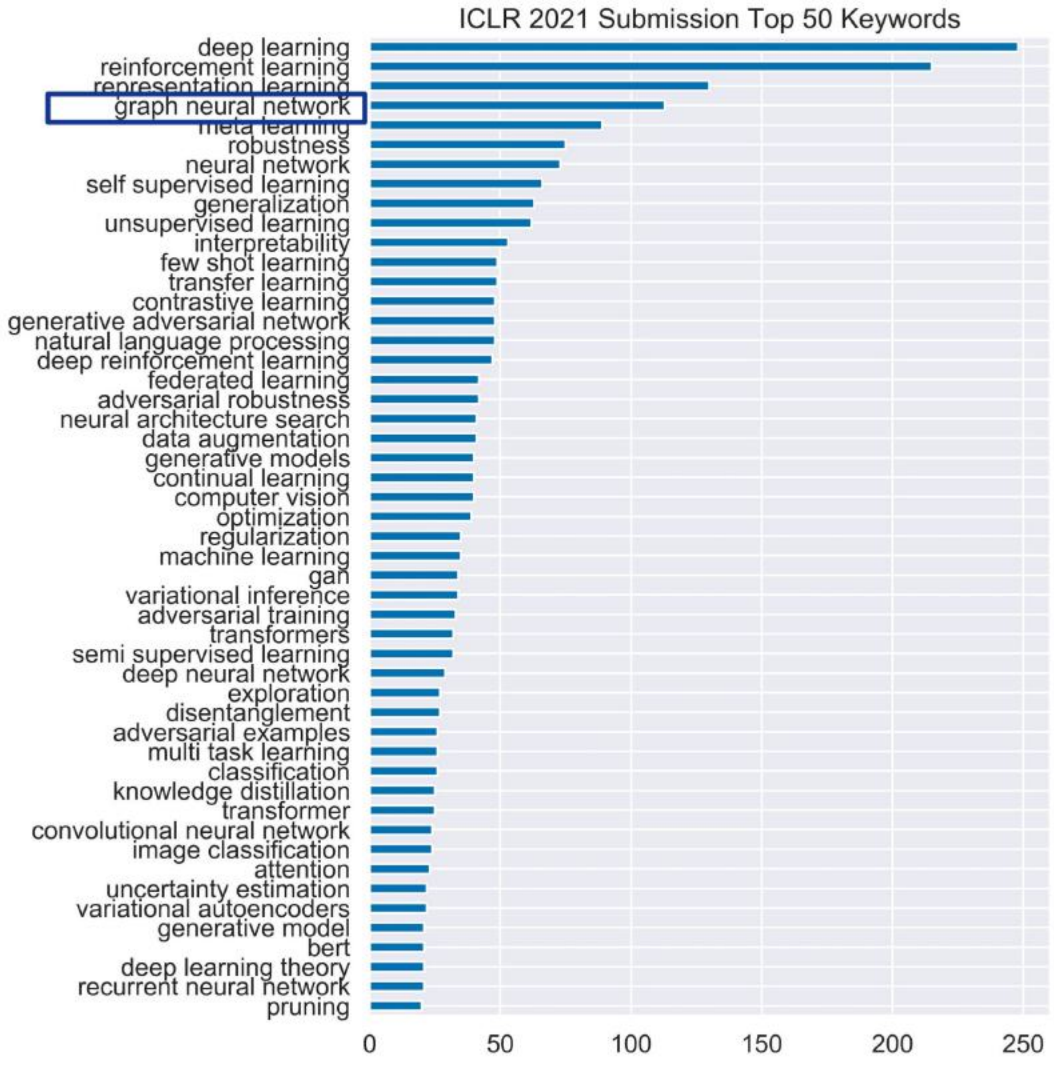
Opinion

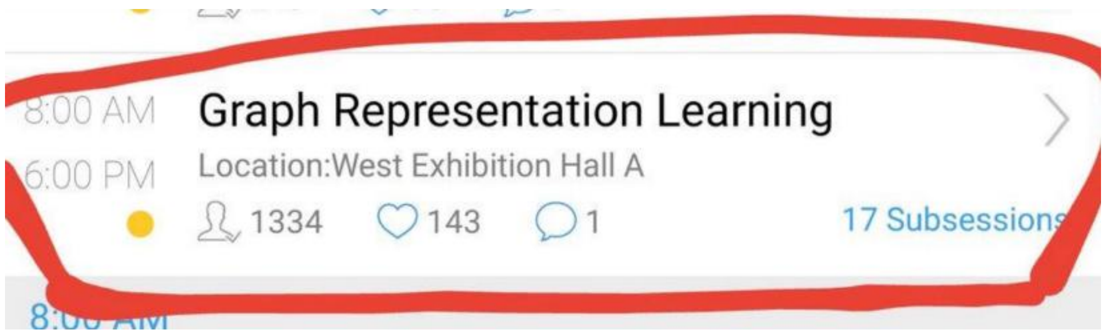
Neural algorithmic reasoning

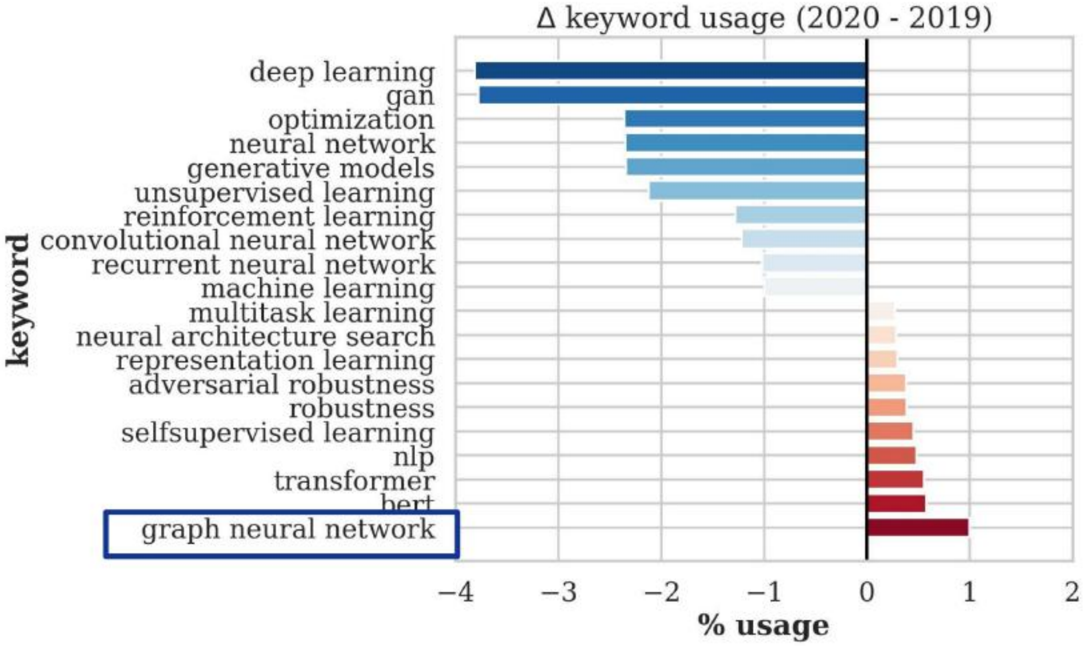
Petar Veličković[1,*] and Charles Blundell[1]
[1]DeepMind, London, Greater London, UK
*Correspondence: petarv@google.com
https://doi.org/10.1016/j.patter.2021.100273

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.

A very hot research topic







Slide credit: P. Veličković

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**

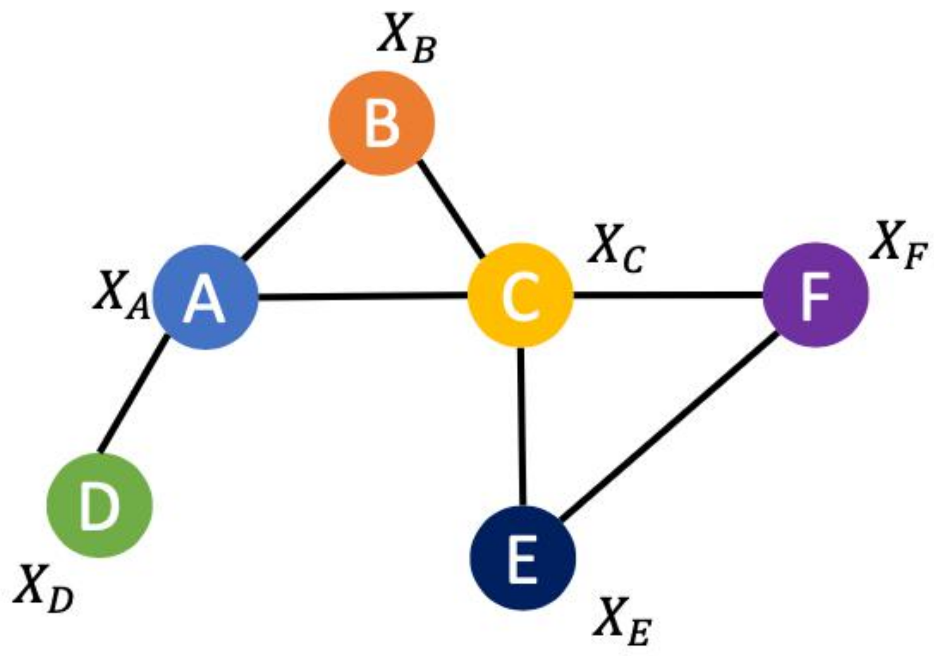
- Handle arbitrary node ordering
  Requires: **A model invariant to node permutations**

- Utilize graph structure
  Requires: **A mechanism to communicate information on graphs**

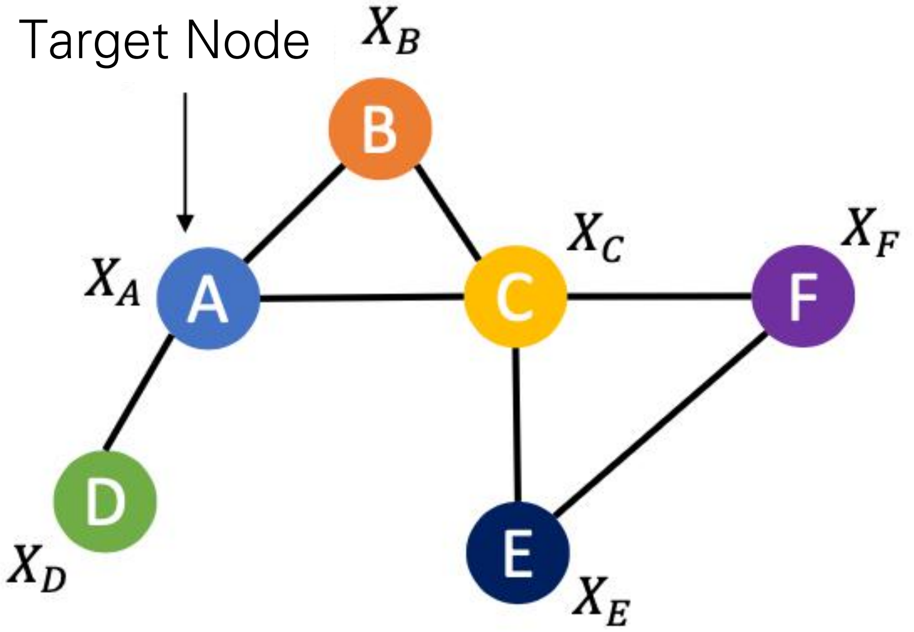
What is Graph Neural Network?

Problem definition

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

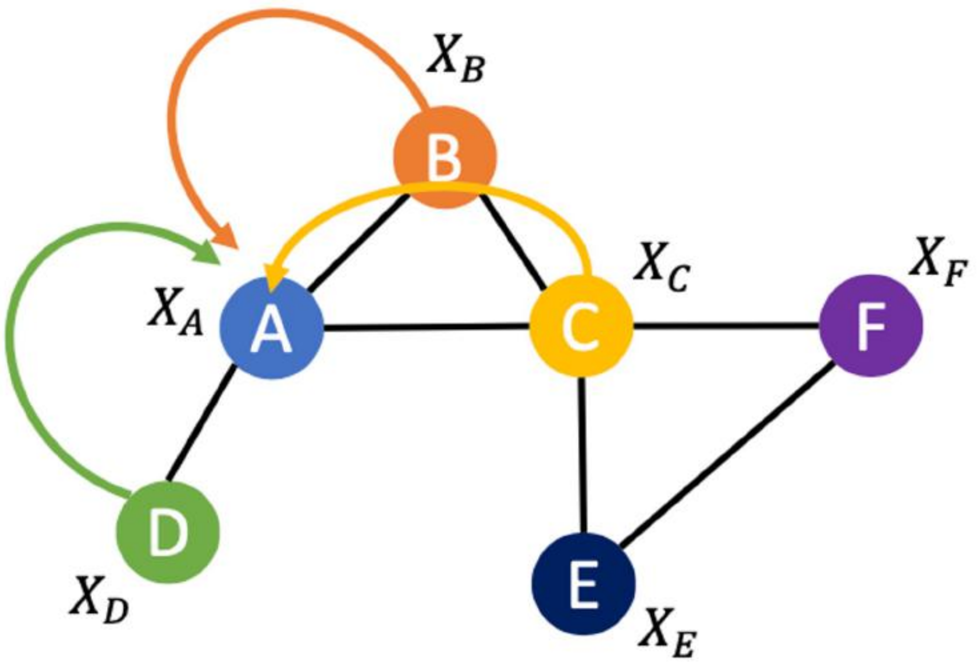


Graph Neural Networks (GNNs)



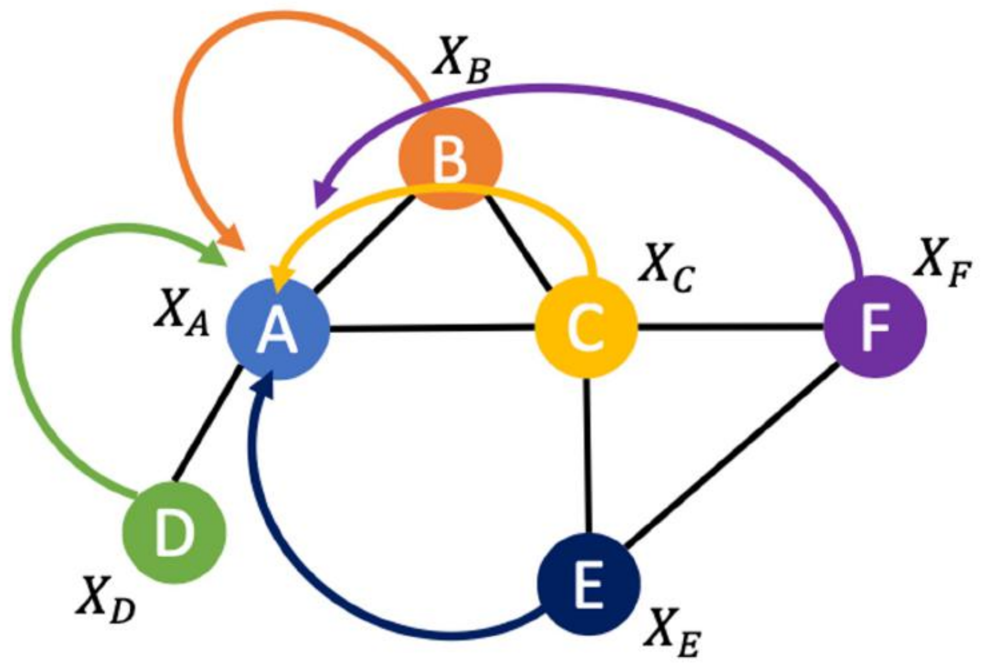
"Homophily: connected nodes are related/informative/similar"

Graph Neural Networks (GNNs)



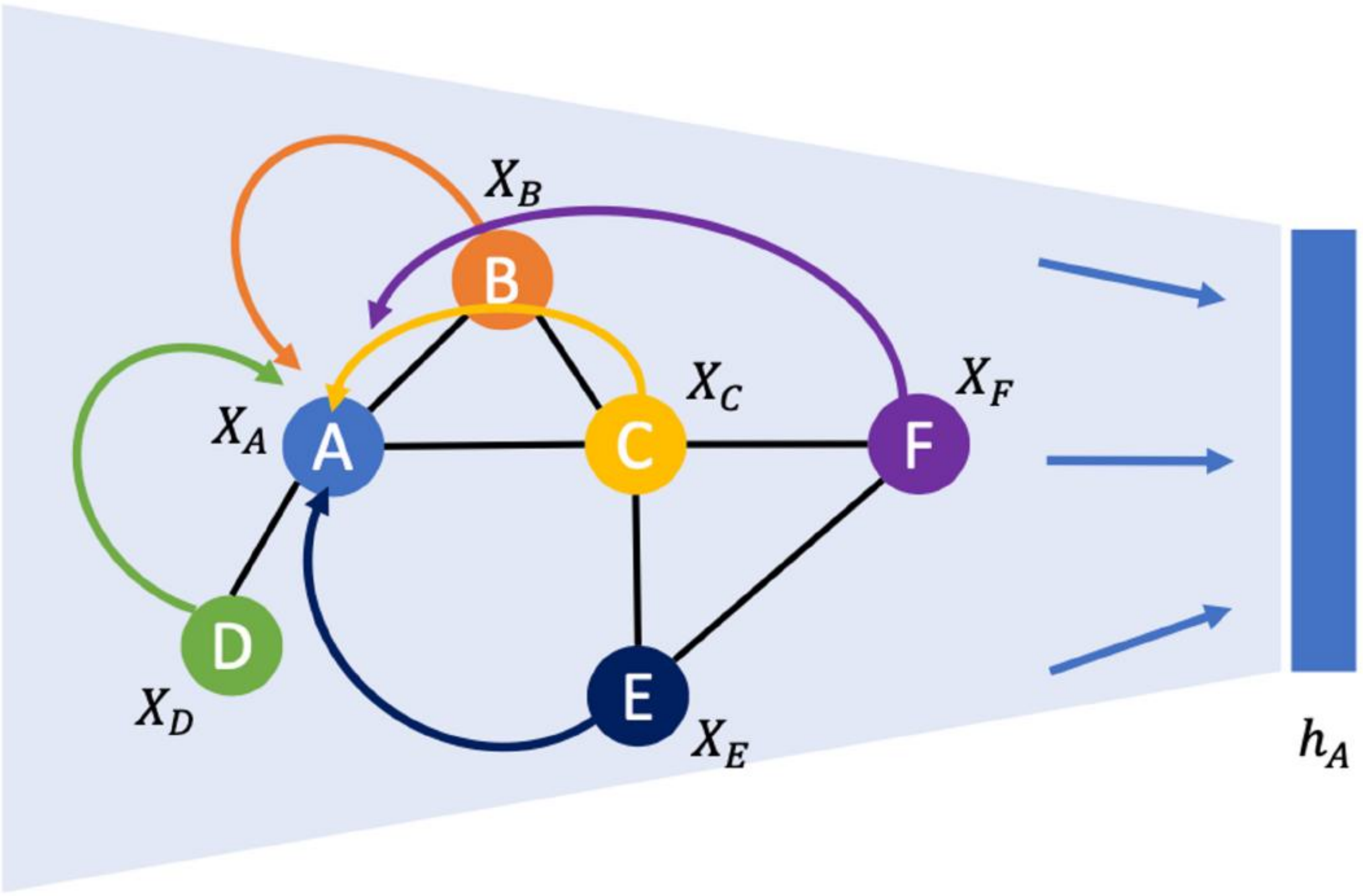
"Homophily: connected nodes are related/informative/similar"

Graph Neural Networks (GNNs)

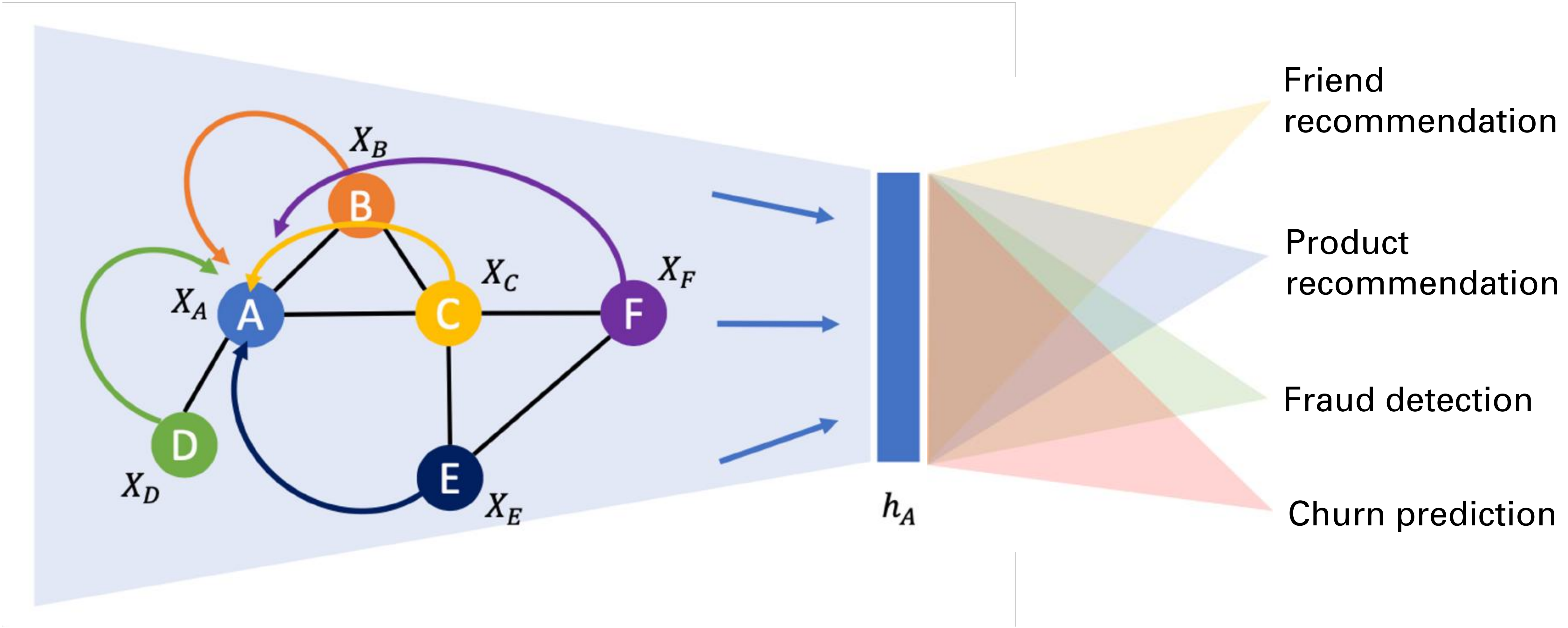


"Homophily: connected nodes
are
related/informative/similar"

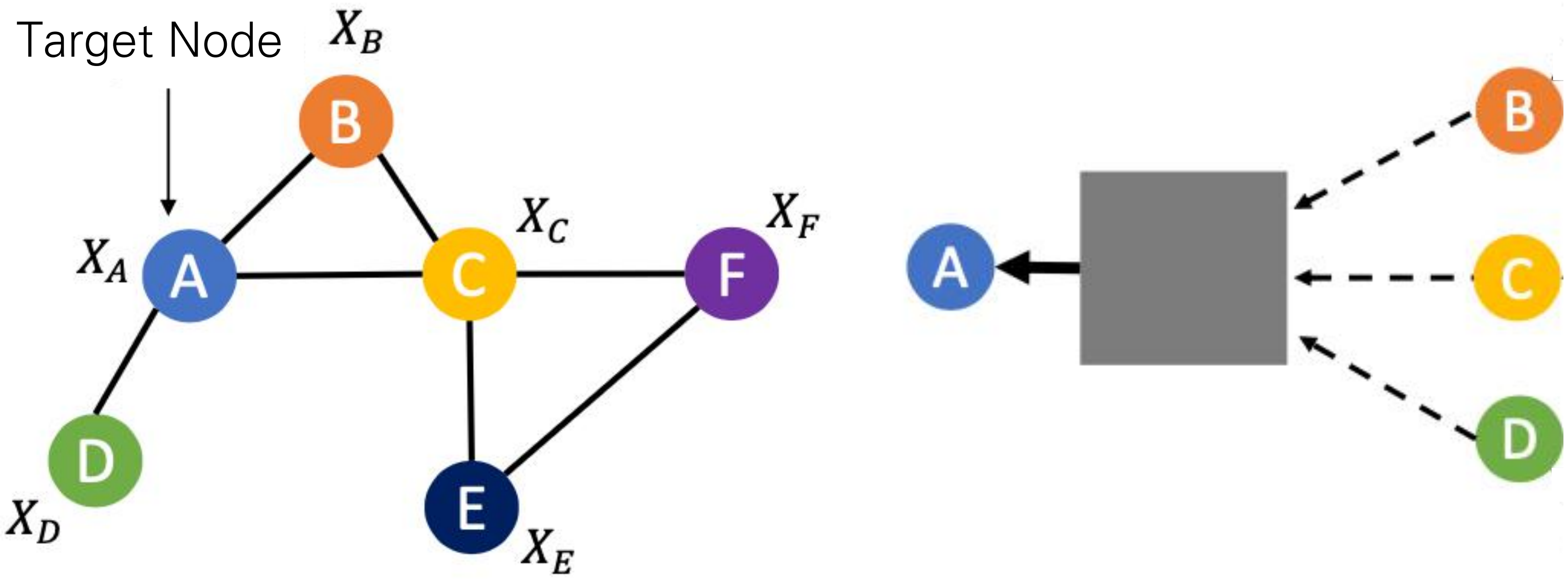
Graph Neural Networks (GNNs)



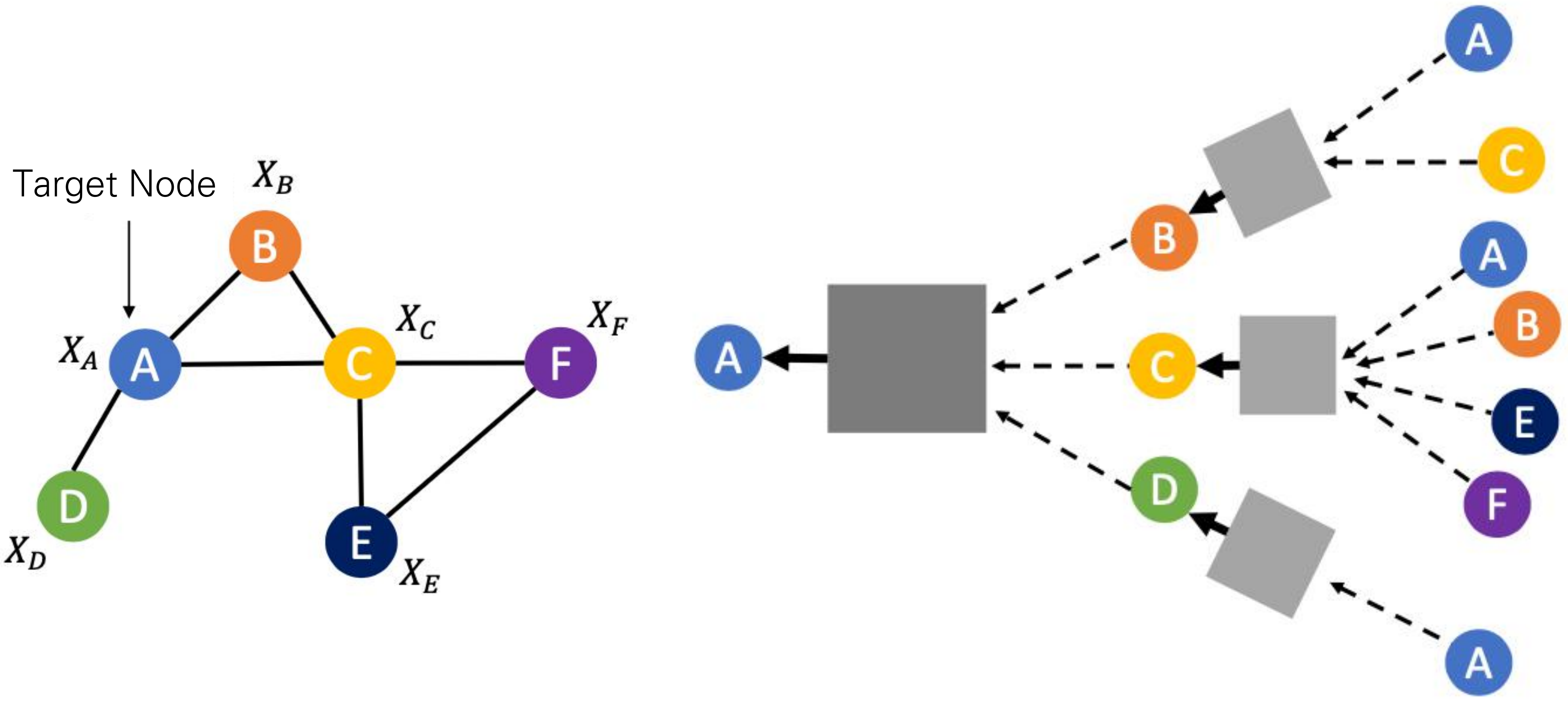
Graph Neural Networks (GNNs)



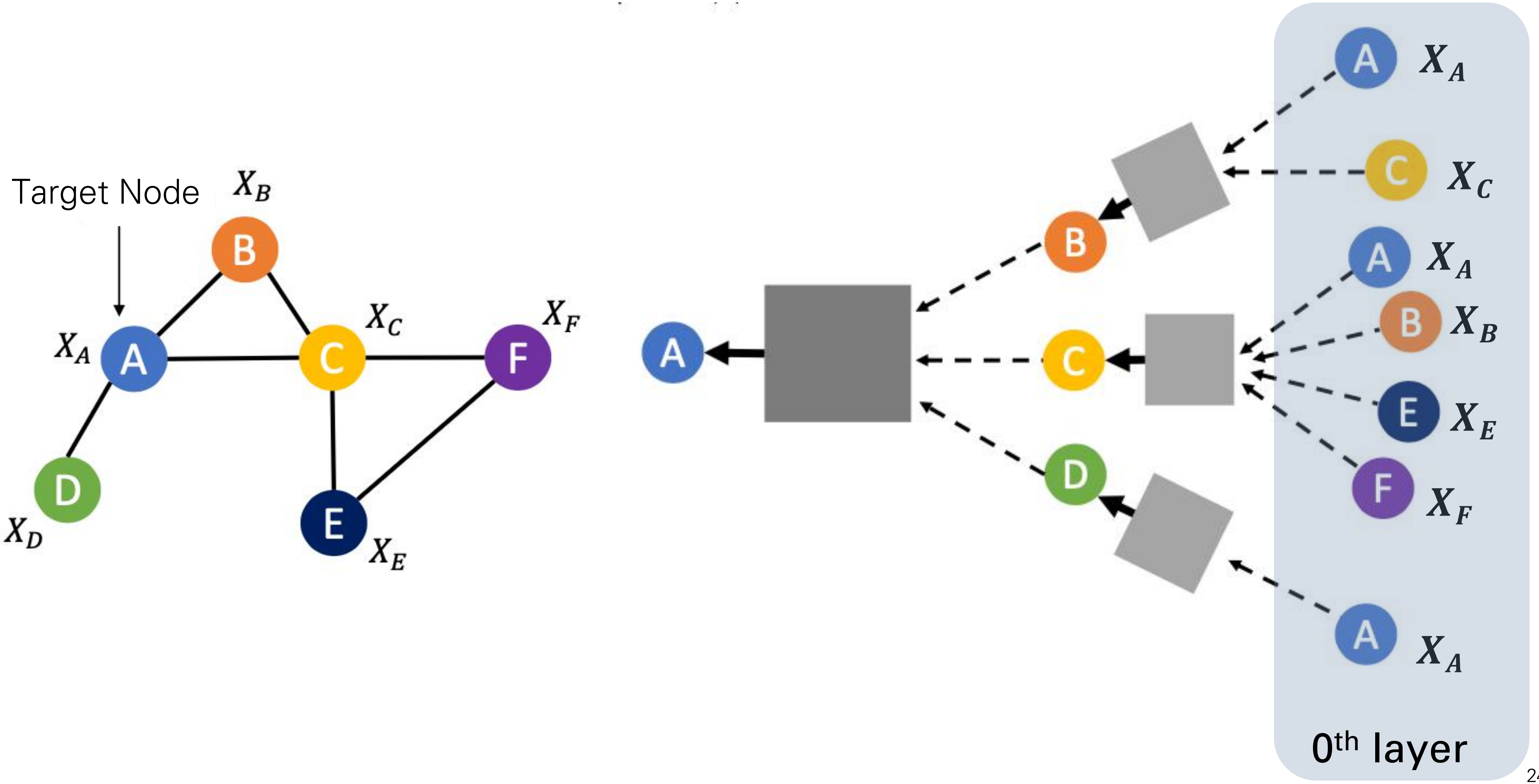
Graph Neural Networks (GNNs)



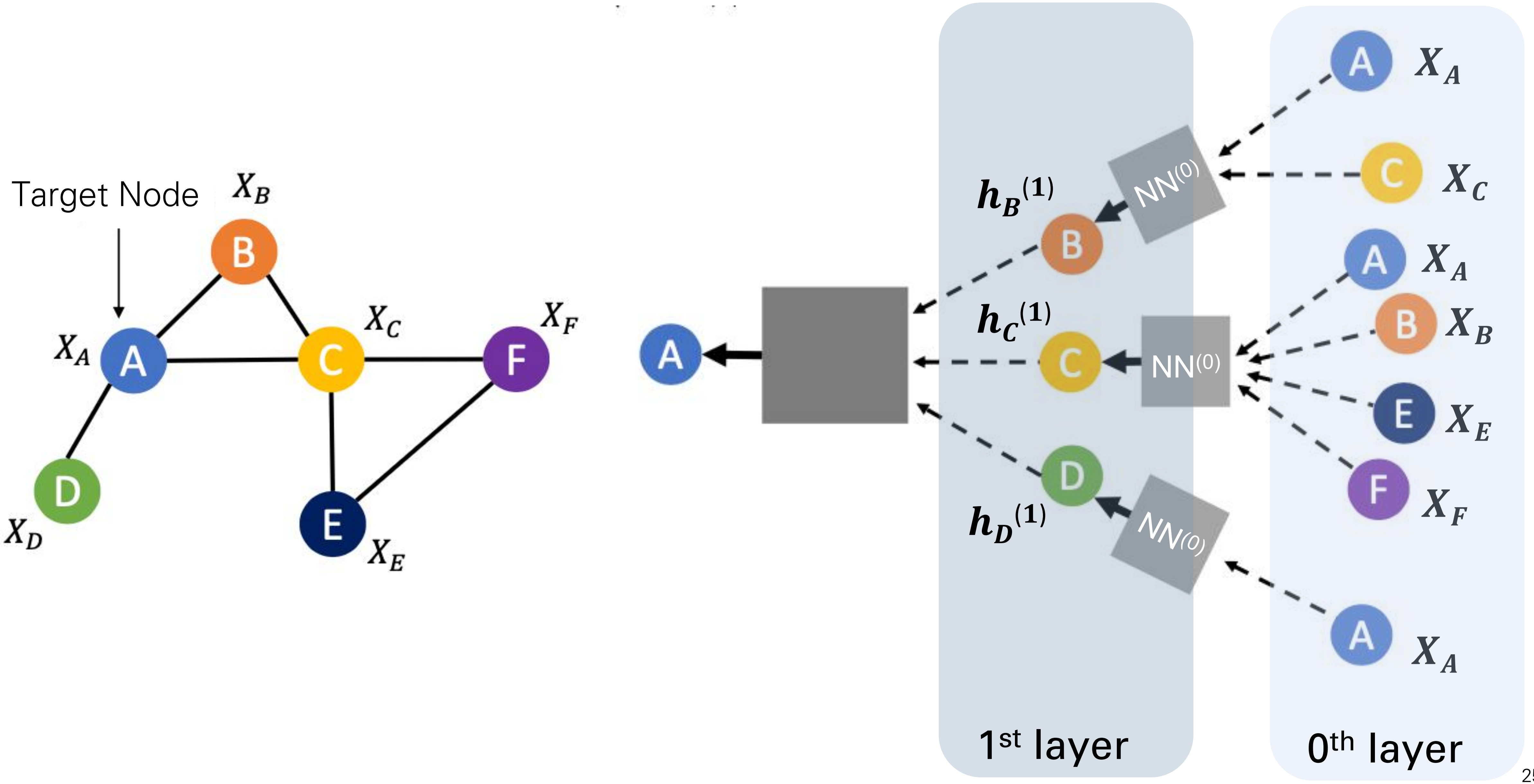
Graph Neural Networks (GNNs)



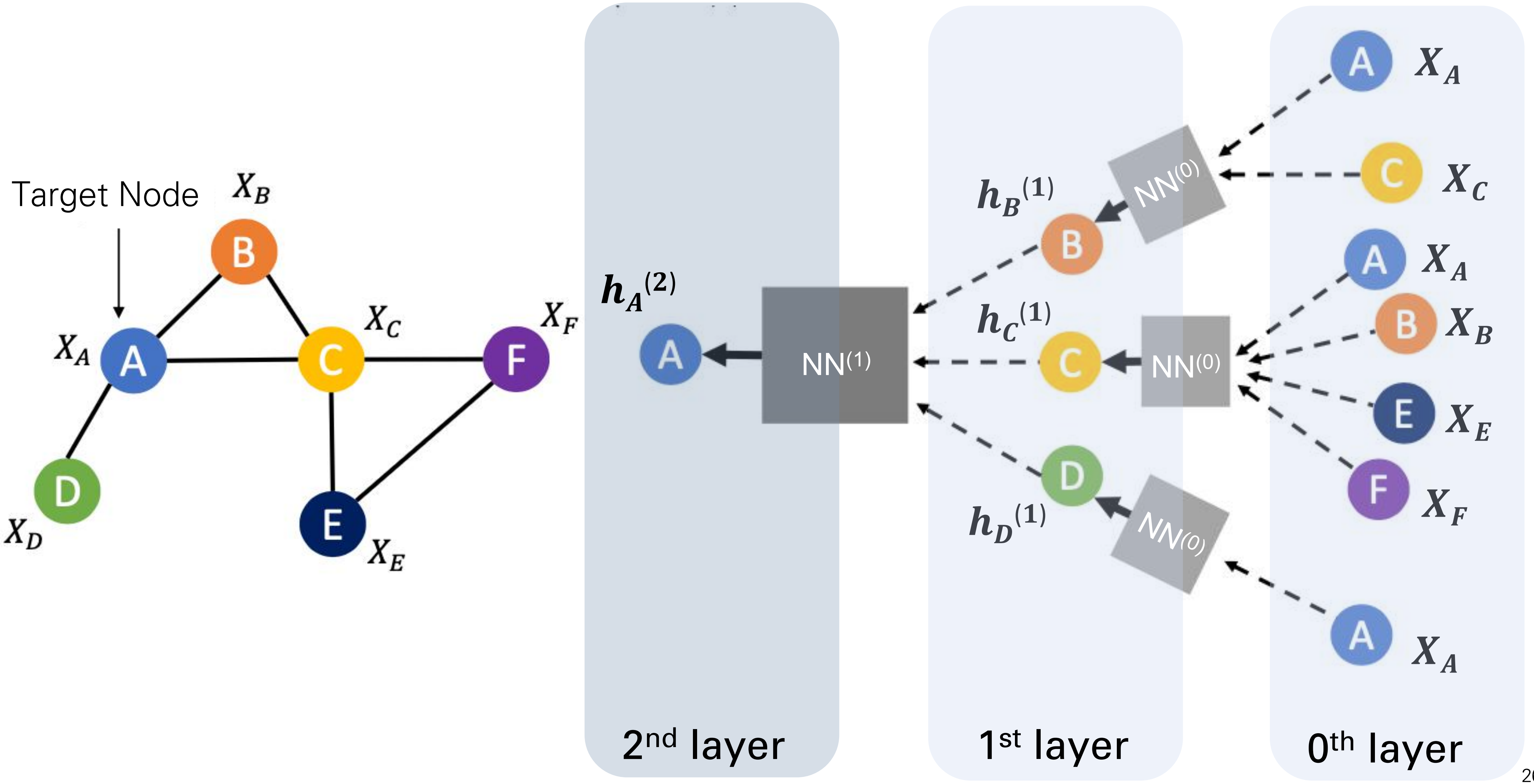
Graph Neural Networks (GNNs)



Graph Neural Networks (GNNs)



Graph Neural Networks (GNNs)

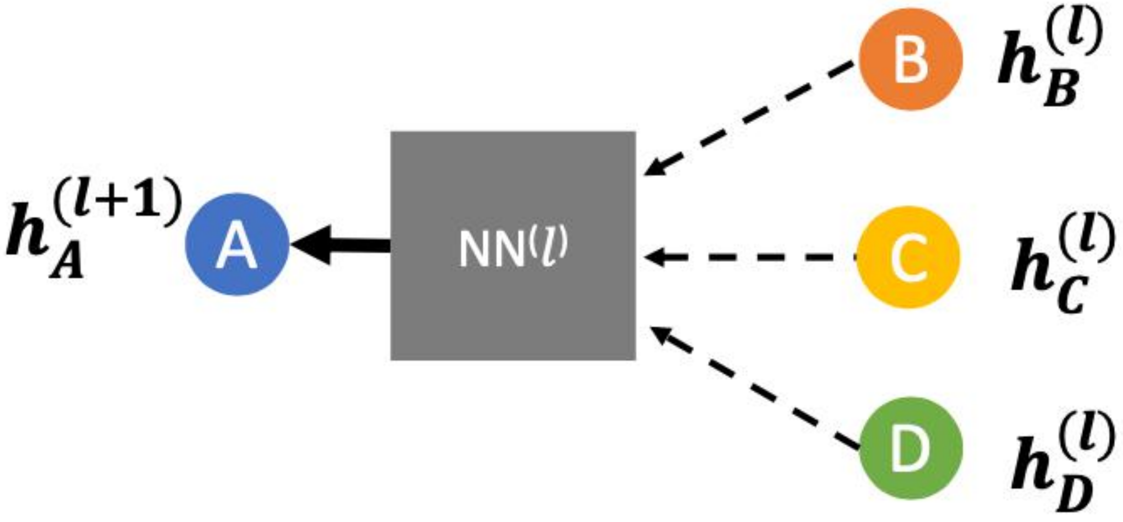


Graph Neural Networks (GNNs)

1. Aggregate messages from neighbors

$h_v^{(l)}$: node embedding of v at l-th layer
$\mathcal{N}(v)$: neighboring nodes of v
$\boldsymbol{f}^{(l)}$: aggregation function at l-th layer
$m_v^{(l)}$: message vector of v at l-th layer

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



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

Graph Neural Networks (GNNs)

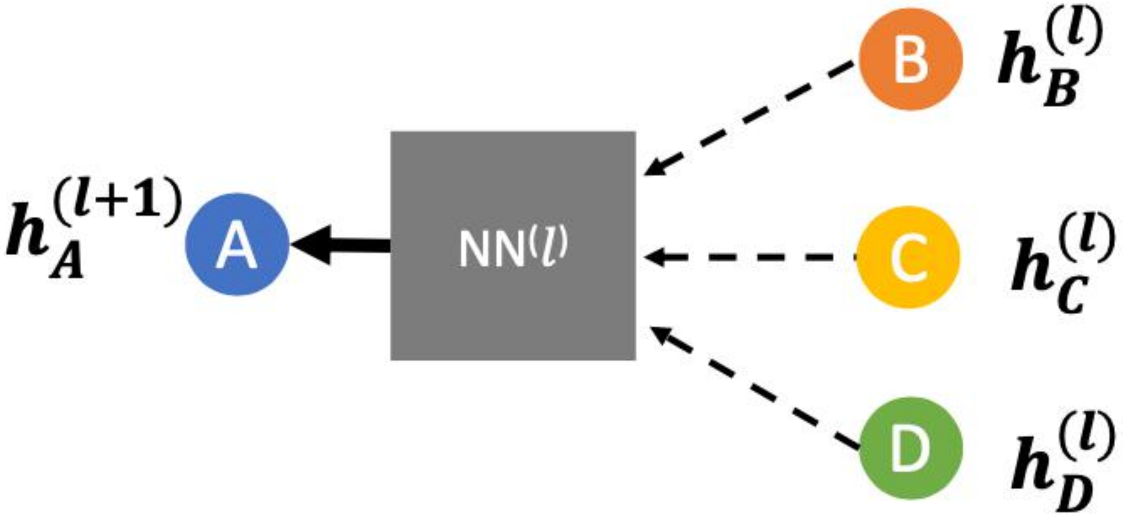
1. Aggregate messages from neighbors

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

2. Transform messages

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

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



Neighbors of node A

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

Graph Neural Networks (GNNs)

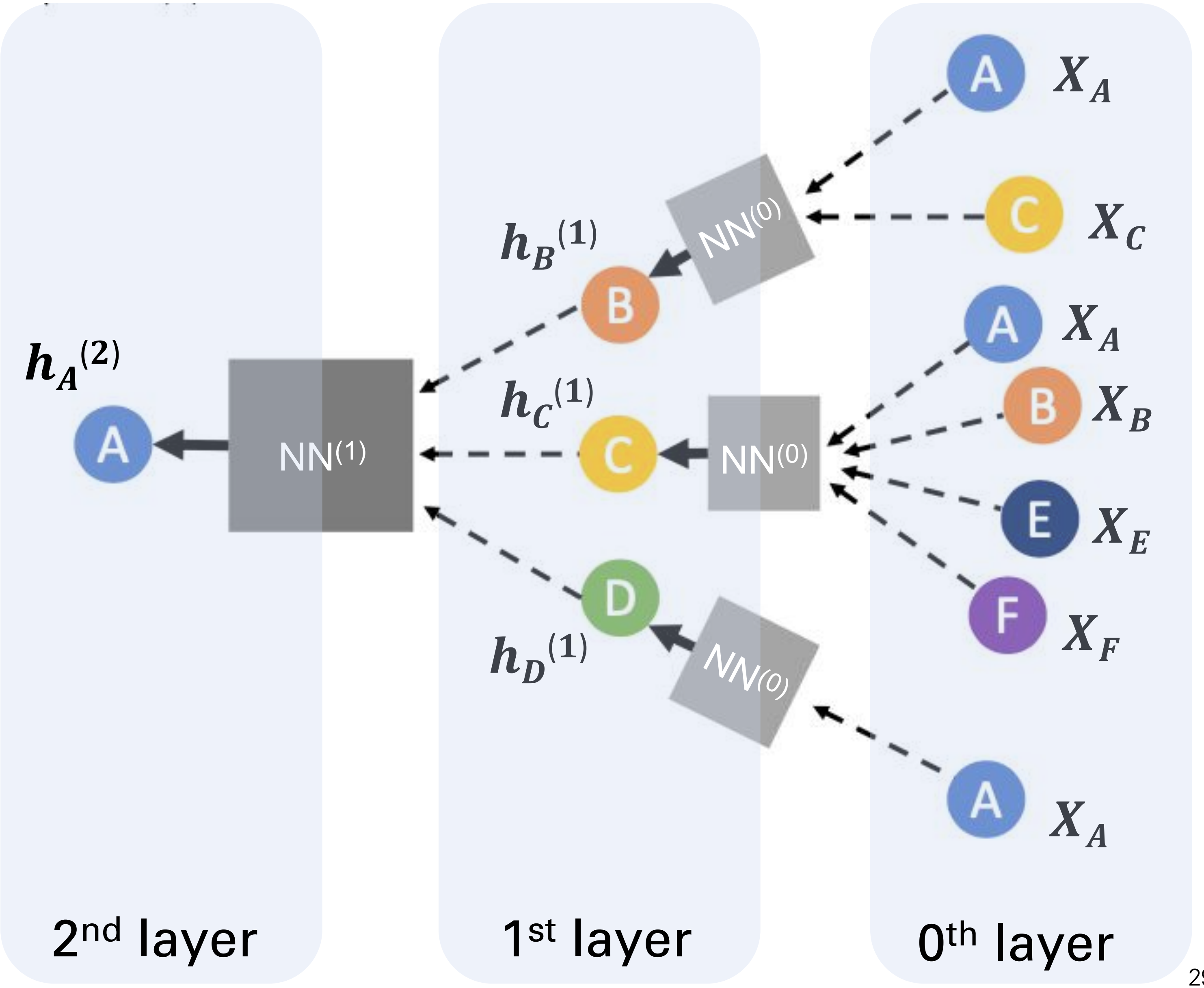
In each layer l ,
for each target node v:

1. Aggregate messages

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

2. Transform messages

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



Graph Neural Networks (GNNs)

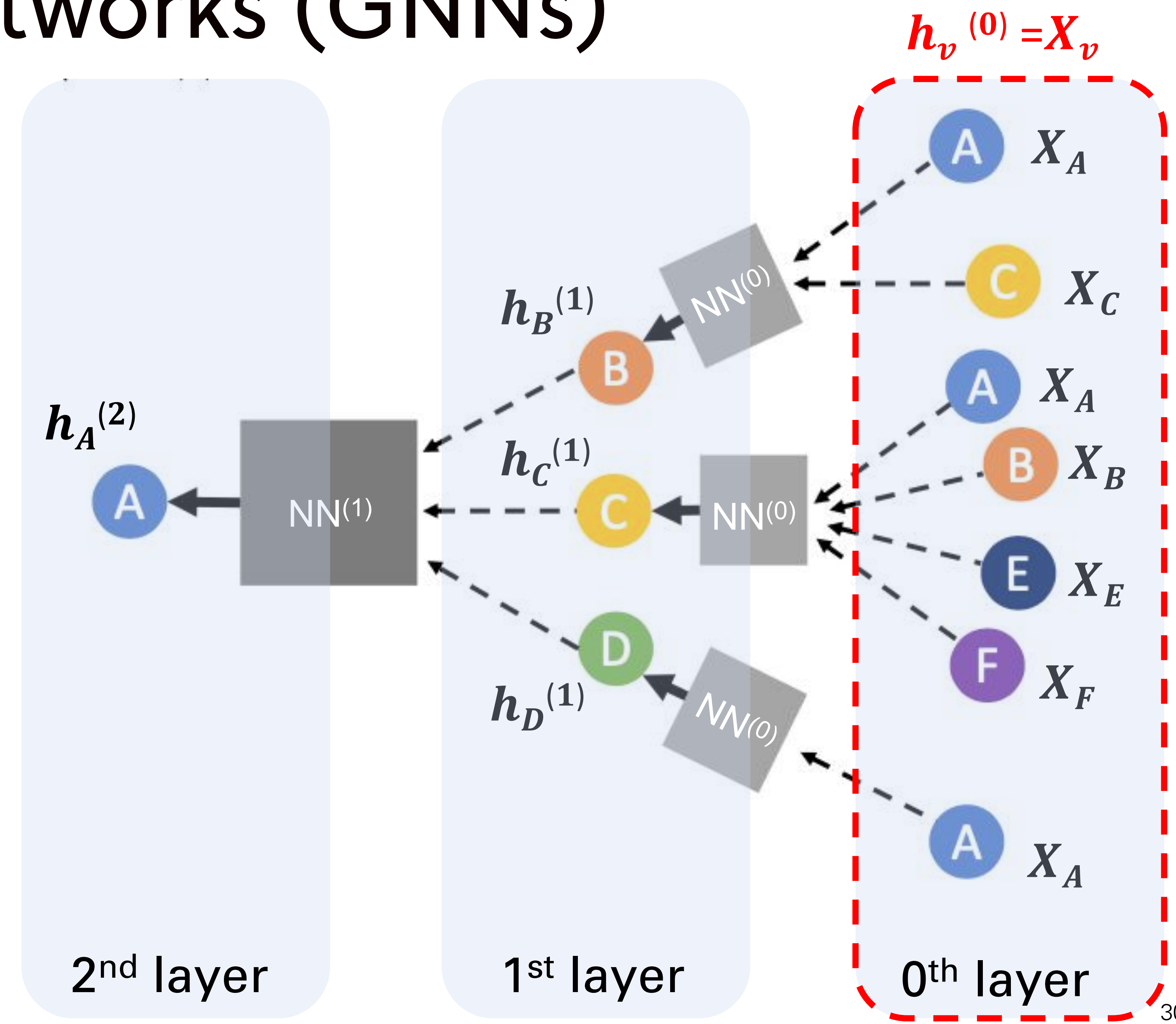
In each layer l ,
for each target node v:

1. Aggregate messages

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

2. Transform messages

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



Graph Neural Networks (GNNs)

In each layer l ,
for each target node v:

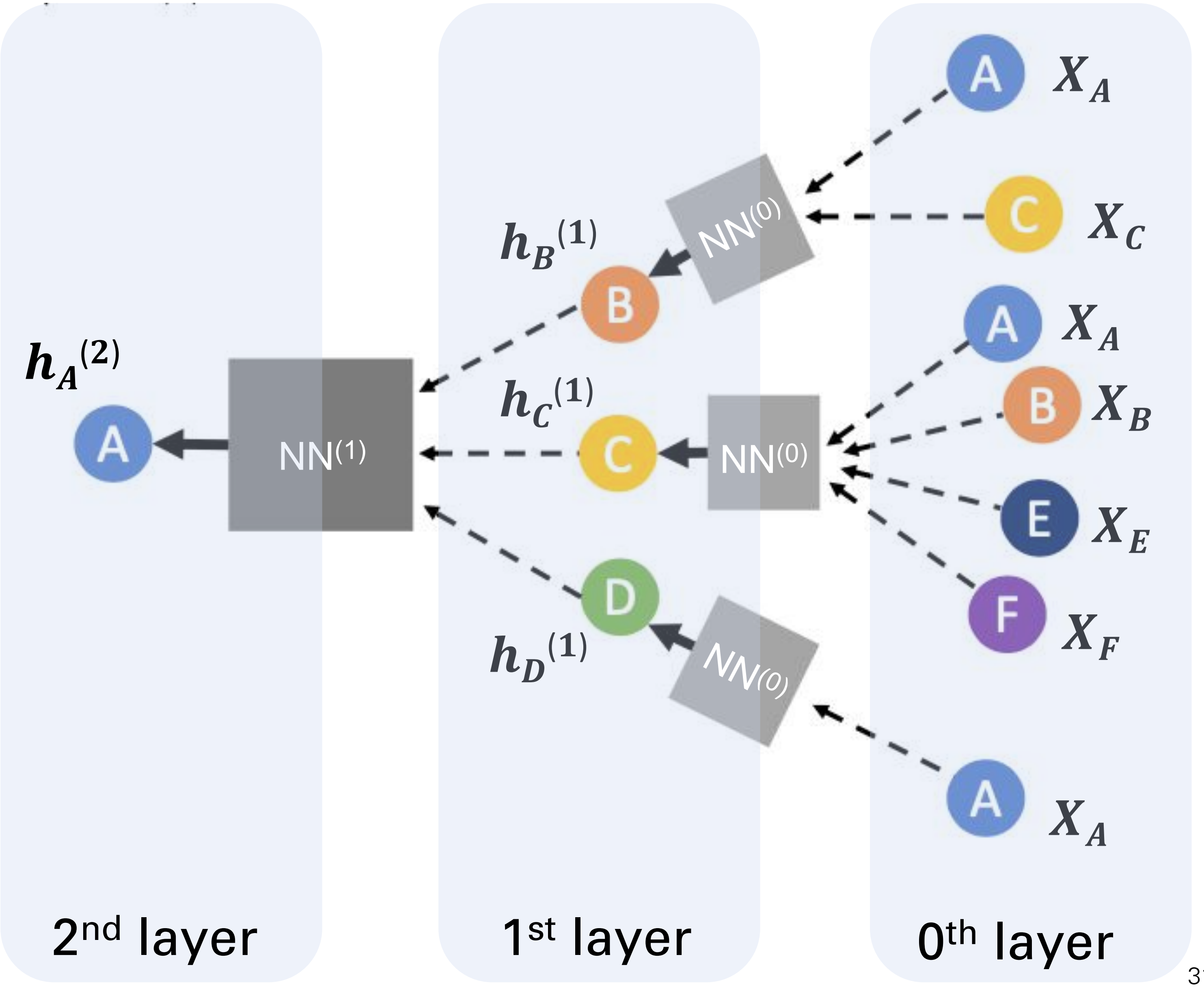
1. Aggregate messages

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

2. Transform messages

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

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



Graph Neural Networks (GNNs)

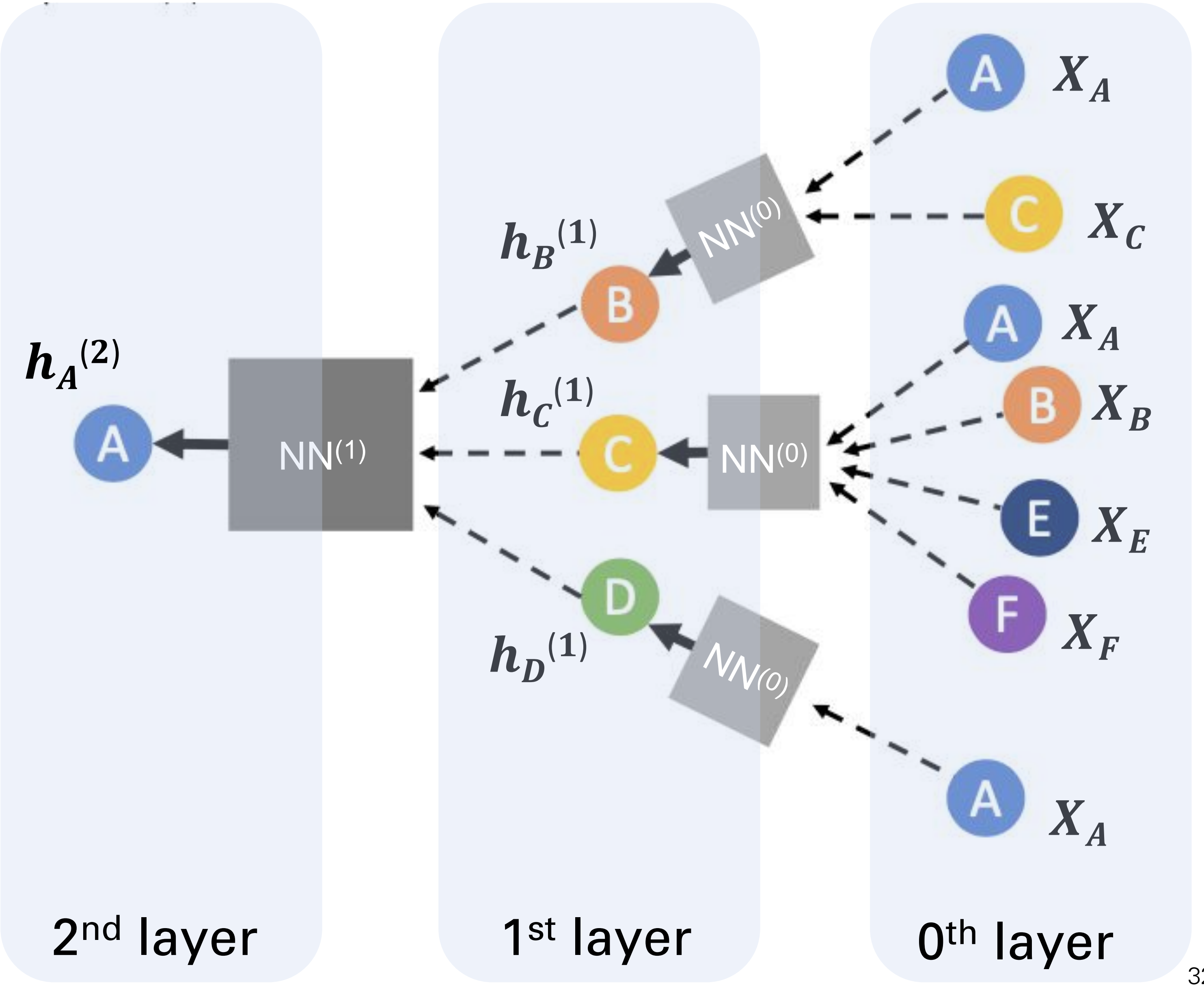
In each layer l ,
for each target node v:

1. Aggregate messages

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

2. Transform messages

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



Graph Neural Networks (GNNs)

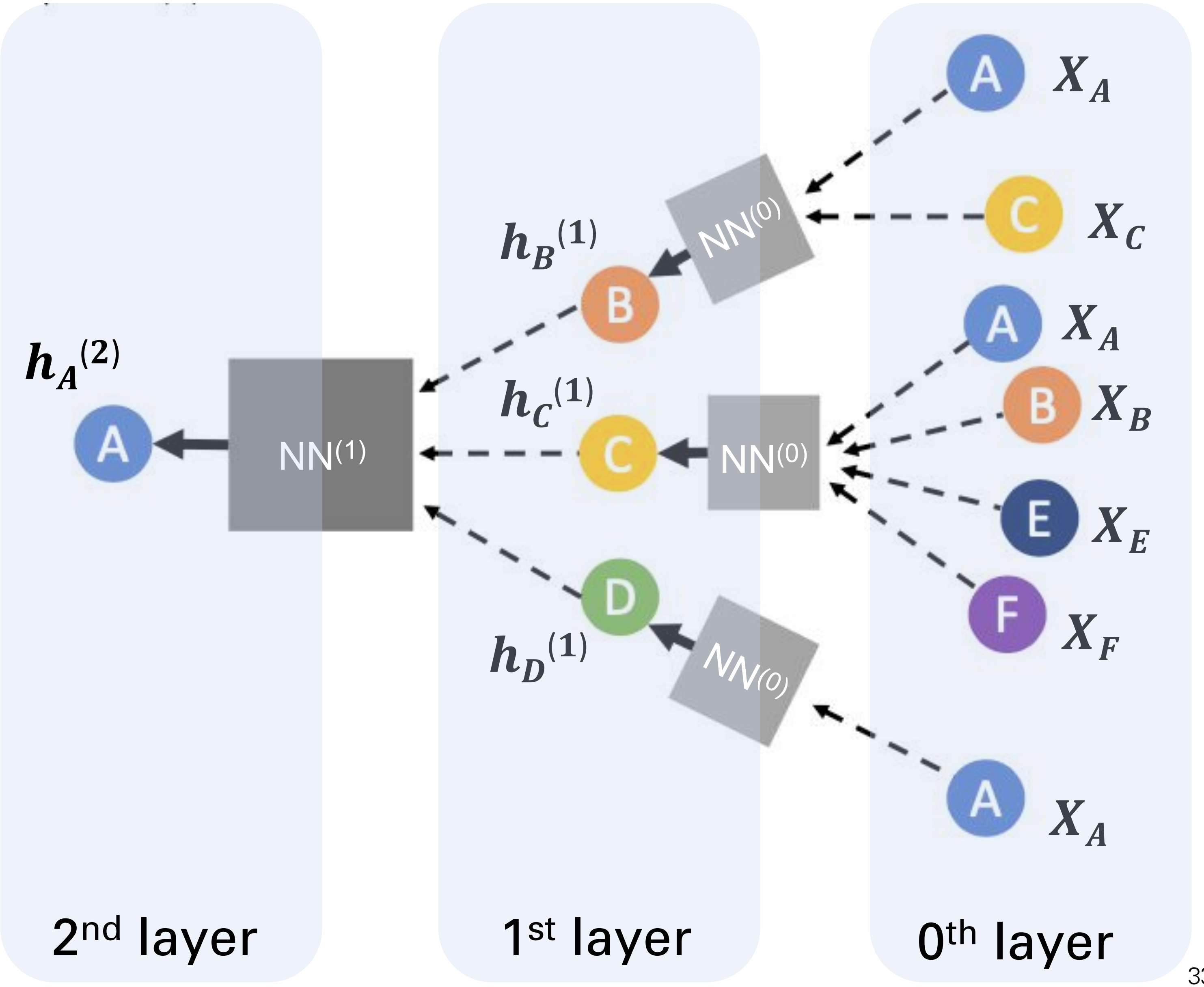
Graph Convolutional Networks[1]

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

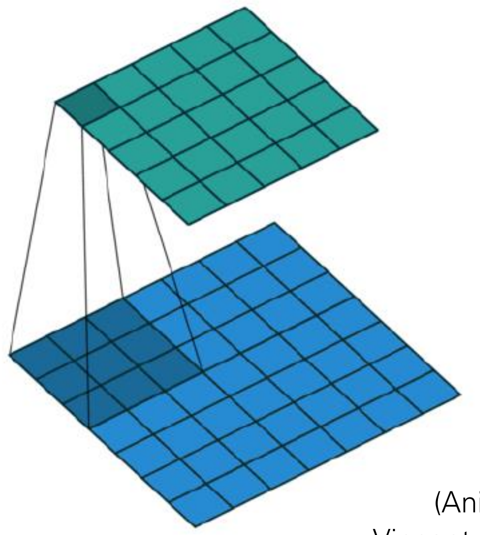
$$h_v^{(l+1)} = \sigma(\boldsymbol{W}^{(l)} \circ m_v^{(l)})$$



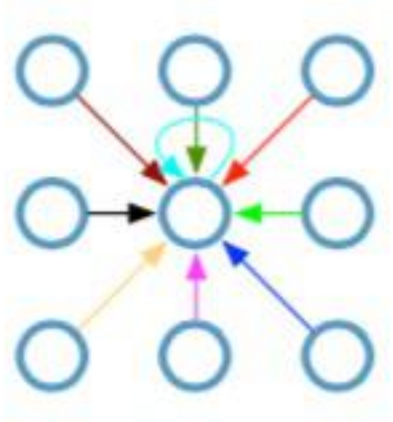
[1] Kipf, Thomas N., et al. "Semi-supervised classification with graph convolutional networks."

Recap: Convolutional neural networks (on grids)

Single CNN layer
with 3x3 filter:

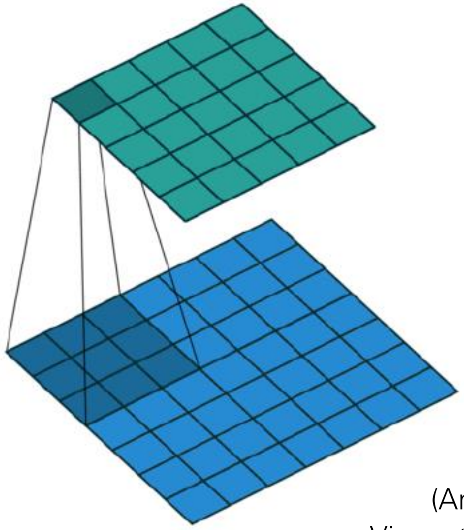


(Animation by
Vincent Dumoulin)

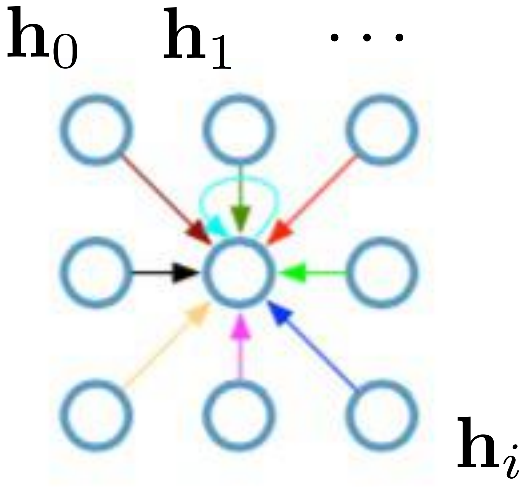


Recap: Convolutional neural networks (on grids)

Single CNN layer
with 3x3 filter:

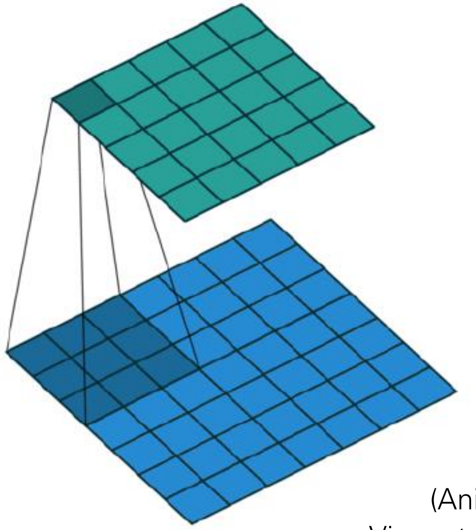


(Animation by
Vincent Dumoulin)

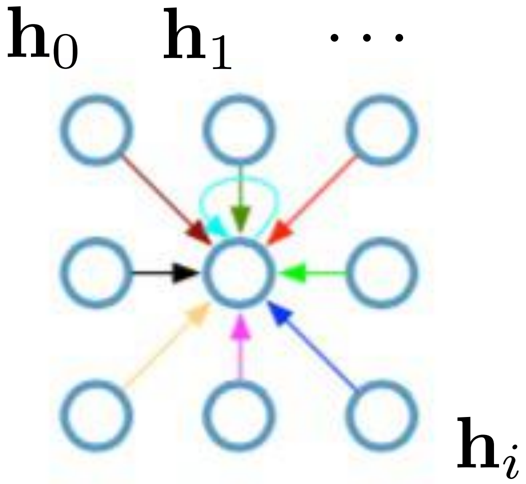


Recap: Convolutional neural networks (on grids)

Single CNN layer
with 3x3 filter:



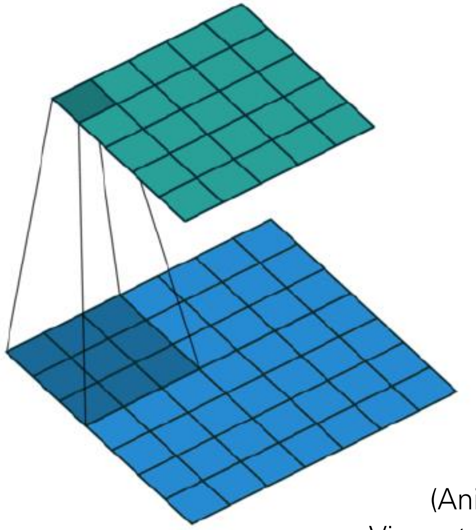
(Animation by
Vincent Dumoulin)



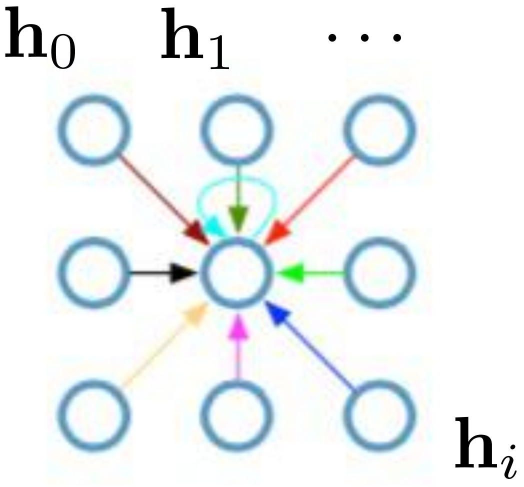
$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

Recap: Convolutional neural networks (on grids)

Single CNN layer
with 3x3 filter:



(Animation by
Vincent Dumoulin)



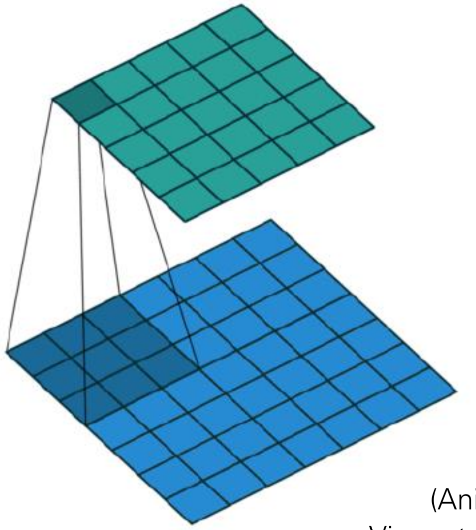
Update for a single pixel:

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

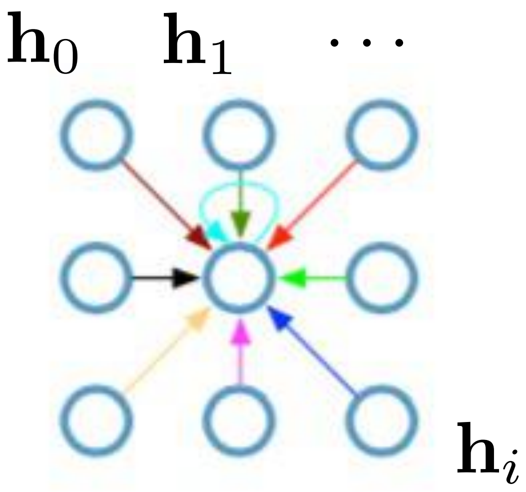
$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

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 $\sum_i \mathbf{W}_i \mathbf{h}_i$

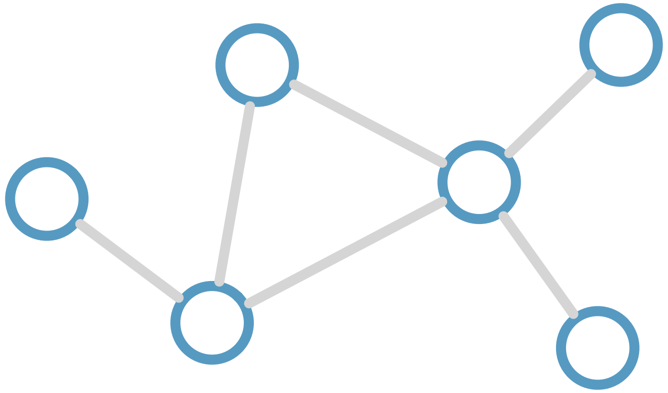
$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

Full update:

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

Graph Convolutional Networks (GCNs)

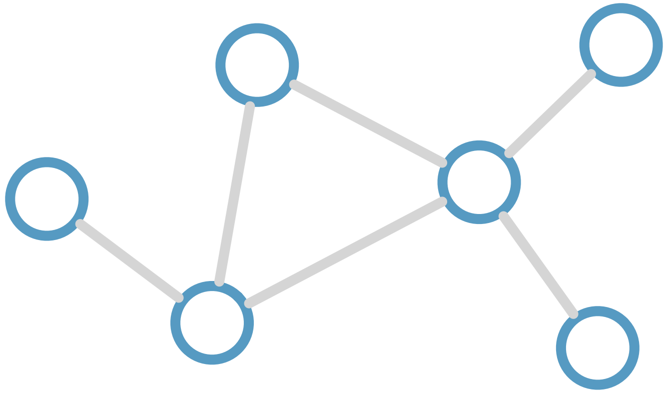
Consider this
undirected graph:



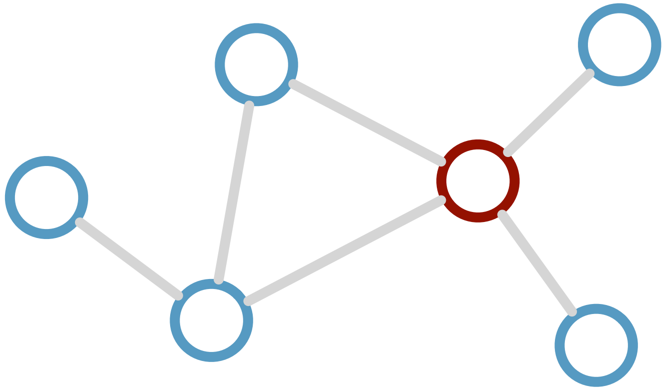
Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Graph Convolutional Networks (GCNs)

Consider this undirected graph:



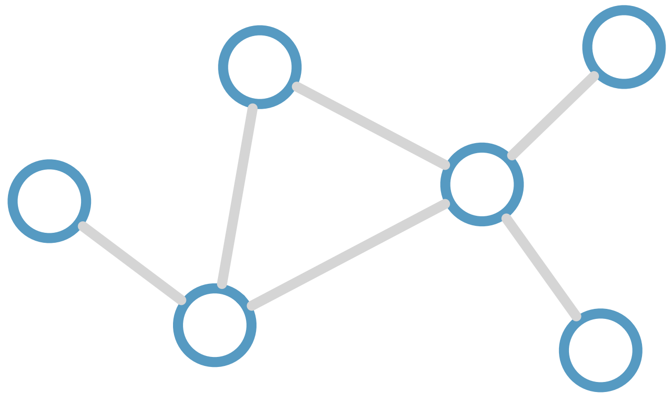
Consider update for node in red:



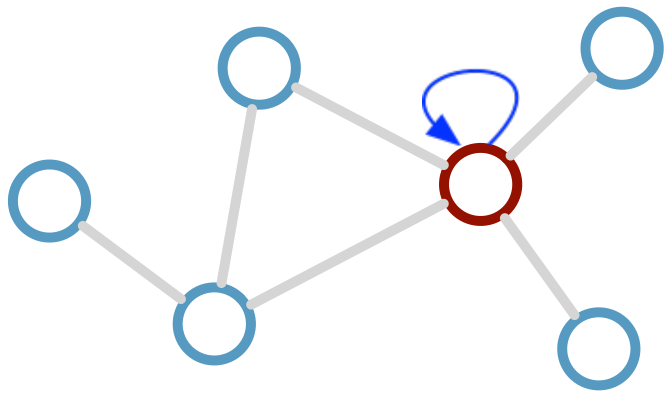
Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Graph Convolutional Networks (GCNs)

Consider this undirected graph:



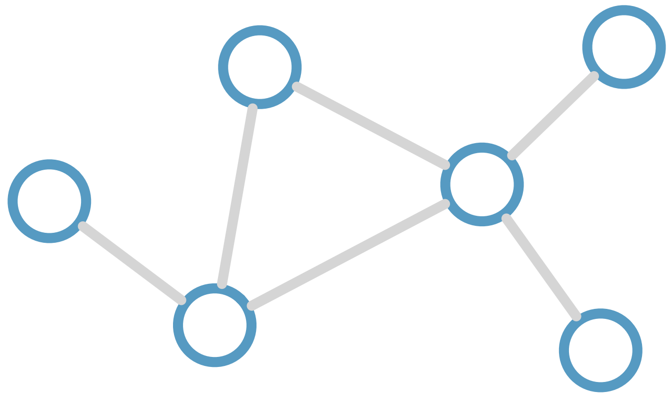
Consider update for node in red:



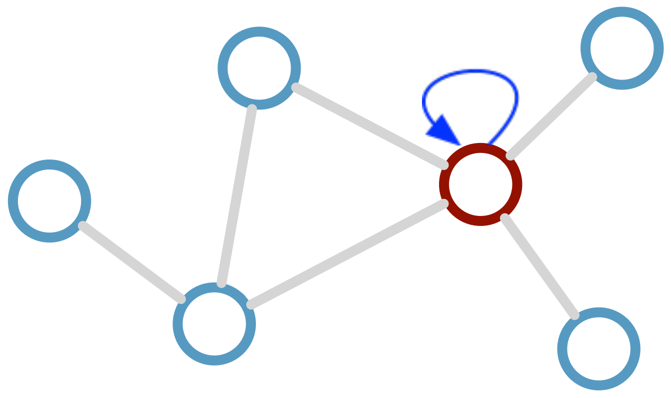
Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Graph Convolutional Networks (GCNs)

Consider this undirected graph:



Consider update for node in red:



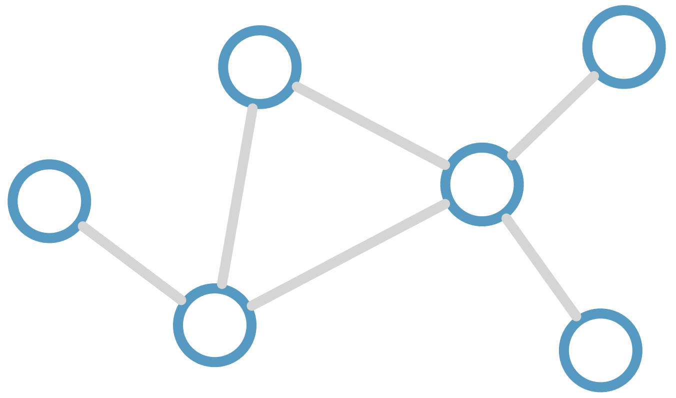
Update rule: $$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

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

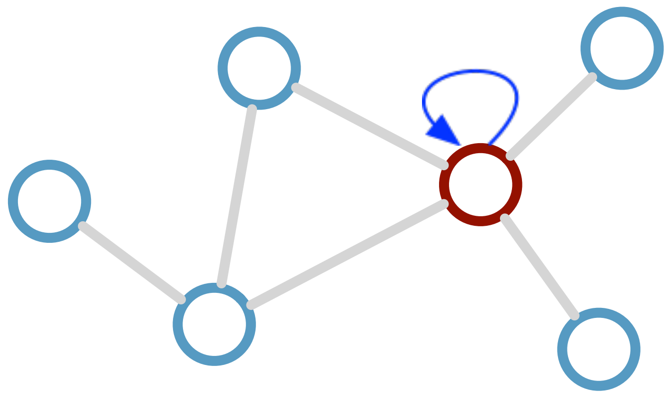
Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

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:

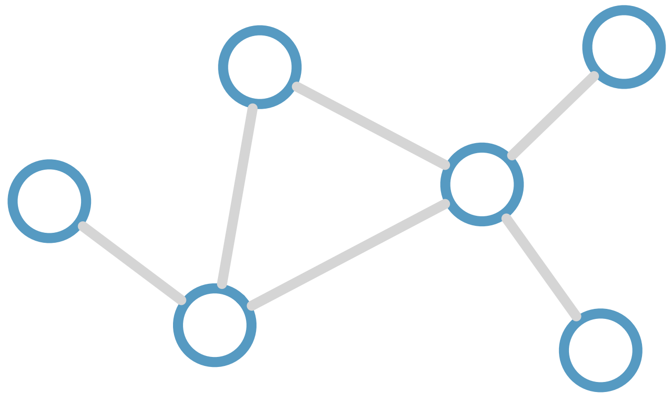
$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

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

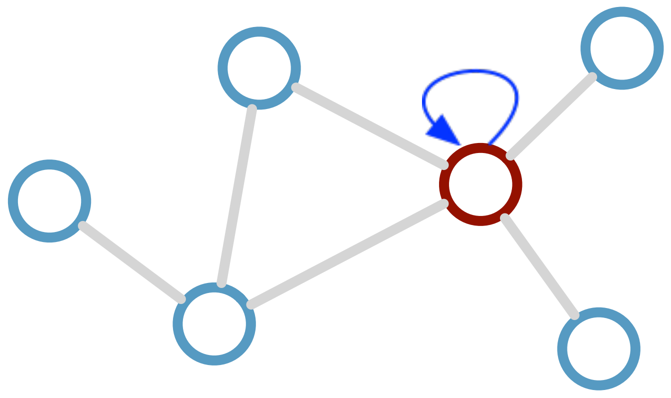
Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

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 $\quad c_{ij}$: 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

Kipf & Welling (ICLR 2017), related previous works by Duvenaud et al. (NIPS 2015) and Li et al. (ICLR 2016)

Graph Neural Networks (GNNs)

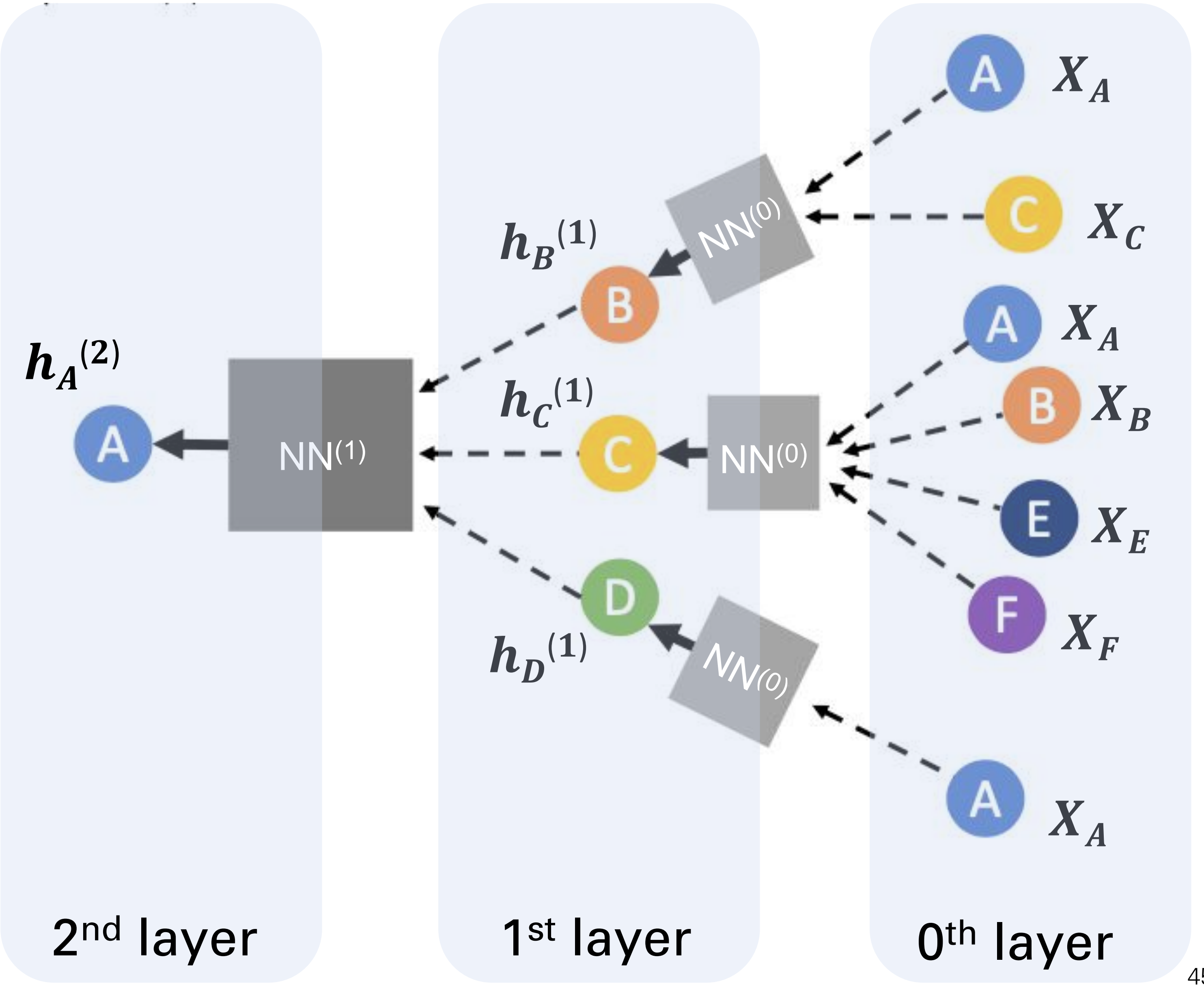
Graph Convolutional Networks[1]

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

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



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

Graph Neural Networks (GNNs)

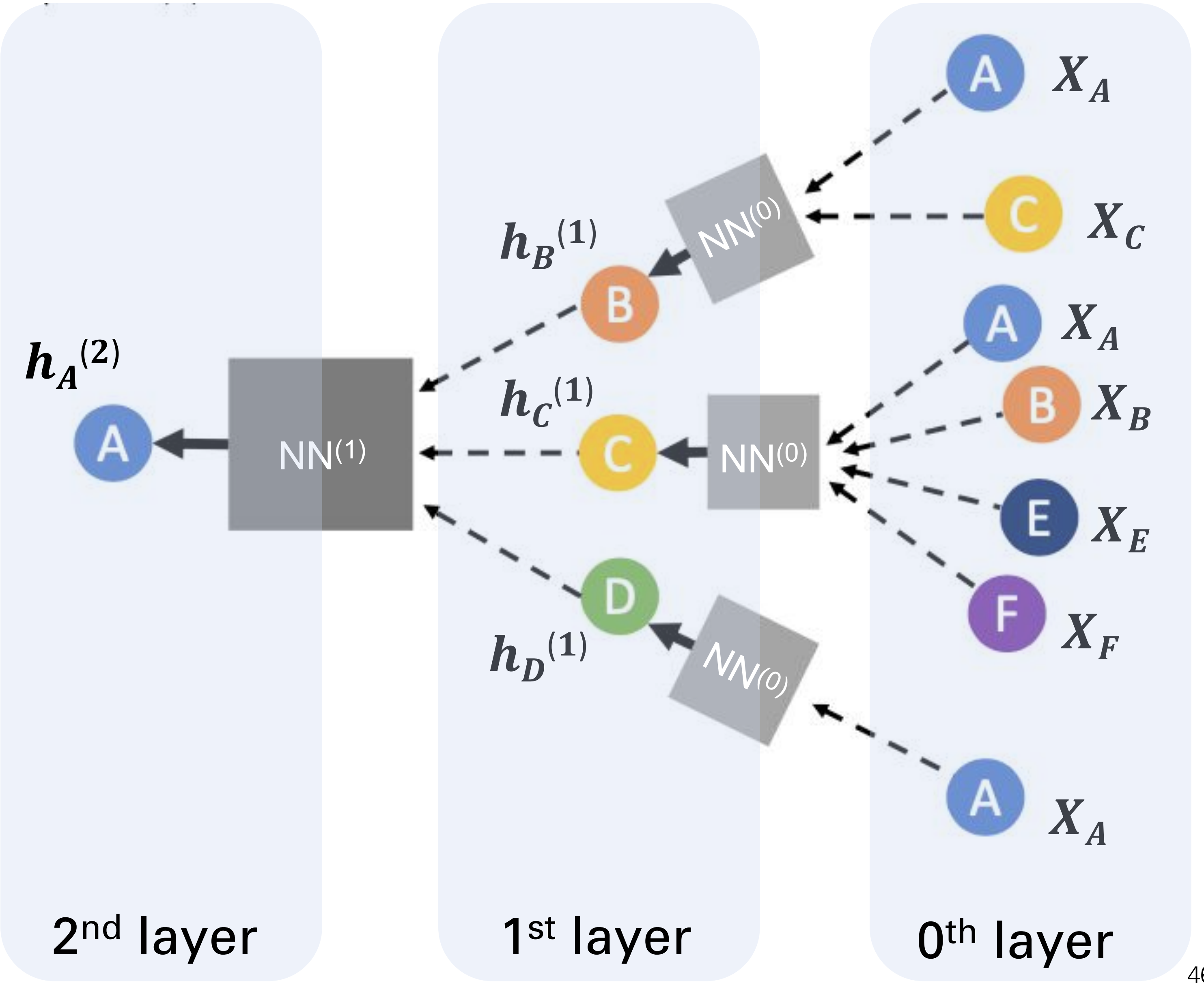
Graph Isomorphism Networks[2]

1. Aggregate messages

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

2. Transform messages

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



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

Graph Neural Networks (GNNs)

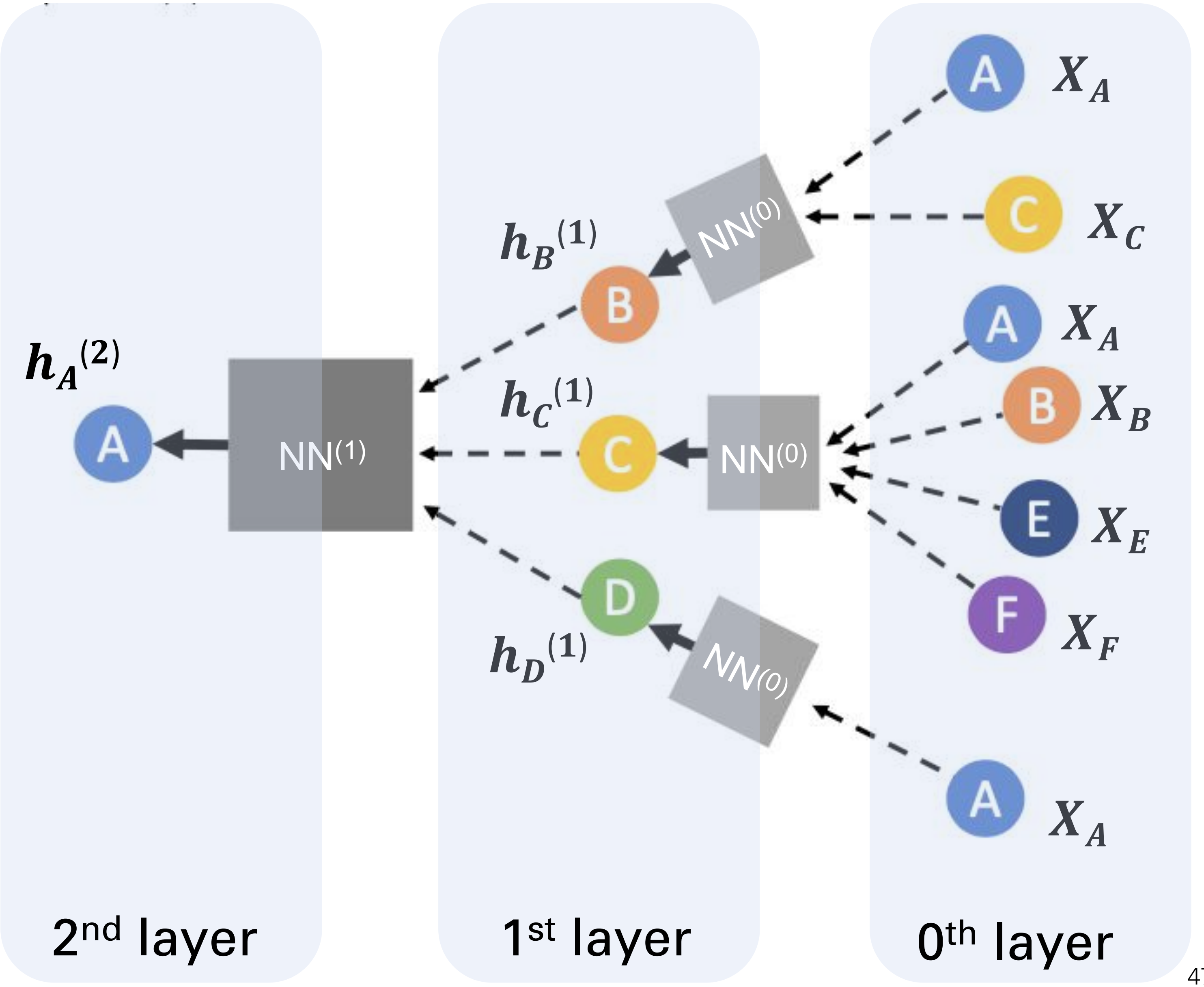
Simplified Graph Convolutional Networks[2]

1. Aggregate messages

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

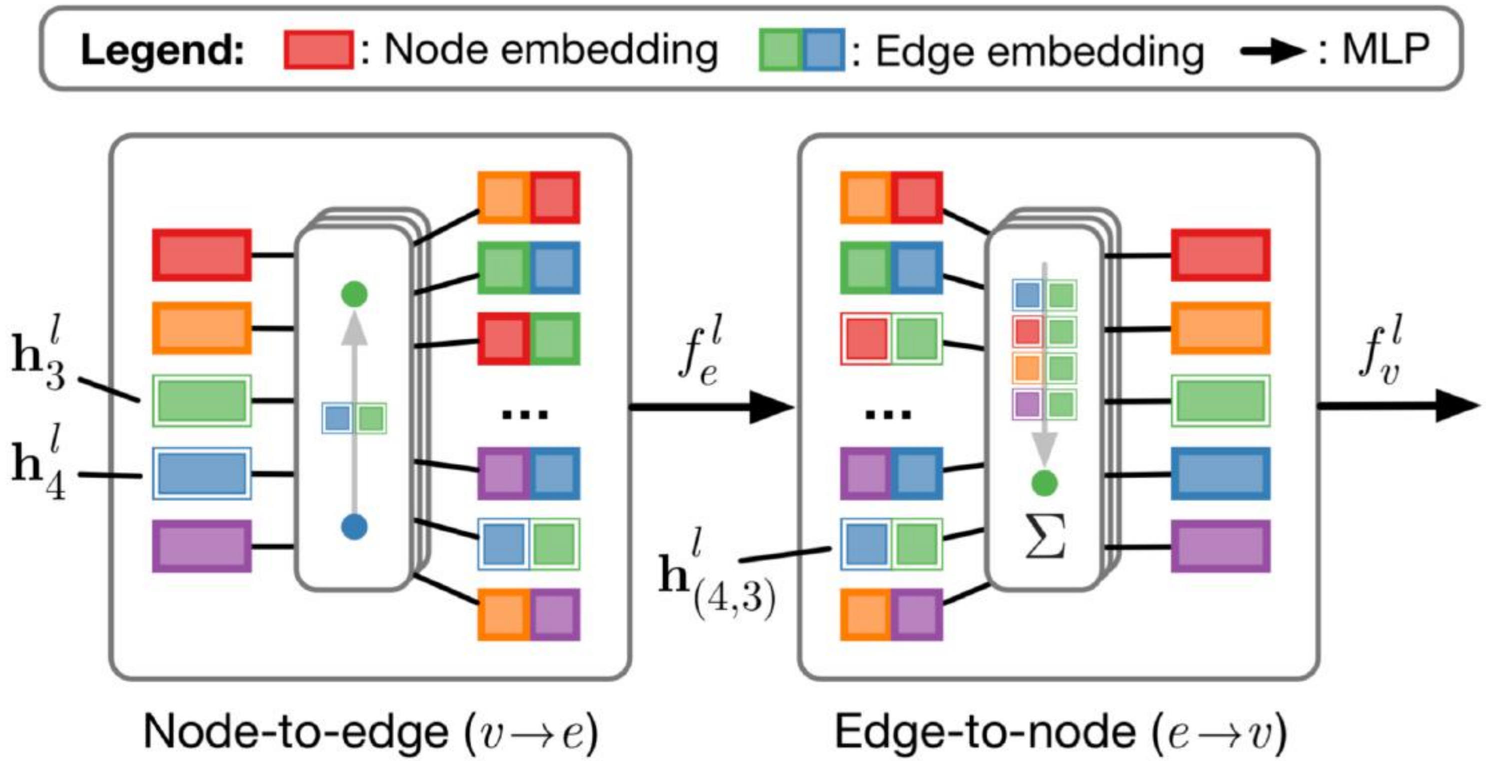
2. Transform messages

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



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

GCNs with edge embeddings

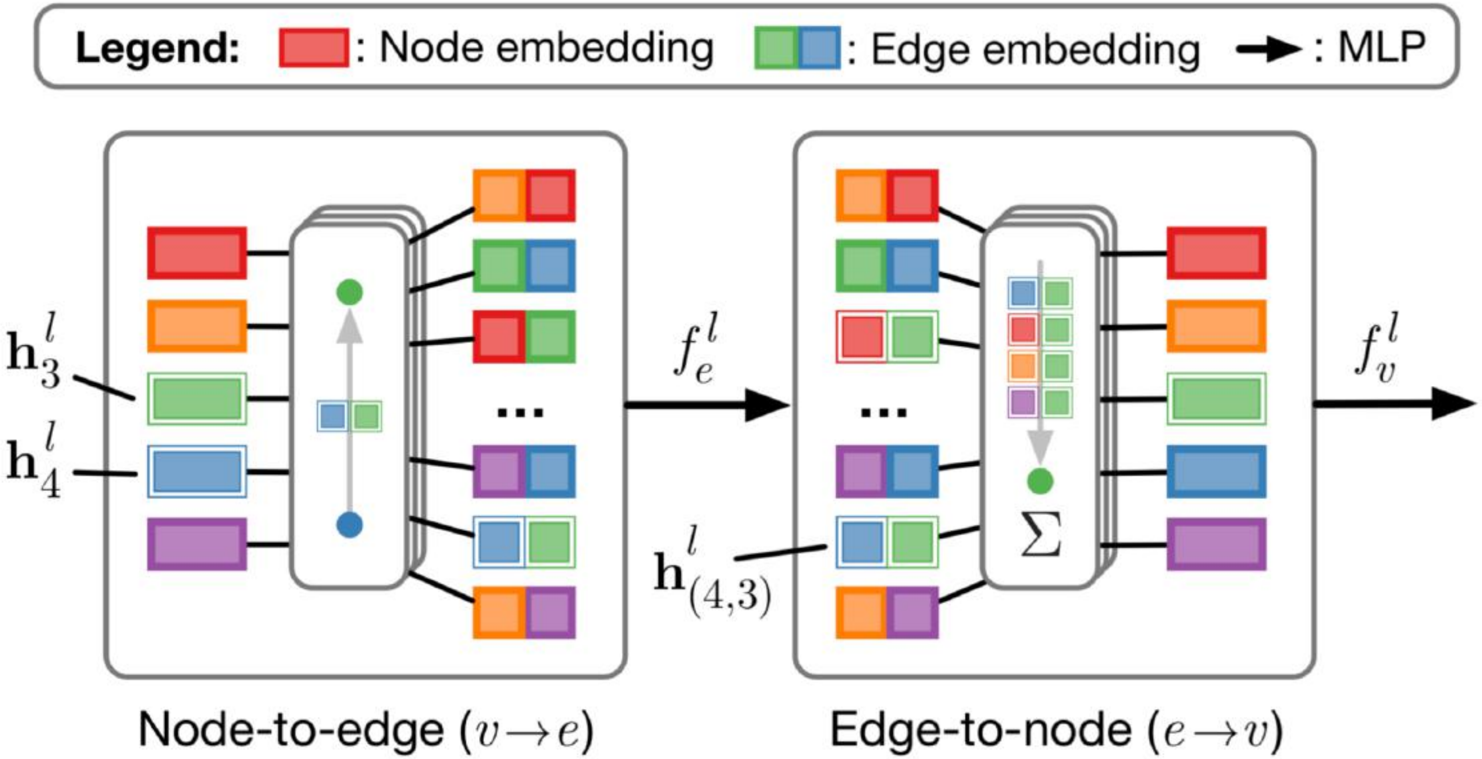


Formally: $v \to e: \mathbf{h}_{(i,j)}^l = f_e^l\left(\left[\mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{x}_{(i,j)}\right]\right)$

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

Battaglia et al. (NIPS 2016), Gilmer et al. (ICML 2017), Kipf et al. (ICML 2018)

GCNs with edge embeddings



Pros:

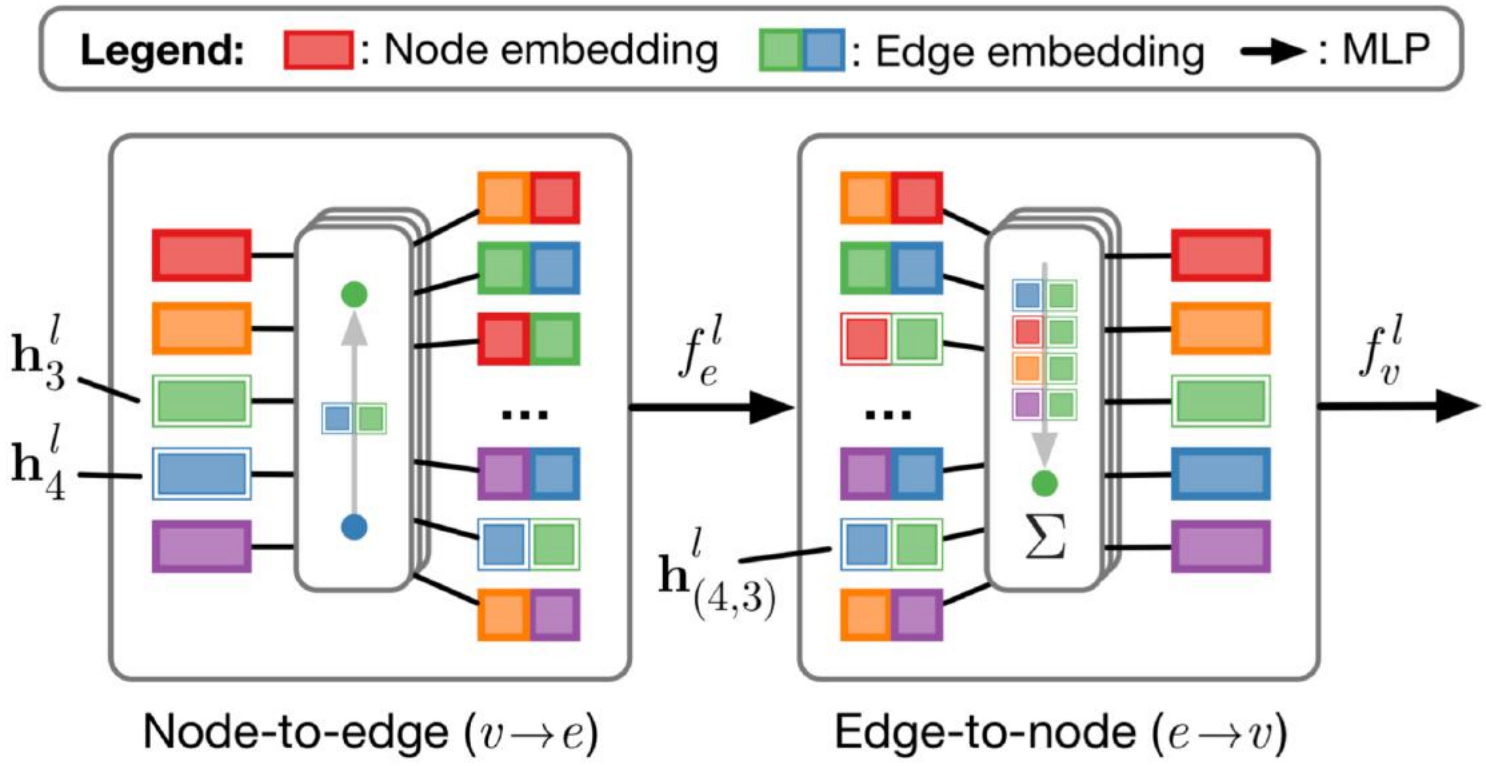
- Supports edge features
- More expressive than GCN
- As general as it gets (?)
- Supports sparse matrix ops

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

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

Battaglia et al. (NIPS 2016), Gilmer et al. (ICML 2017), Kipf et al. (ICML 2018)

GCNs with edge embeddings



Pros:

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

Cons:

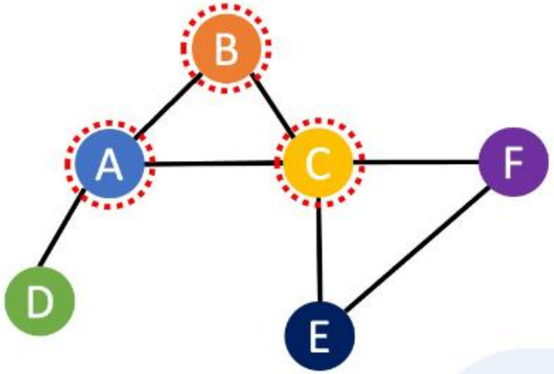
- Need to store intermediate edge-based activations
- Difficult to implement with subsampling
- In practice limited to small graphs

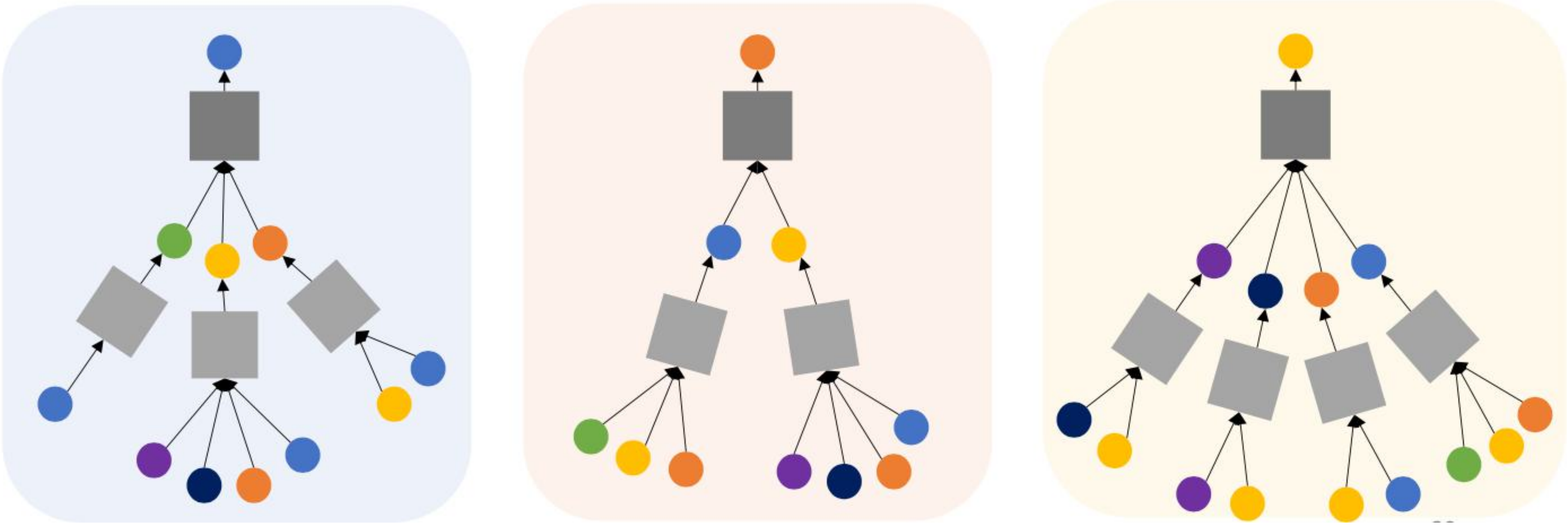
Formally: $v \to e : \mathbf{h}_{(i,j)}^l = f_e^l\left(\left[\mathbf{h}_i^l, \mathbf{h}_j^l, \mathbf{x}_{(i,j)}\right]\right)$

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

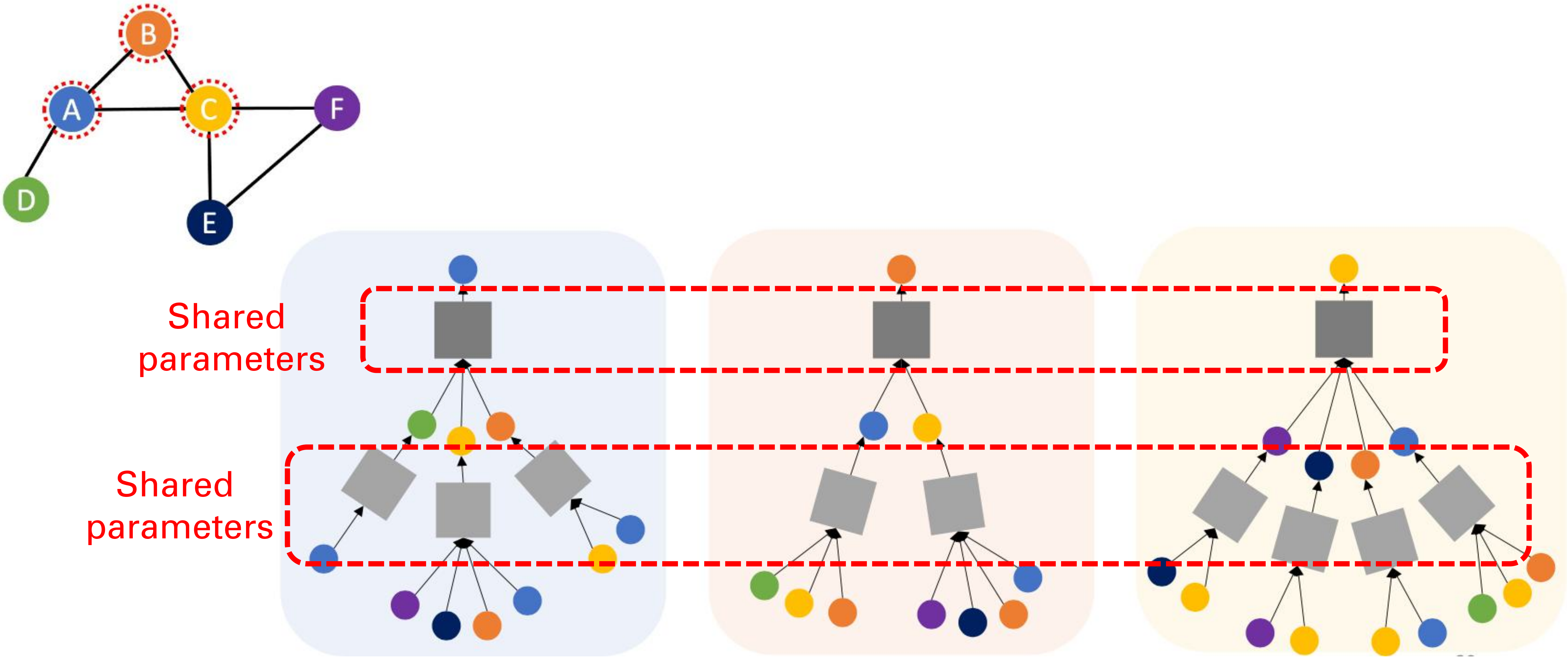
Battaglia et al. (NIPS 2016), Gilmer et al. (ICML 2017), Kipf et al. (ICML 2018)

Computation graphs



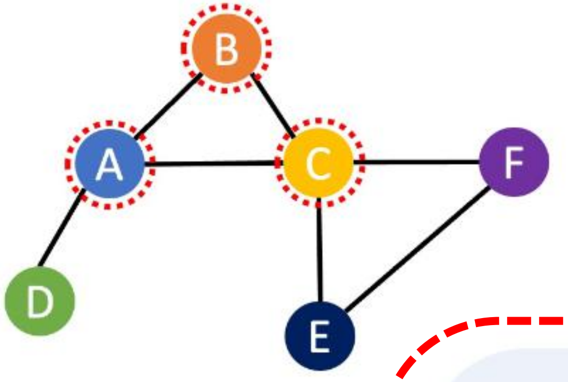


Computation graphs

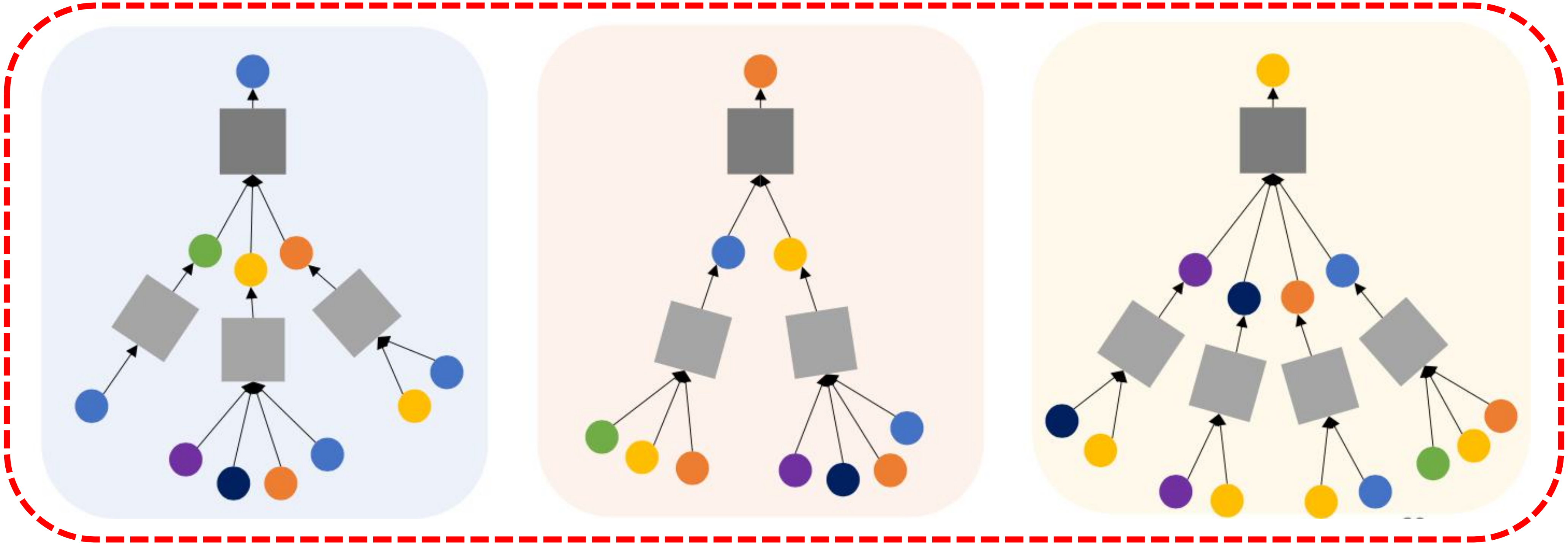


Batch execution

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



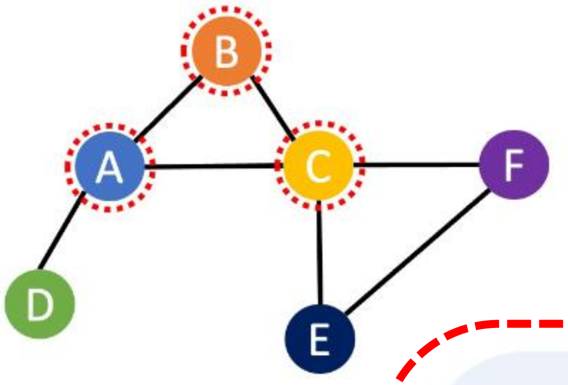
Batch size = 3



Batch execution

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

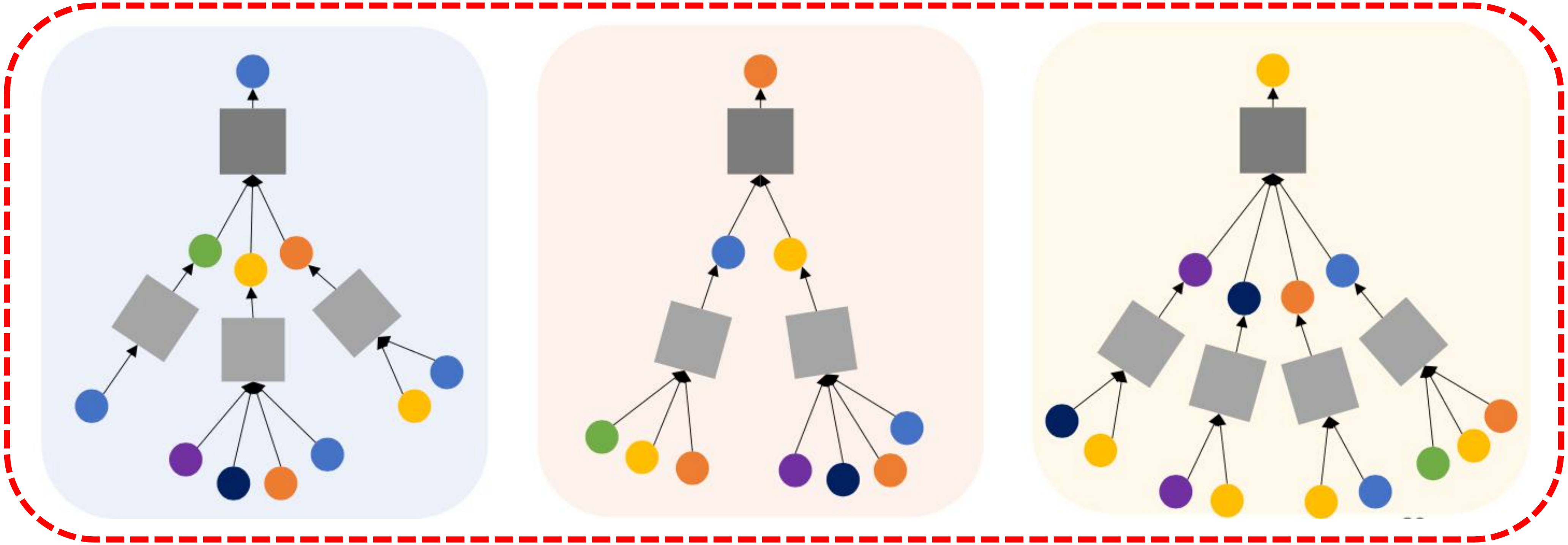
$$\boxed{\boldsymbol{H}^{(l)}} = \sigma(\boxed{(\widetilde{\boldsymbol{A}+\boldsymbol{I}})}\boldsymbol{H}^{(l-1)}\boldsymbol{W}^{(l)})$$



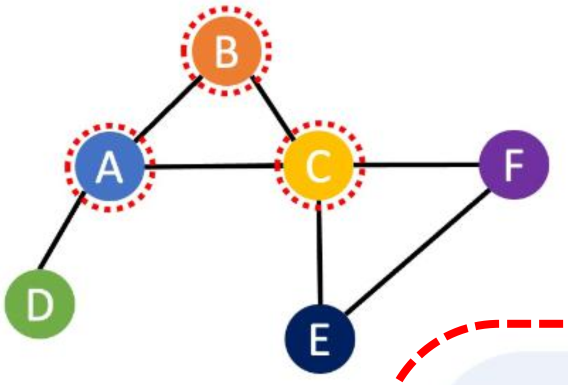
Node embedding matrix

(row-normalized) Adjacency matrix

Batch size = 3



Batch execution

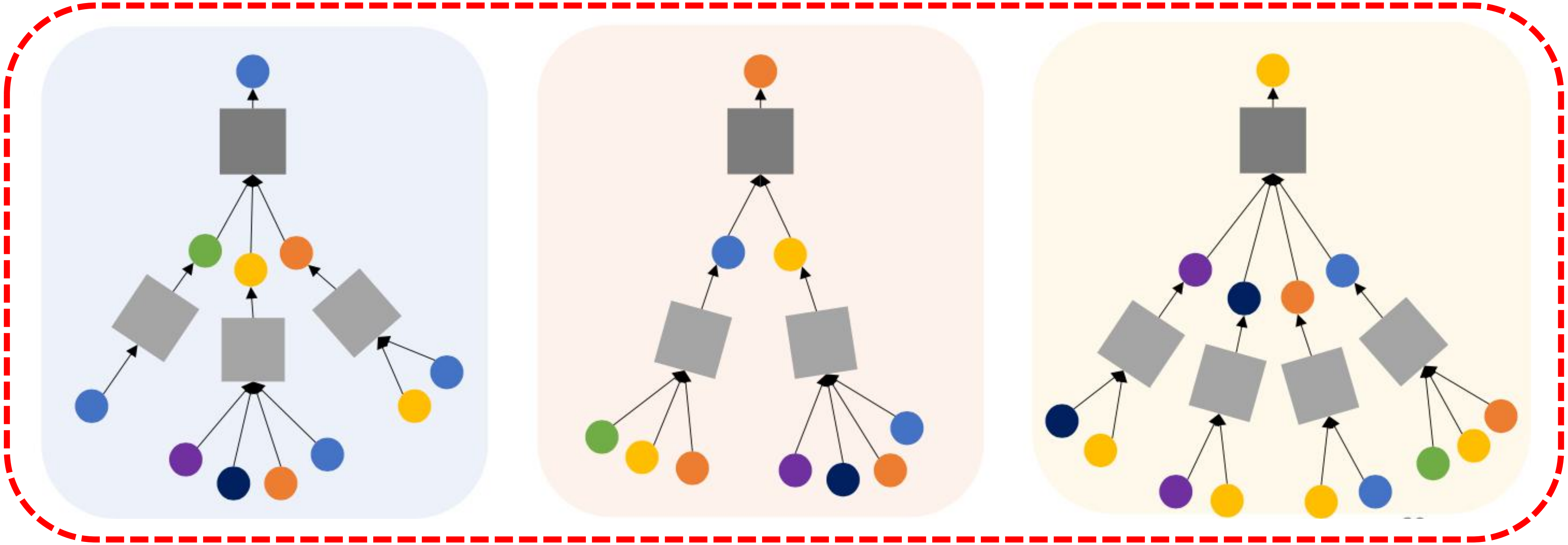


$$h_v^{(l)} = \sigma(\boldsymbol{W}^{(l)} \circ (\frac{1}{|\mathcal{N}(v)+1|} \Sigma_{u \in \mathcal{N}(v) \cup \{v\}} h_u^{(l-1)}))$$

$$\boldsymbol{H}^{(l)} = \sigma(\boxed{\widetilde{(\boldsymbol{A}+\boldsymbol{I})}}\boldsymbol{H}^{(l-1)}\boxed{\boldsymbol{W}^{(l)}})$$

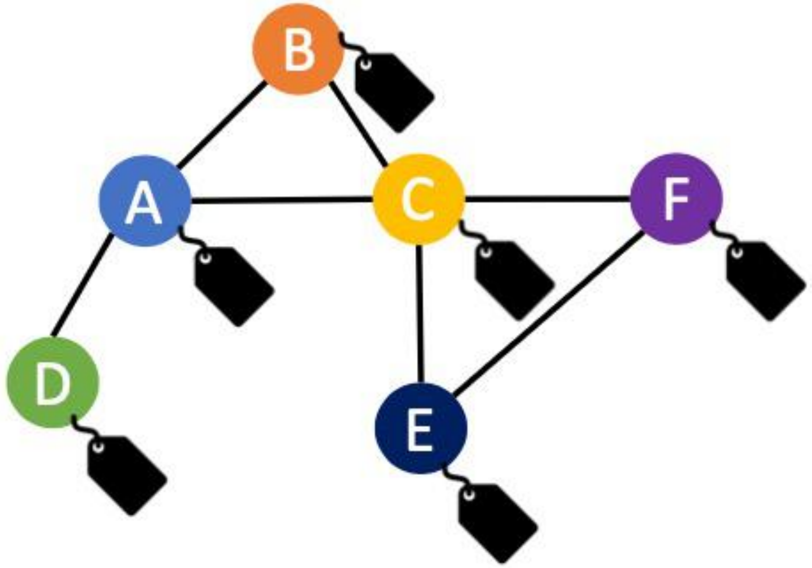
Fixed Trainable

Batch size = 3



Downstream tasks

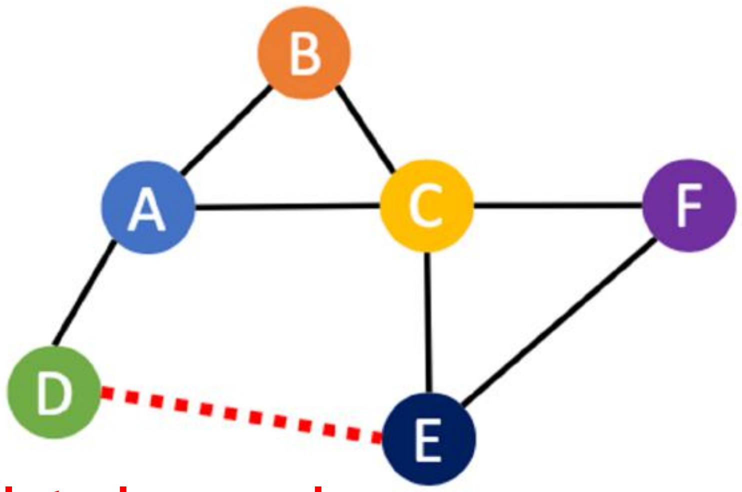
- Node-level prediction



Downstream tasks

- Node-level prediction

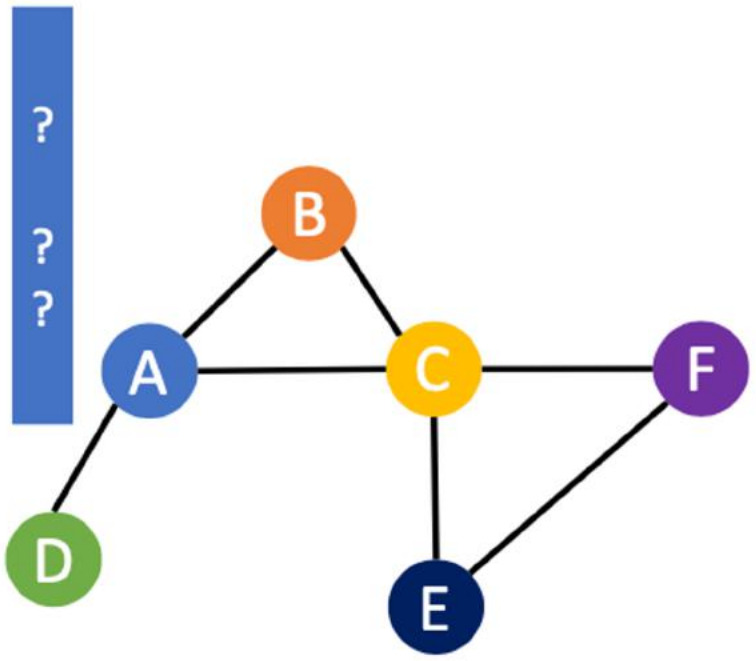
- Edge-level prediction



D and E are related enough
to be connected?

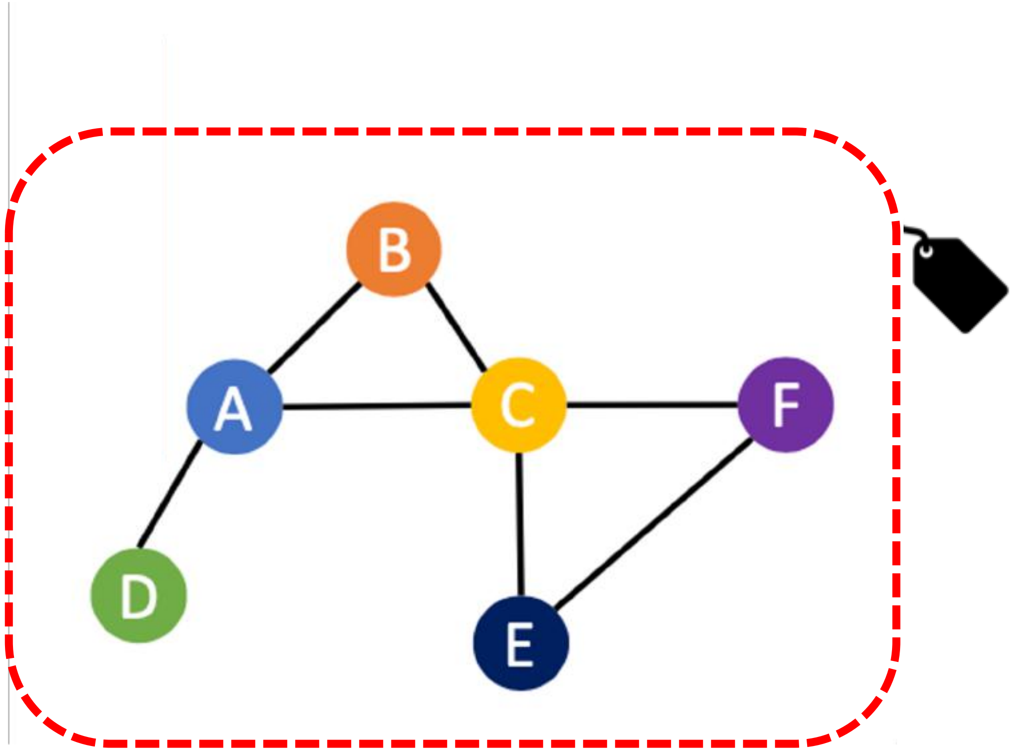
Downstream tasks

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



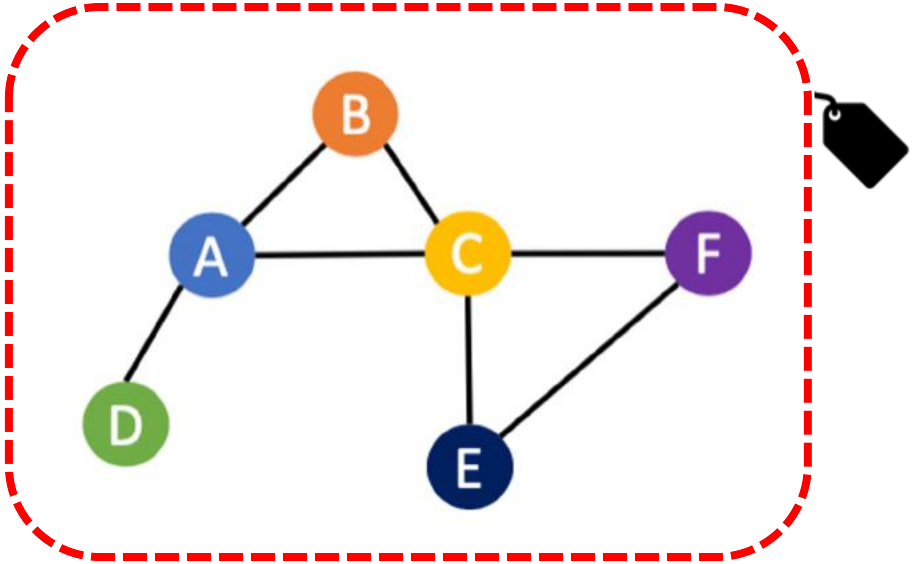
Downstream tasks

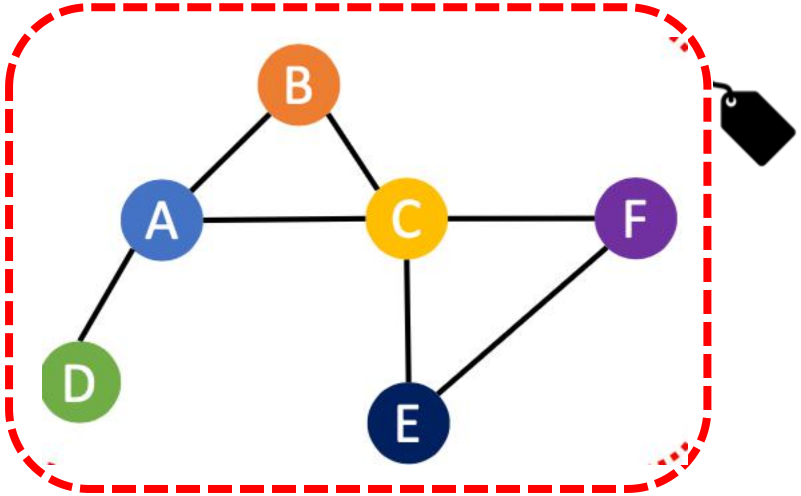
- Node-level prediction
- Edge-level prediction
- Attribute-level prediction
- Graph-level prediction

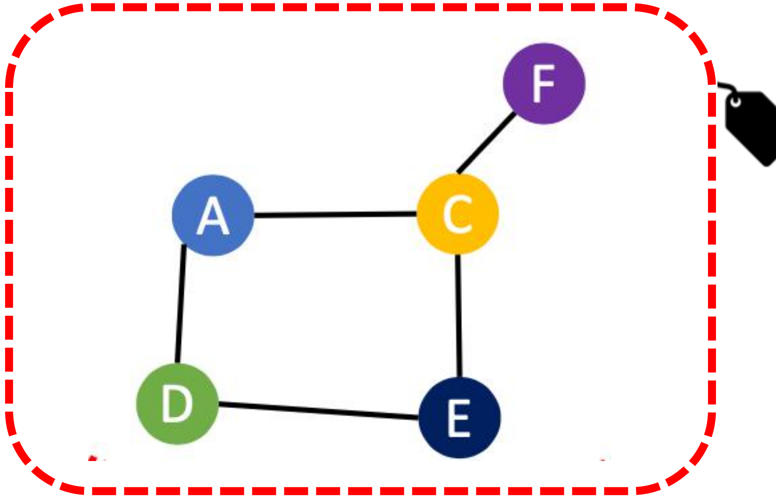


Downstream tasks

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

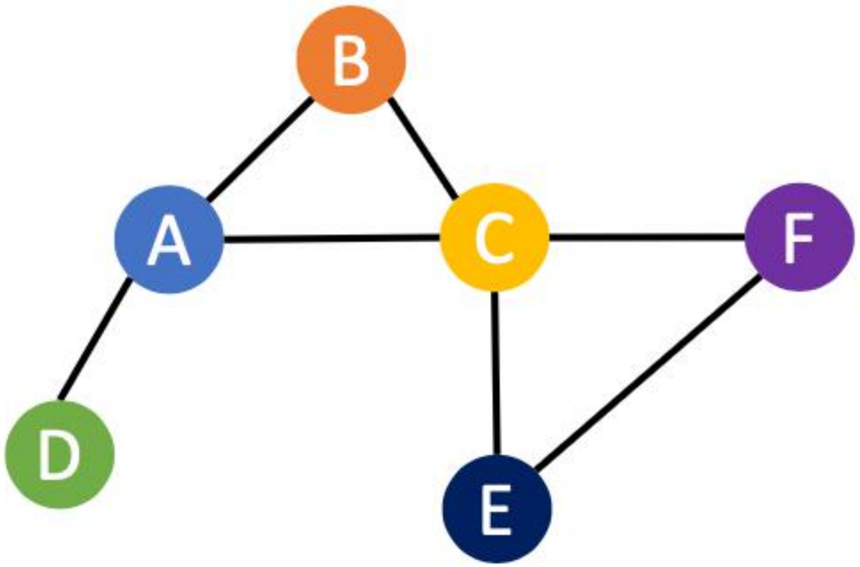




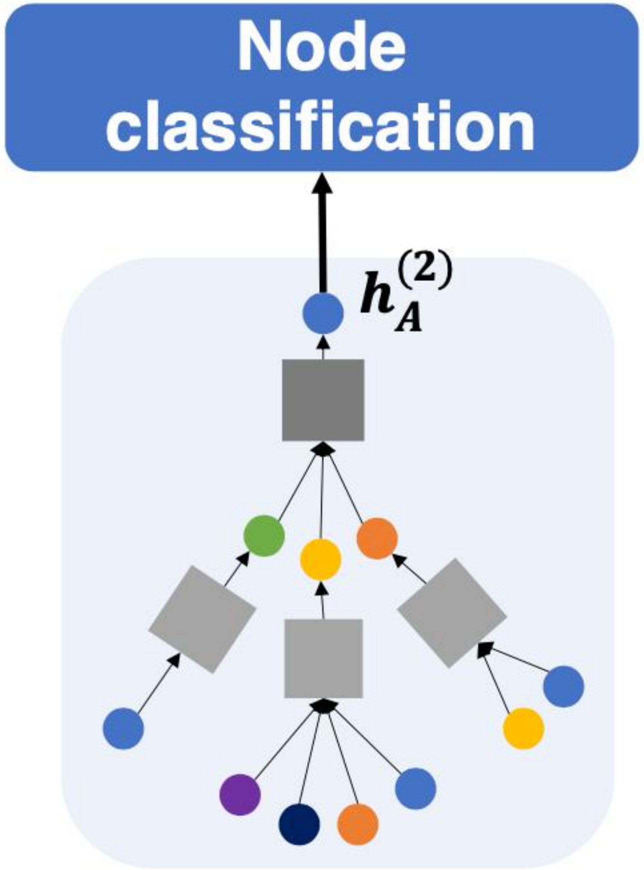


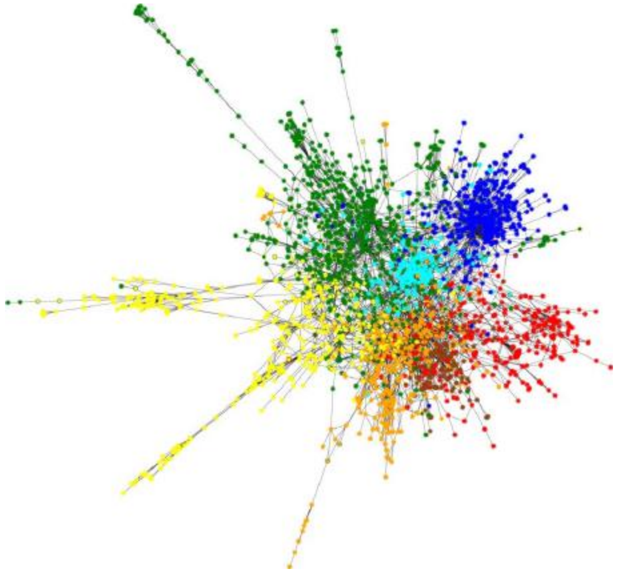
Downstream tasks

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



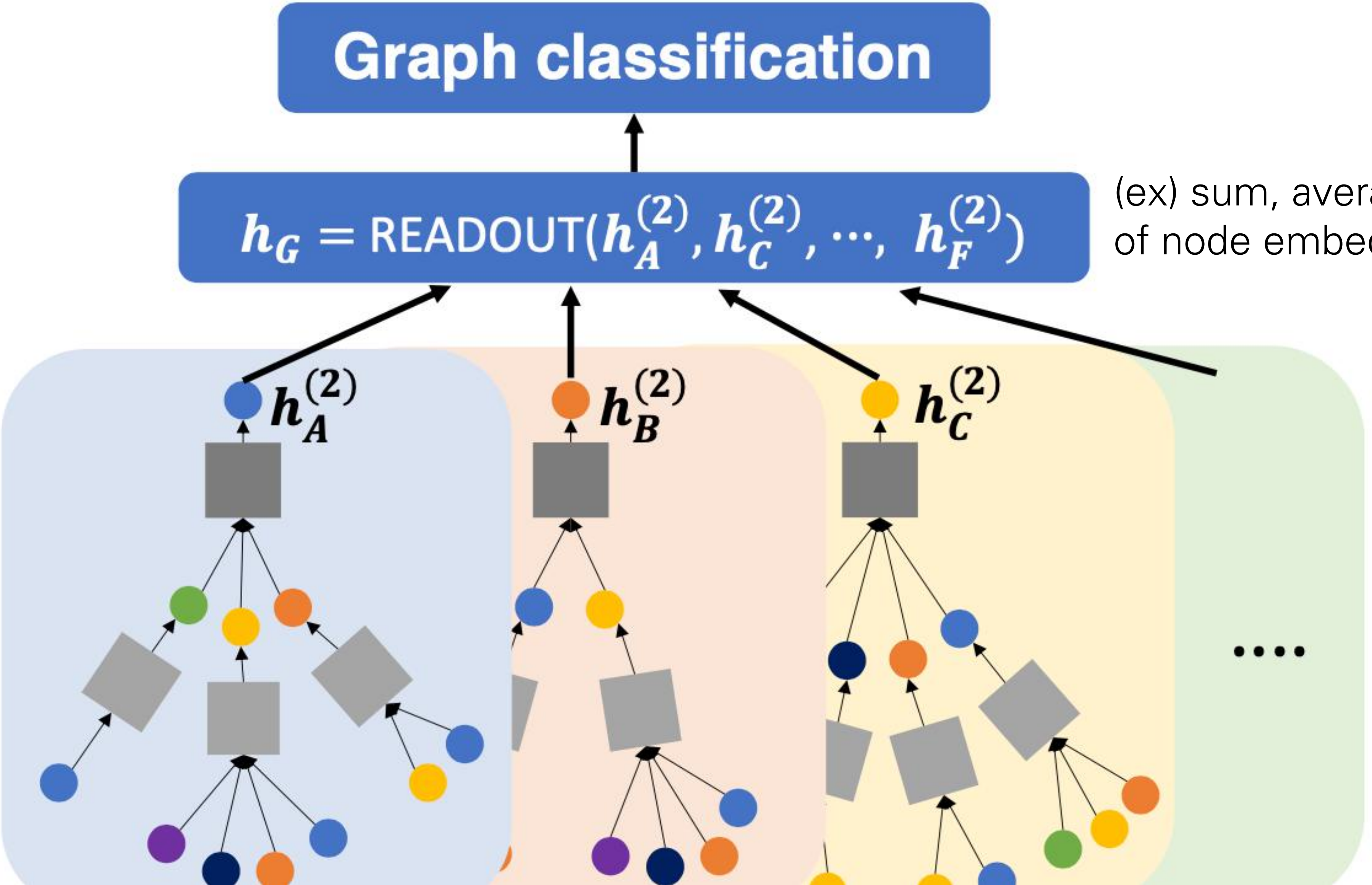
Node-level prediction tasks





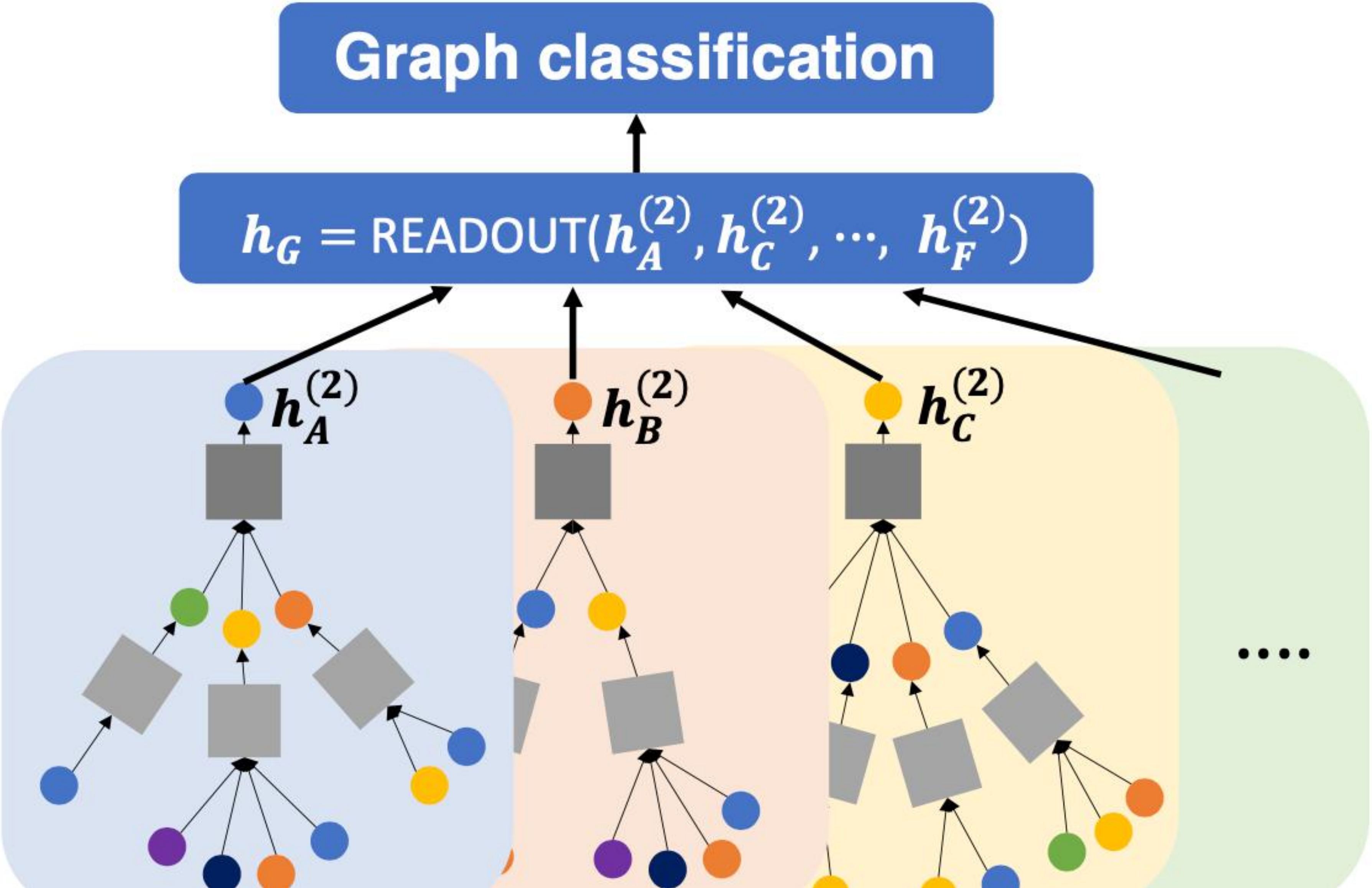
- Classify **papers** into topics on **citation networks**
- Cluster **posts** into subgroups on **Reddit networks**
- Classify **products** into categories on **Amazon co-purchase graphs**

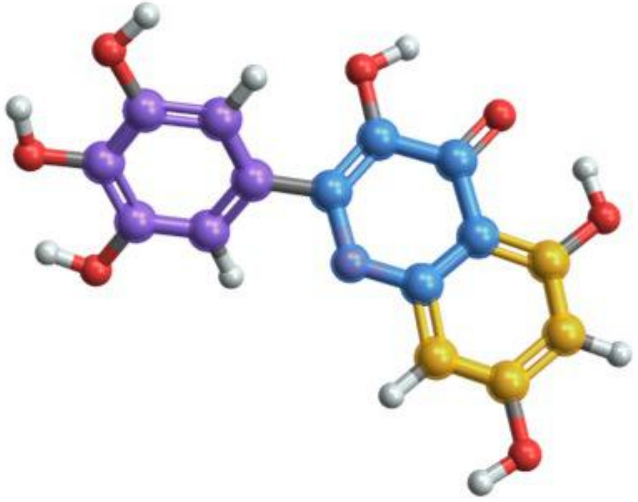
Graph-level prediction tasks



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

Graph-level prediction tasks

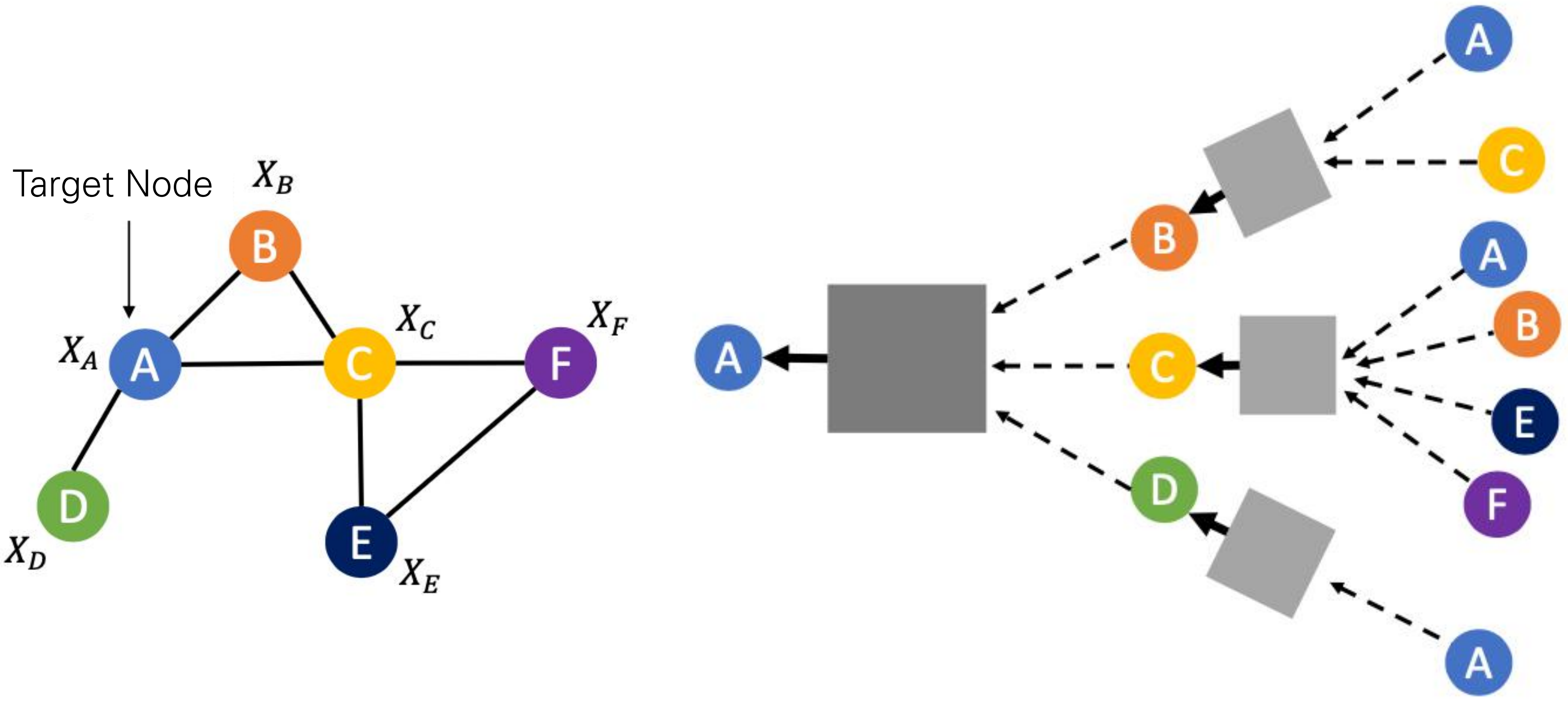




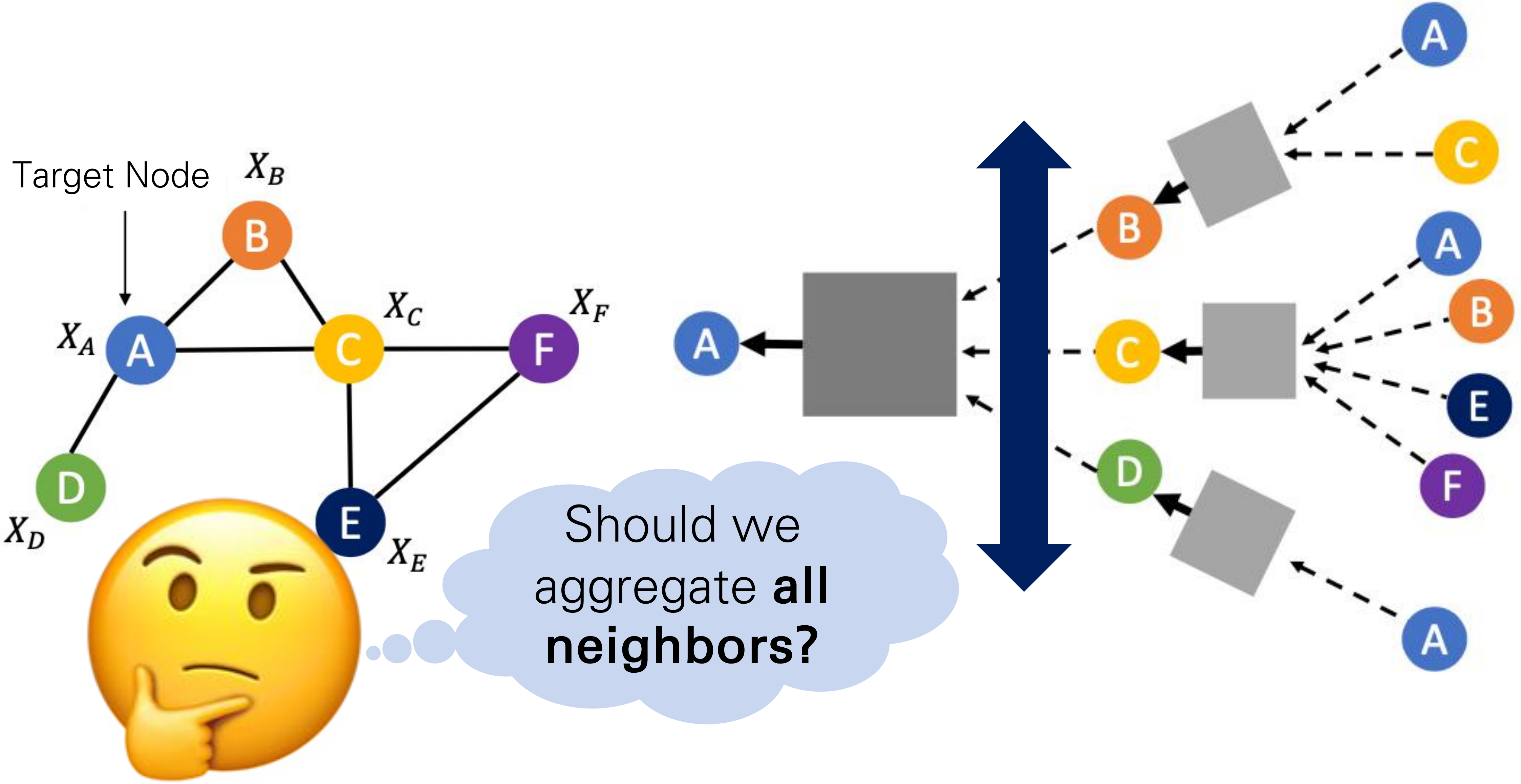
- Predict **properties of a molecule (graph)** where nodes are atoms and edges are chemical bonds

More on aggregation and Transformation operations

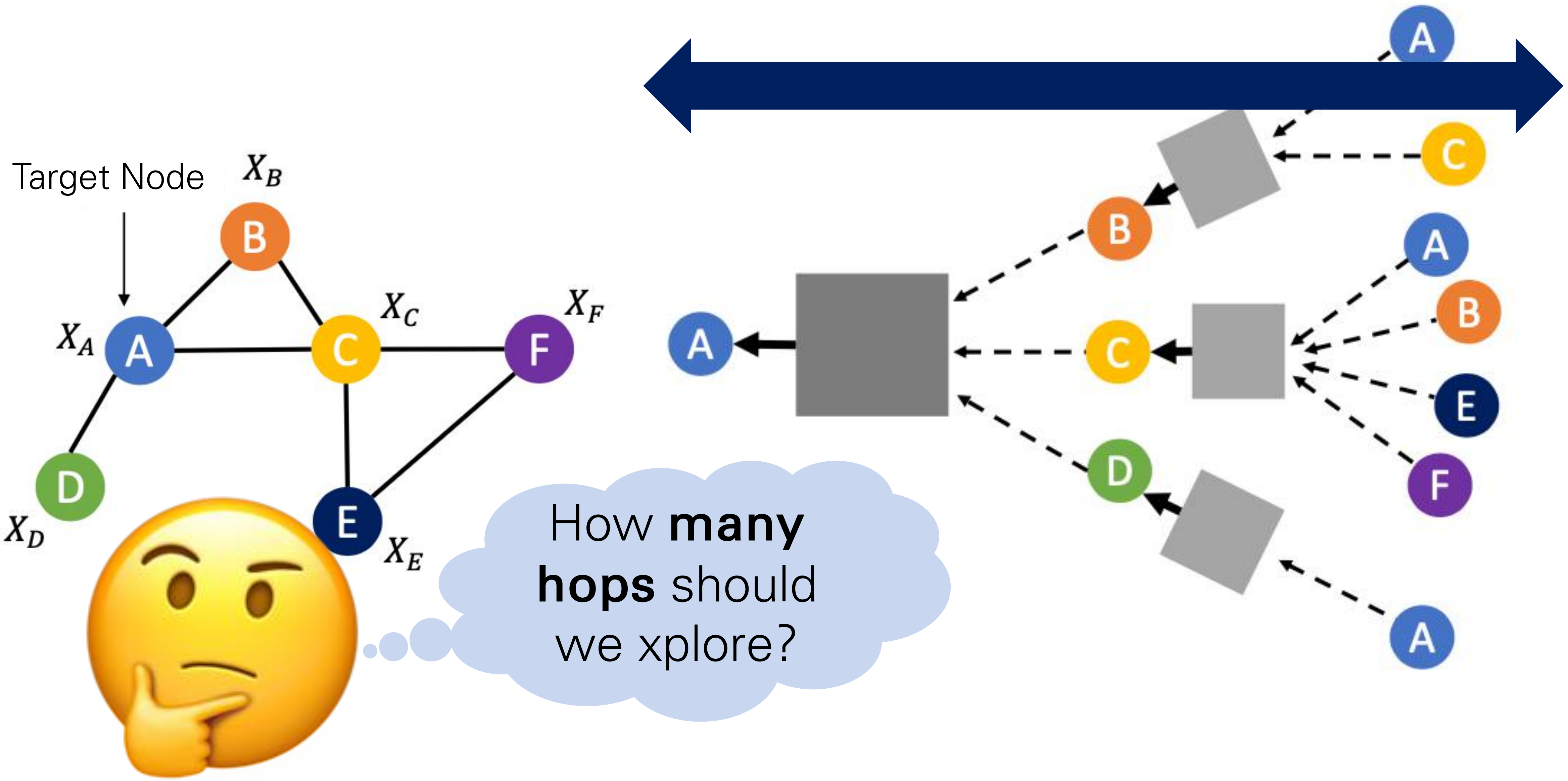
Graph Neural Networks – Width



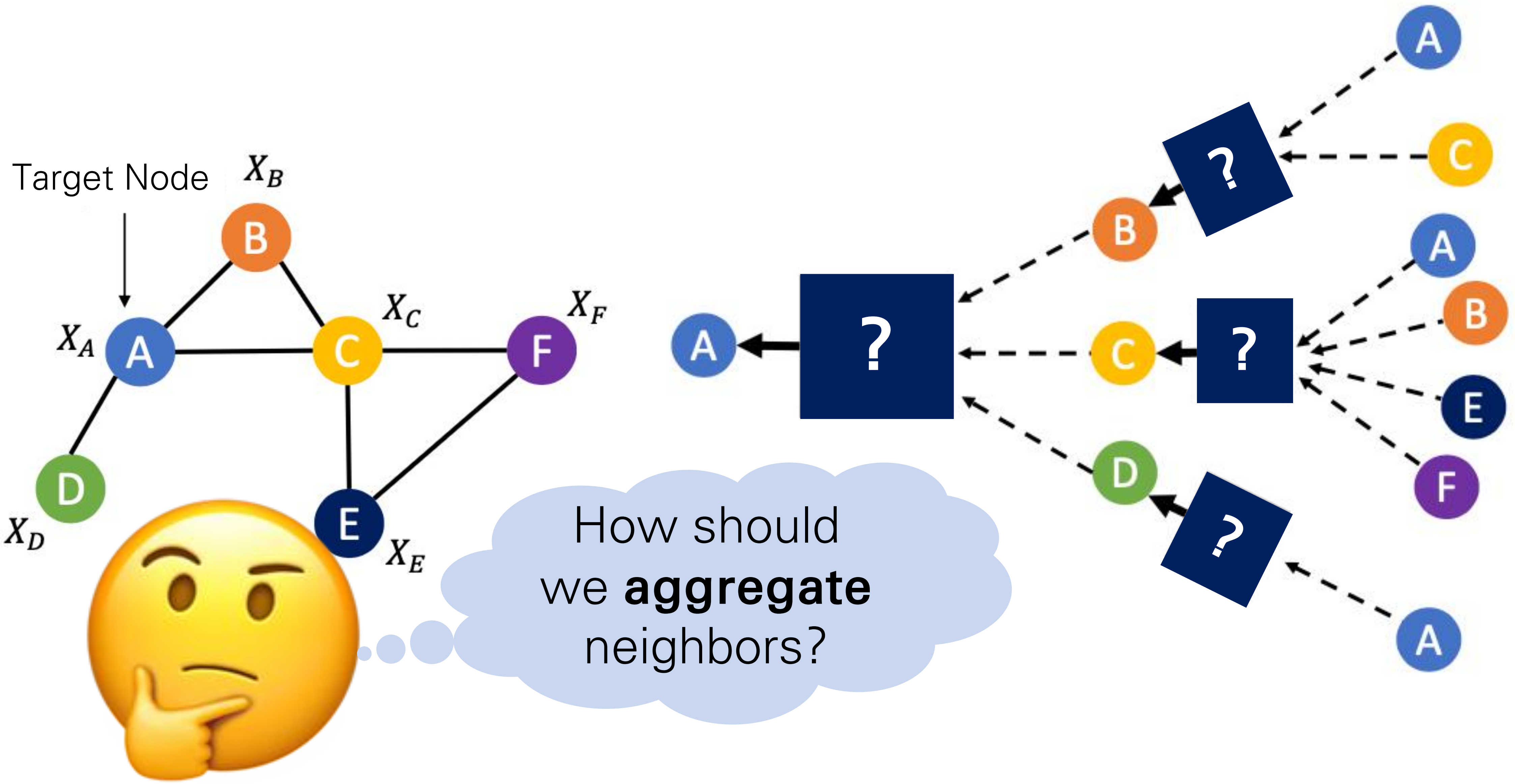
Graph Neural Networks (GNNs)



Graph Neural Networks (GNNs)

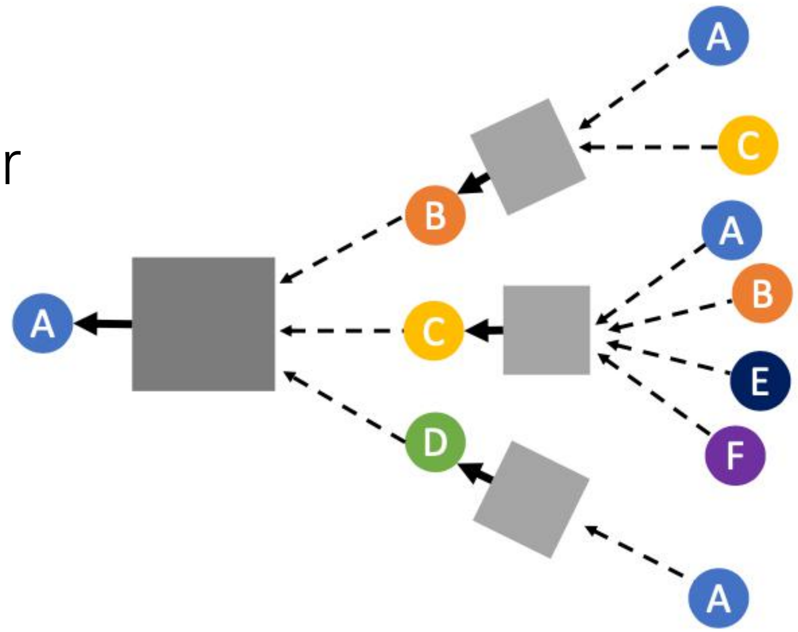


Graph Neural Networks (GNNs)



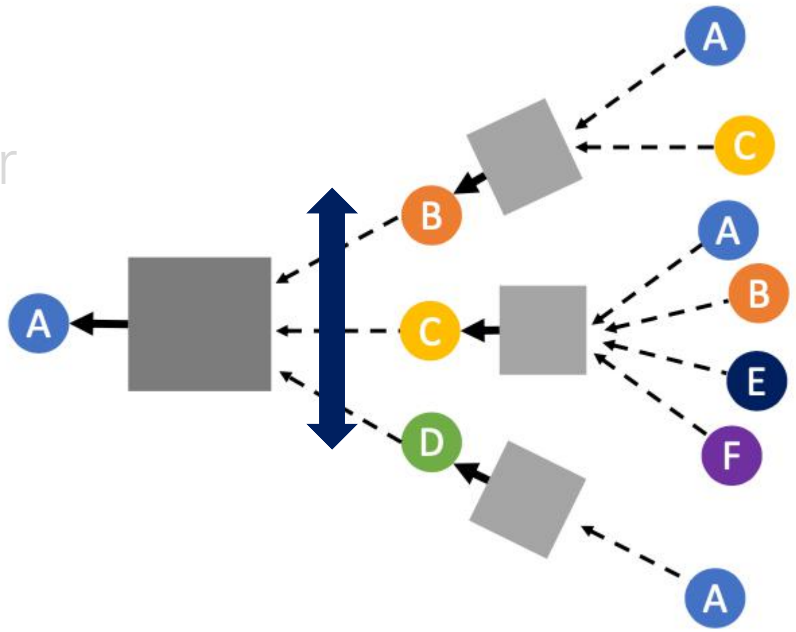
Graph Neural Network Architectures

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



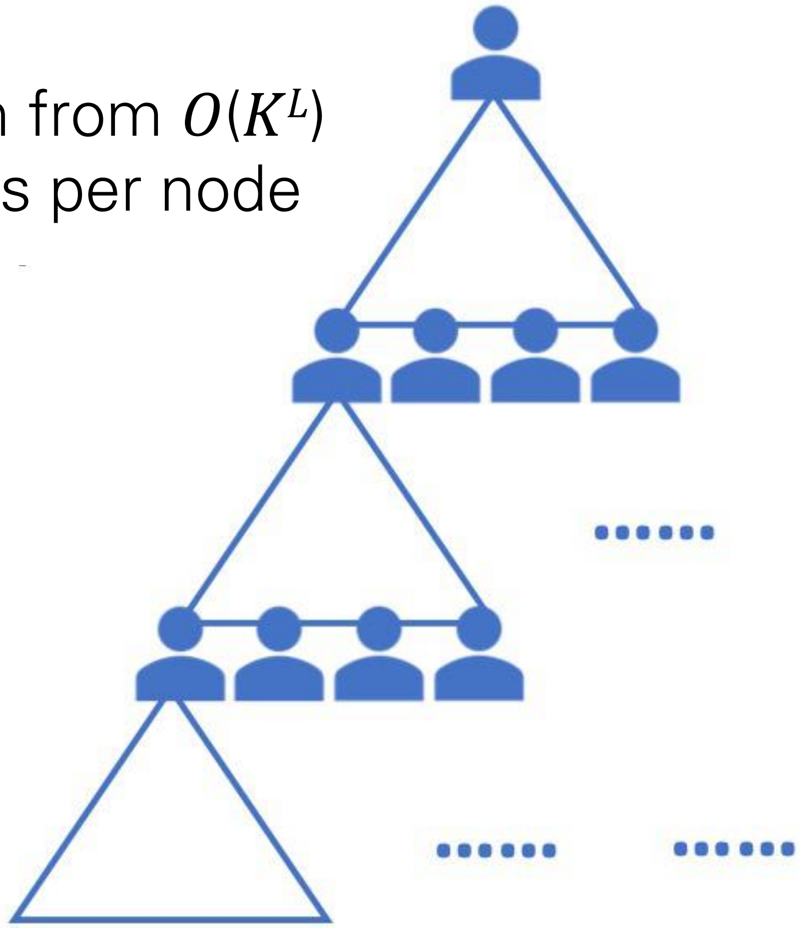
Graph Neural Network Architectures

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



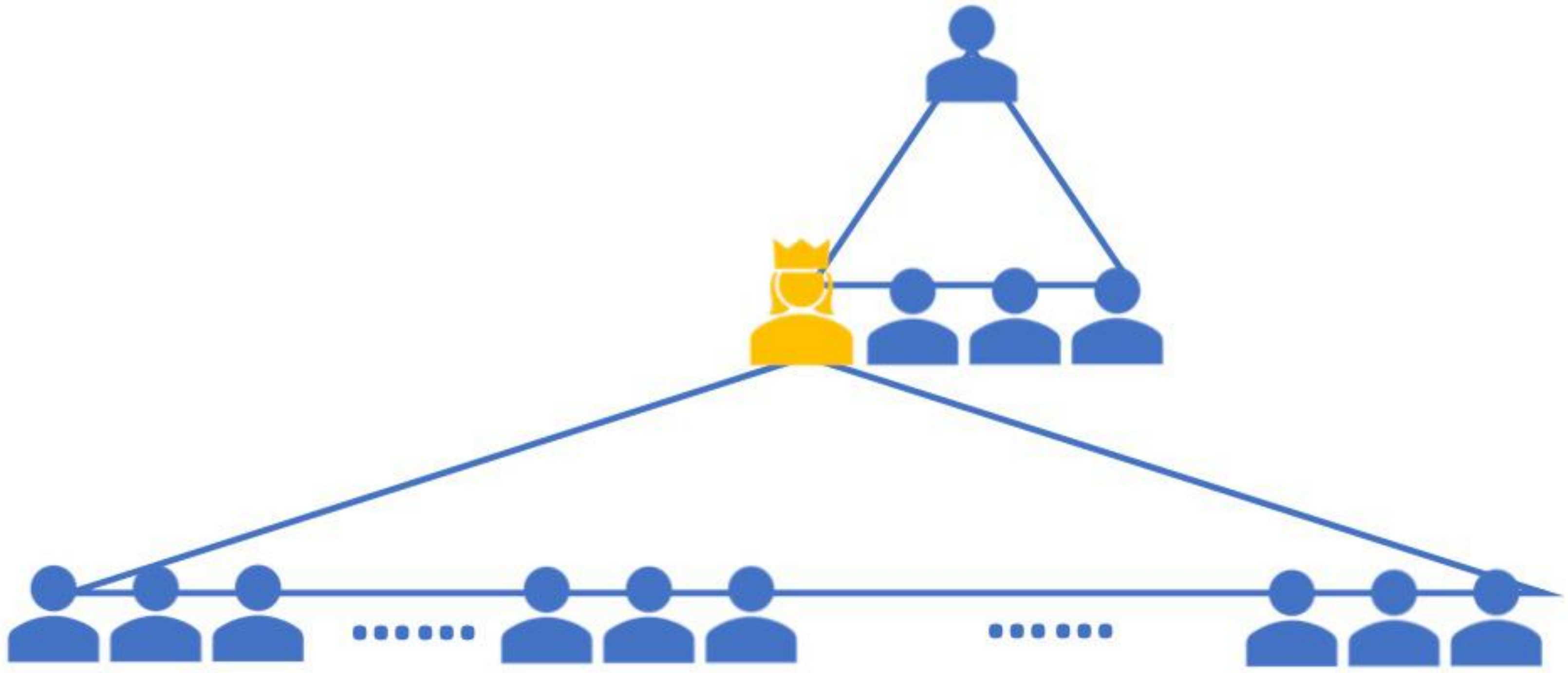
Aggregation Width in GNNs

- 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

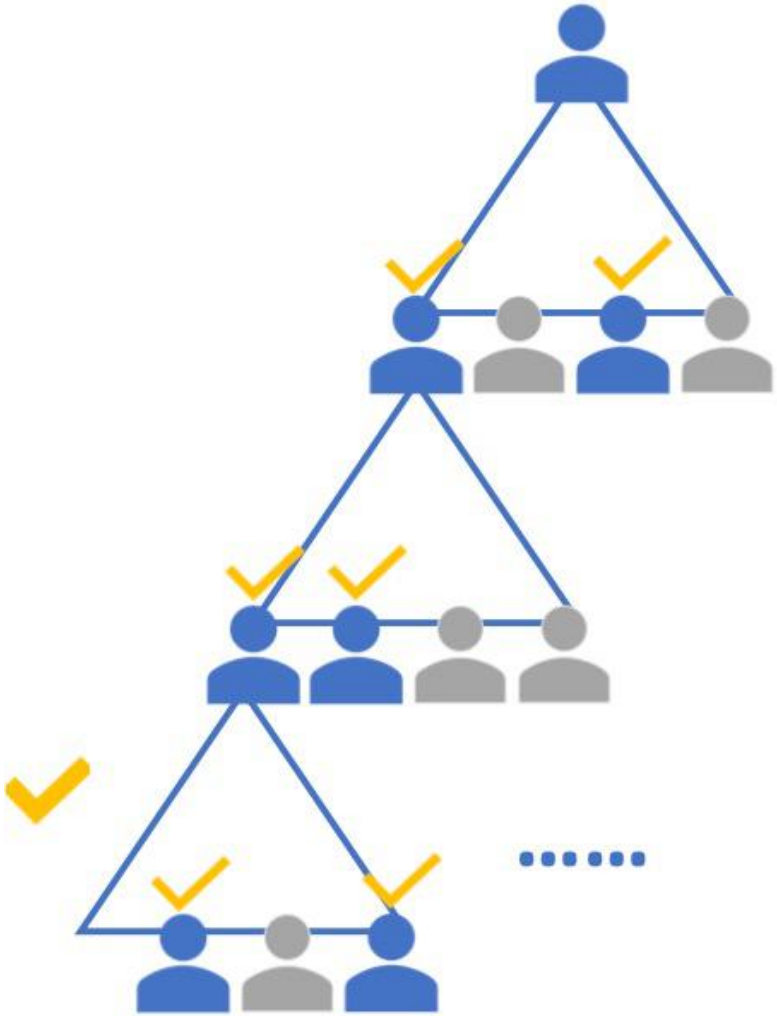
- If we aggregate all neighbors, GNNs have scalability issues
- Neighbor explosion
  - Hub nodes who are connected to a huge number of nodes



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[4]
- Importance sampling
  - Assign **different** sampling probabilities to all neighbors
  - FastGCN[5], LADIES[6], AS-GCN[7], GCN-BS[8], PASS[9]

[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

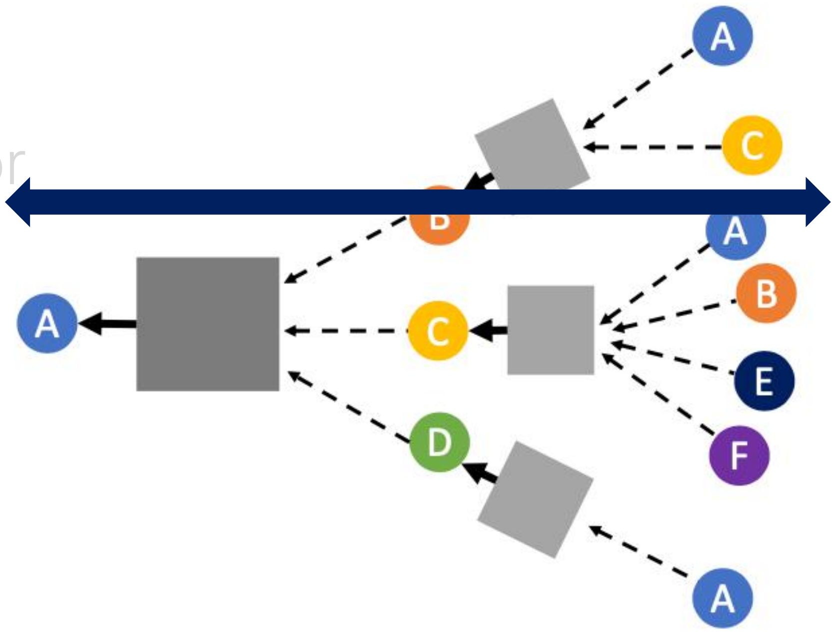
: assign higher sampling probabilities to neighbors who

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

[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”

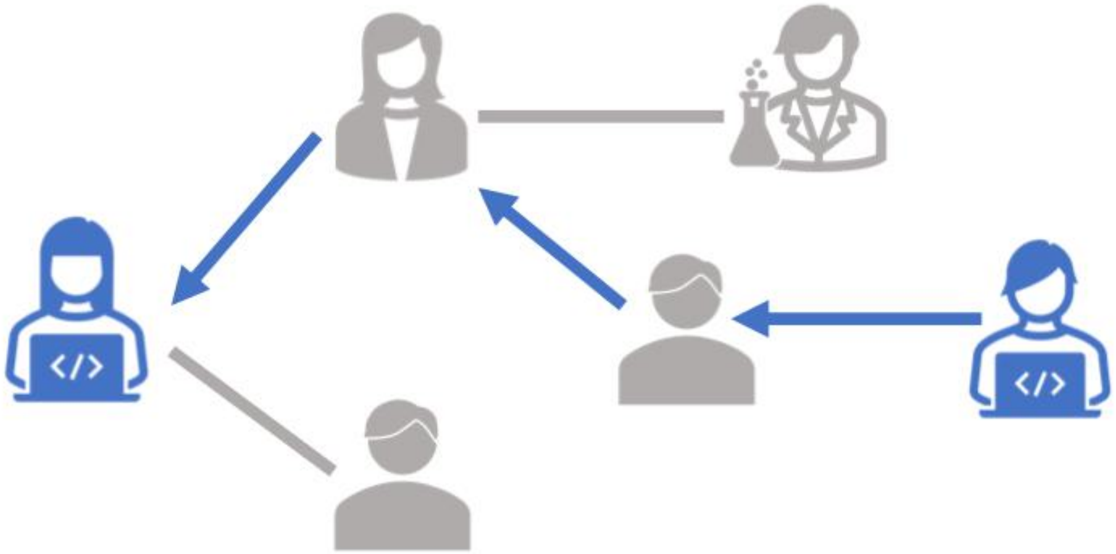
Graph Neural Network Architectures

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



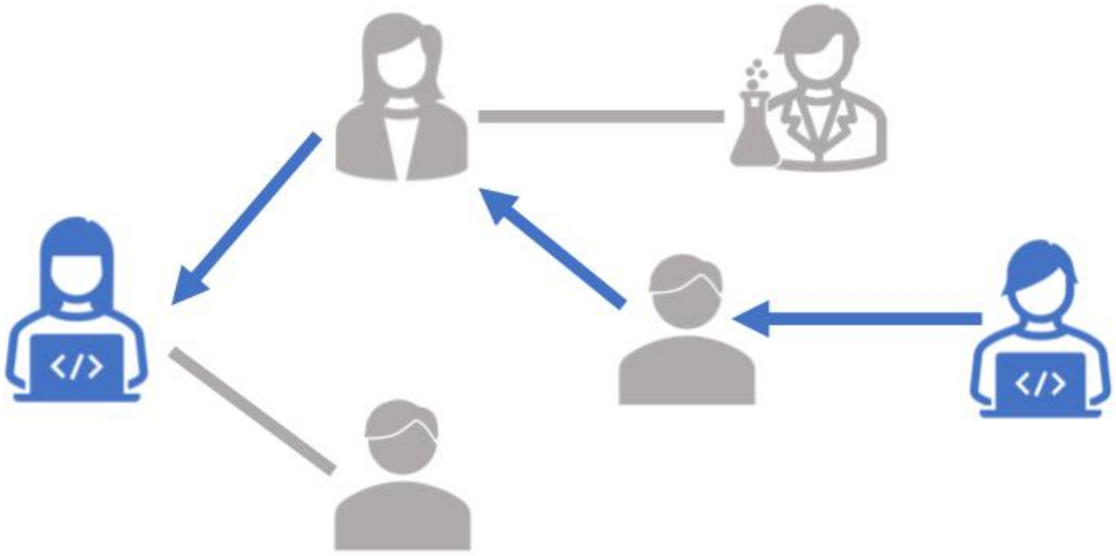
Aggregation Depth in GNNs

- Informative neighbors could be indirectly connected with a target node



Aggregation Depth in GNNs

- Informative neighbors could be indirectly connected with a target node
- Can't we just look multiple hops away from the target node?



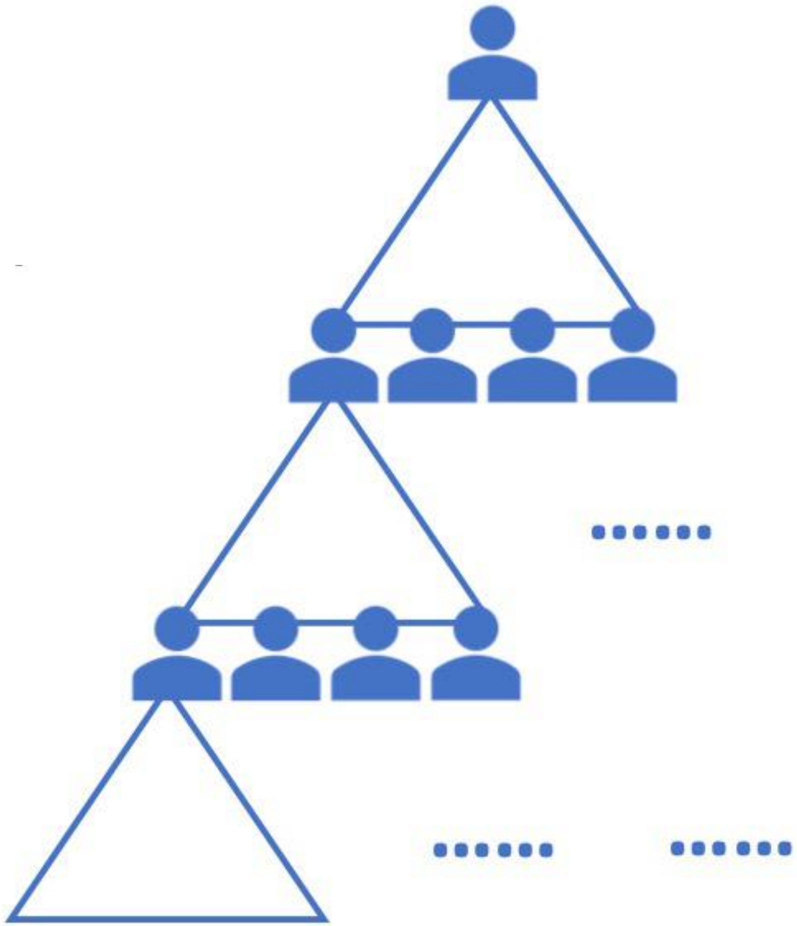
Aggregation Depth in GNNs

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

Wasn’t it Deeeep Learning?

Aggregation Depth in GNNs

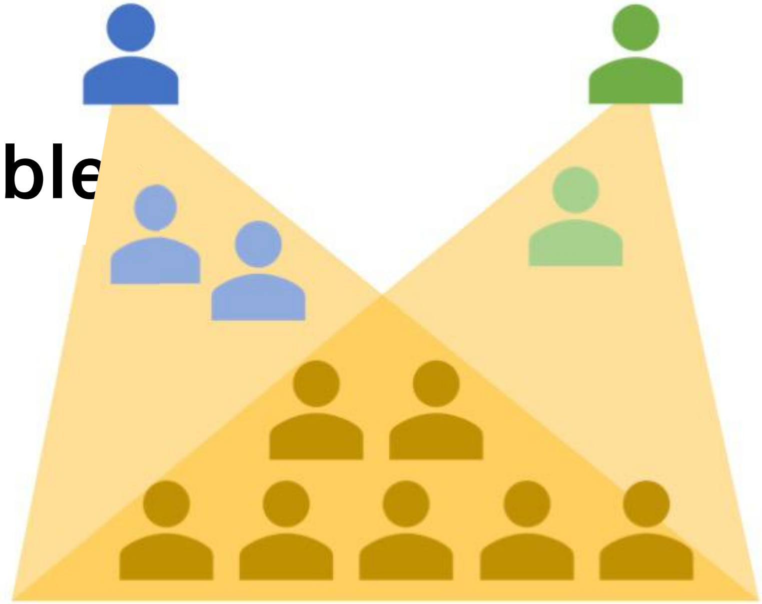
- When we increase the depth L more than this, GNNs face neighbor explosion $O(K^L)$
  - **Over-smoothing**
  - **Over-squashing**



Aggregation Depth in GNNs

Over-smoothing[10]

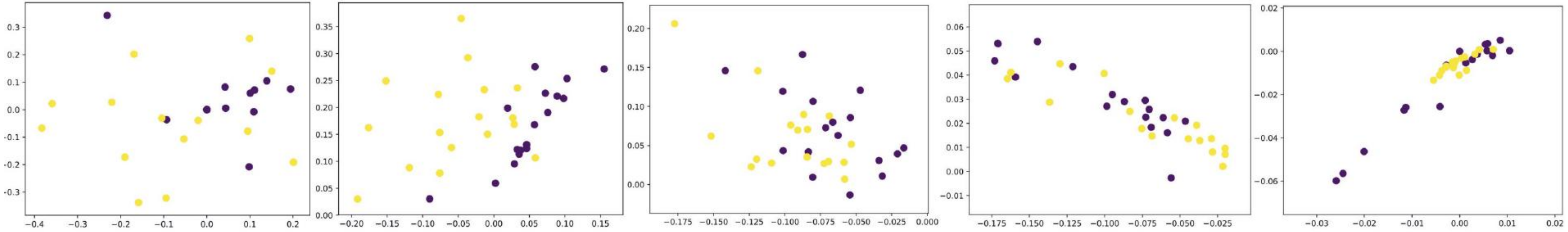
- When GNNs become deep, nodes share many neighbors
- Node embeddings become **indistinguishable**



[10] Qimai Li, et al. "Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning"

Aggregation Depth in GNNs

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



(a) 1-layer (b) 2-layer (c) 3-layer (d) 4-layer (e) 5-layer

[10] Qimai Li, et al. "Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning"

Aggregation Depth in GNNs

Mitigate over-smoothing

PairNorm[11]

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

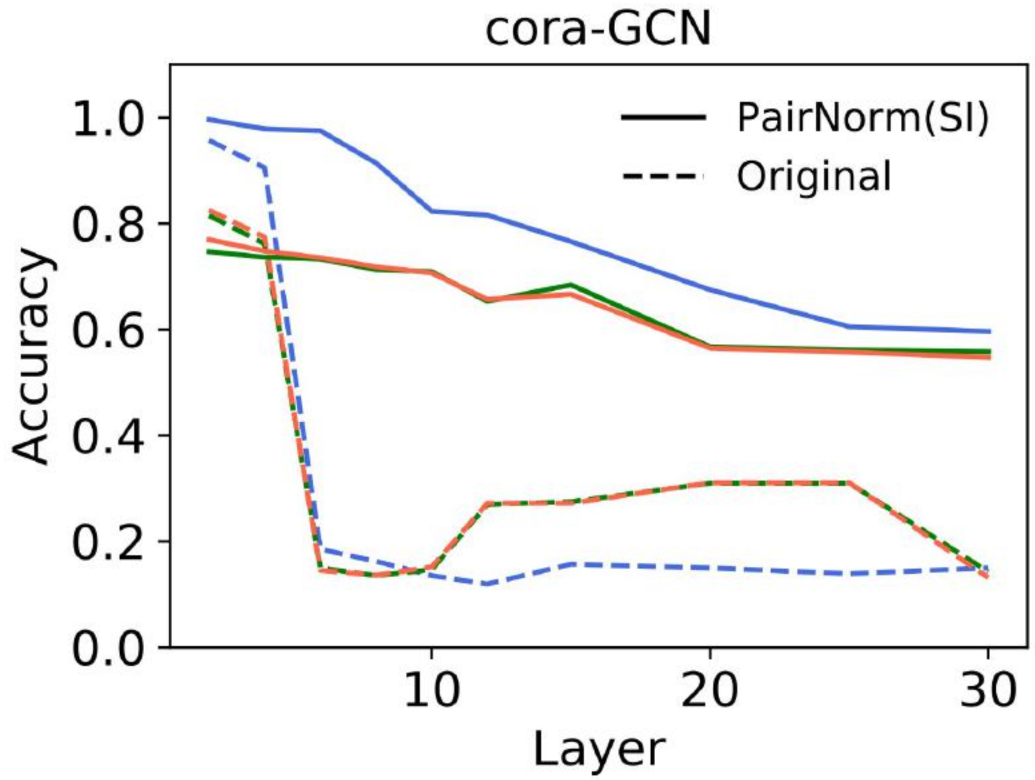
$$\mathrm{TPSD}(\dot{X}) = \overbrace{\boxed{\sum_{(i,j)\in\mathcal{E}} ||\dot{x}_i - \dot{x}_j||_2^2}}^{\text{Connected pairs}} + \overbrace{\boxed{\sum_{(i,j)\notin\mathcal{E}} ||\dot{x}_i - \dot{x}_j||_2^2}}^{\text{Disconnected pairs}} = \boldsymbol{C}$$

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

Aggregation Depth in GNNs

Mitigate over-smoothing

PairNorm[11]

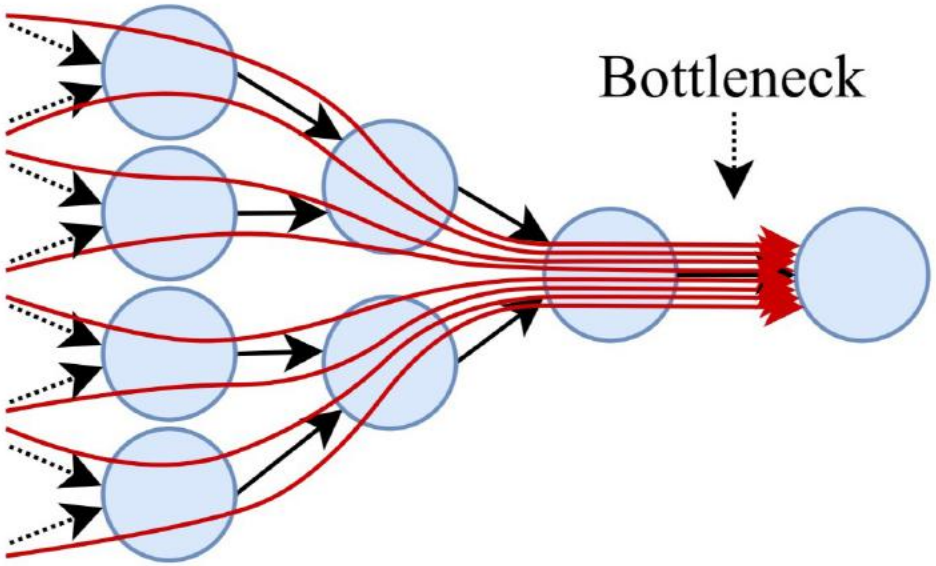


[11] Lingxiao Zhao, et al. "PairNorm: Tackling Oversmoothing in GNNs"

Aggregation Depth in GNNs

Over-squashing[12]

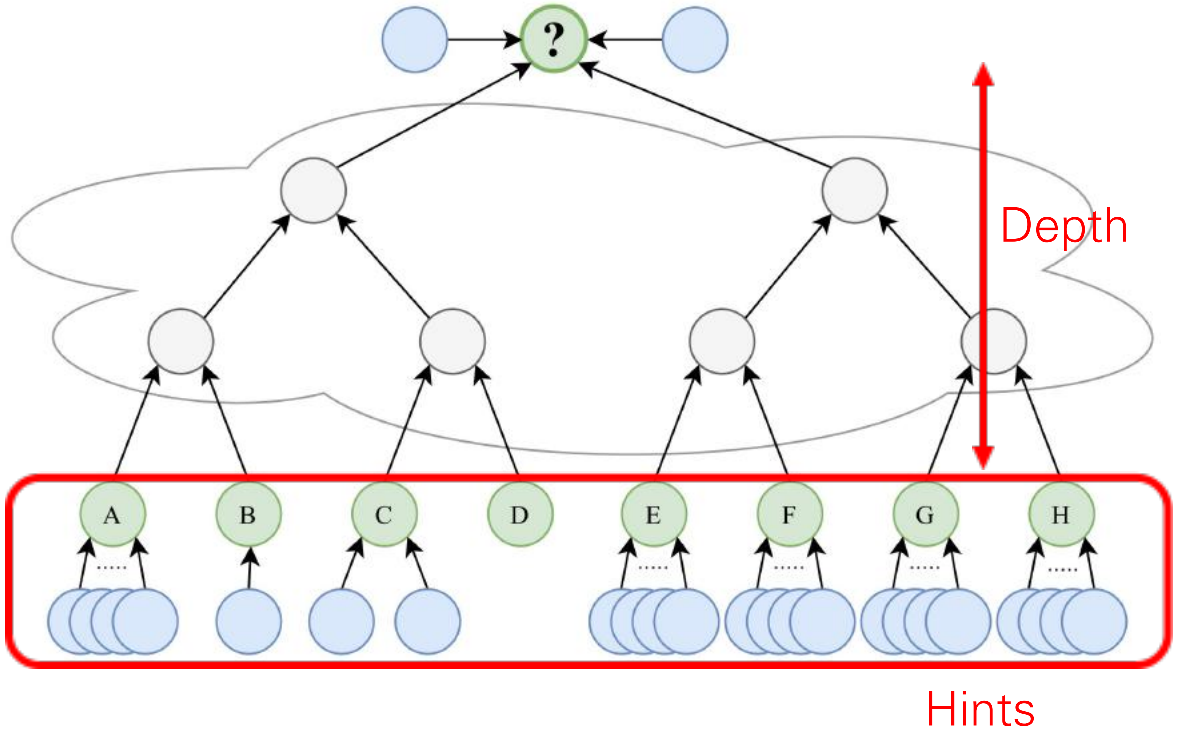
- A node's exponentially-growing neighborhood is compressed into a fixed-size vector

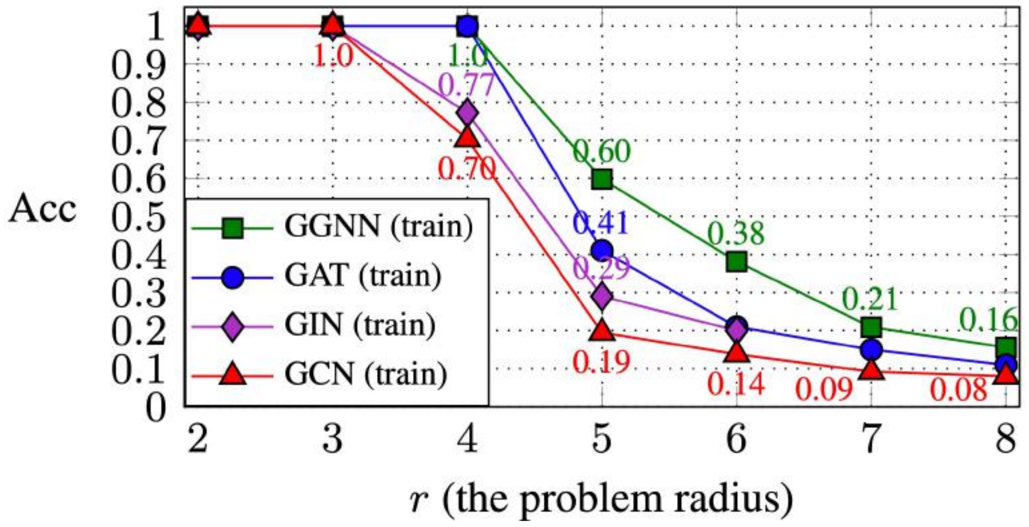


[12] Uri Alon, et al. "On the Bottleneck of Graph Neural Networks and its Practical Implications"

Aggregation Depth in GNNs

Over-squashing[12]





[12] Uri Alon, et al. "On the Bottleneck of Graph Neural Networks and its Practical Implications"

Aggregation Depth in GNNs

Decoupling the two concepts of depths in GNNs[13]

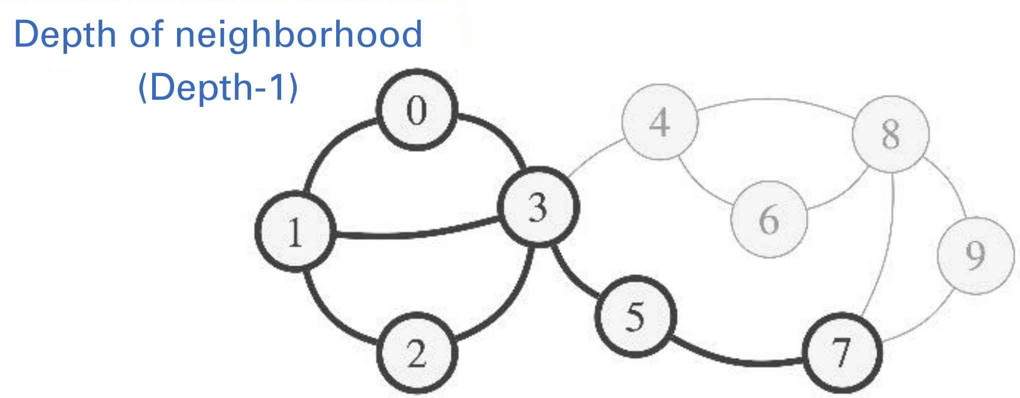
- **Depth-1**: neighborhood that each node aggregates information from
- **Depth-2**: number of layers in GNNs

[13] Hanqing Zeng, et al. "Decoupling the Depth and Scope of Graph Neural Networks"

Aggregation Depth in GNNs

Decoupling the two concepts of depths in GNNs[13]

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



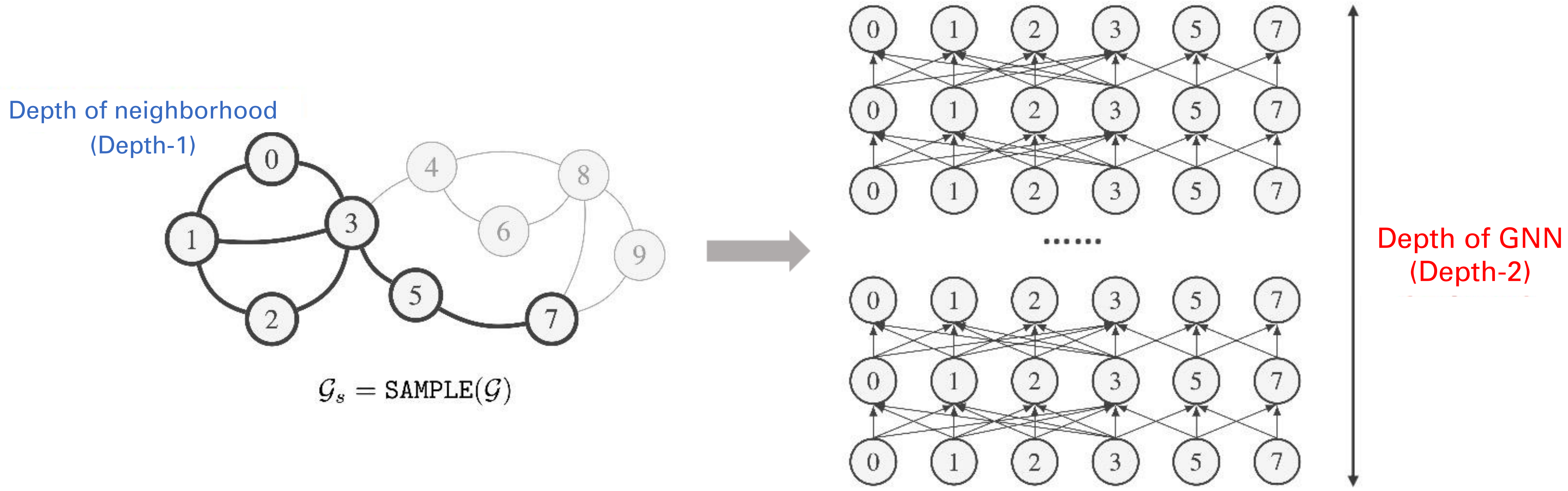
$\mathcal{G}_s = \texttt{SAMPLE}(\mathcal{G})$

[13] Hanqing Zeng, et al. "Decoupling the Depth and Scope of Graph Neural Networks"

Aggregation Depth in GNNs

Decoupling the two concepts of depths in GNNs[13]

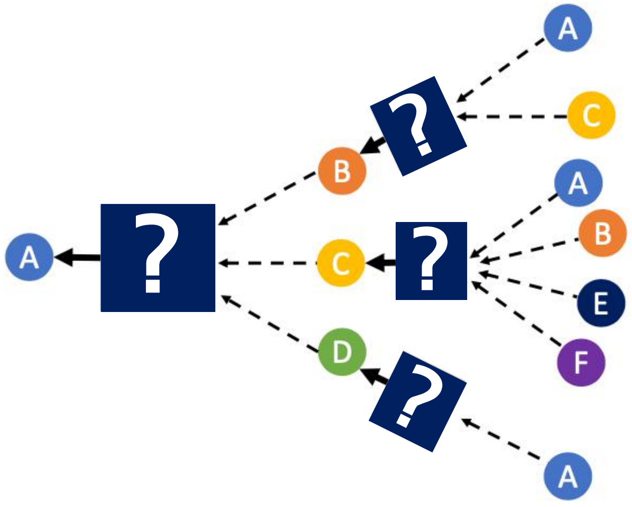
- **Depth-1**: neighborhood that each node aggregates information from
- **Depth-2**: number of layers in GNNs



[13] Hanqing Zeng, et al. "Decoupling the Depth and Scope of 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



Aggregation strategy in GNNs

In each layer l :

Aggregate over neighbors

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

Transform messages

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

Aggregation strategy in GNNs

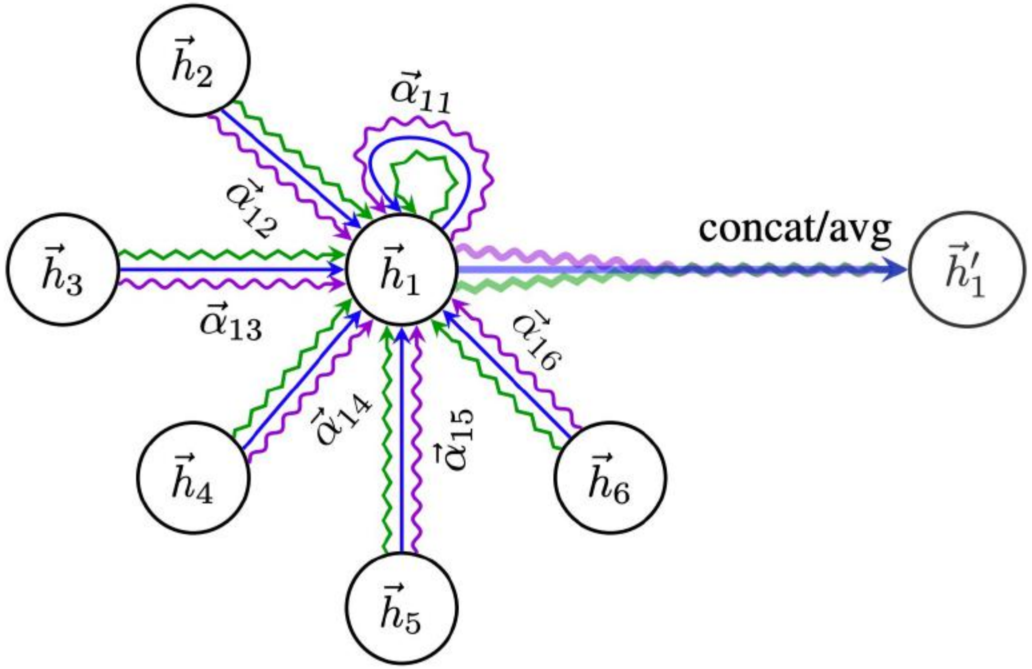
- GCN[1]
  - Average embeddings of neighboring nodes

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

Aggregation strategy in GNNs

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

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



[14] Petar Veličković., et al. "GRAPH ATTENTION NETWORKS."

Aggregation strategy in GNNs

In each layer l :

Aggregate over neighbors

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

Core part of GNNs

Transform messages

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

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

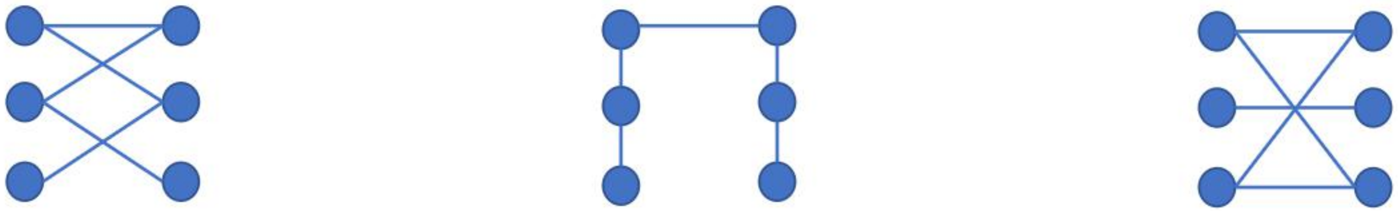
- By measuring the power of GNNs, we can find the best aggregation strategy!!

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

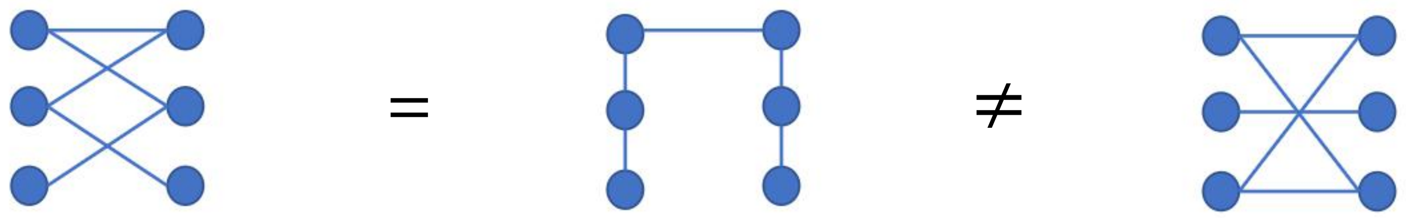
- How powerful are Graph Neural Networks?[2]
- Metric
  - Graph-level prediction task
  - Can a GNN model distinguish two non-isomorphic graphs?



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

Aggregation strategy in GNNs

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



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

Aggregation strategy in GNNs

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

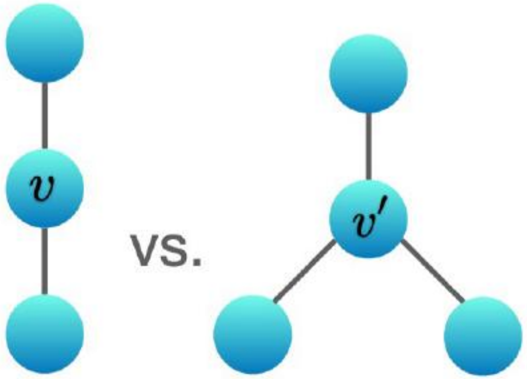
Weisfeiler-Lehman (WL) graph isomorphism test

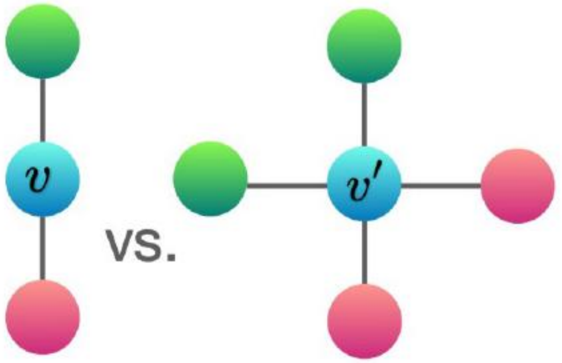
$$\boldsymbol{f(x_1) = f(x_2) \Rightarrow x_1 = x_2}$$

[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?[2]
  - Any aggregation-based GNN is at most as powerful as the **WL test**[15]
  - Maximum power = aggregation strategy is injective
  - (ex) summation





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

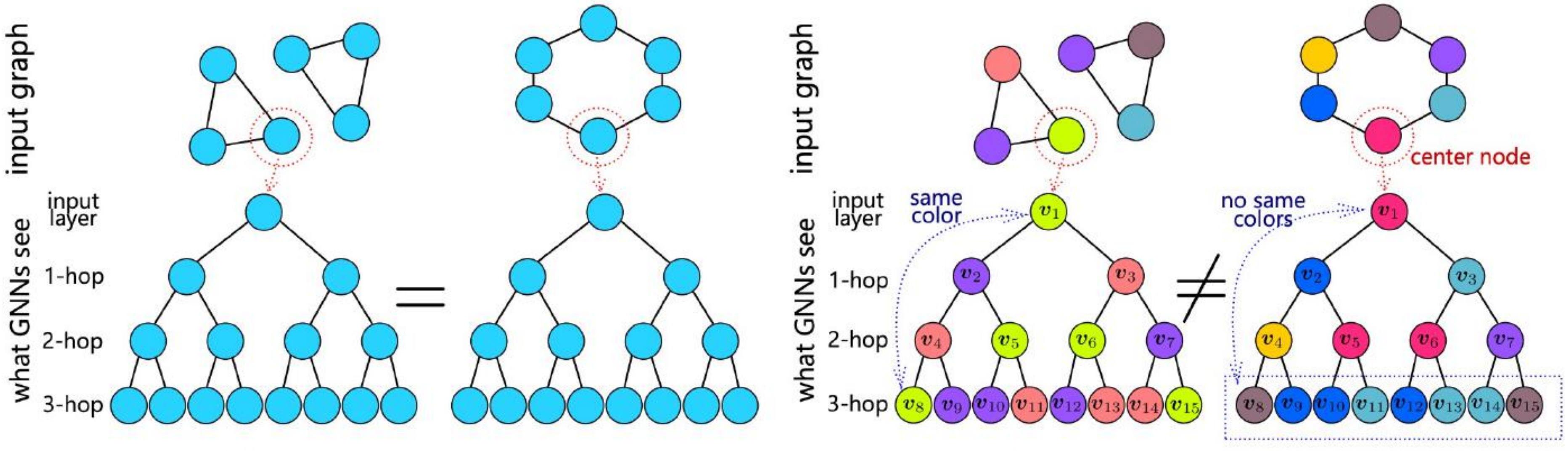
[2] Keyulu Xu., et al. "How Powerful are Graph Neural Networks?"

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[16]

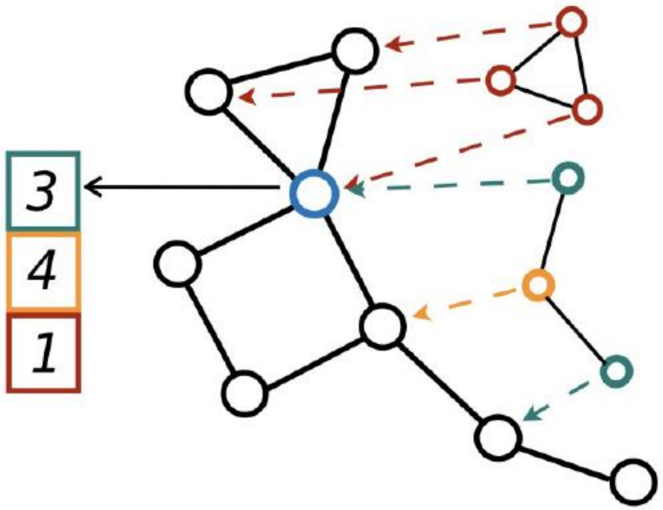


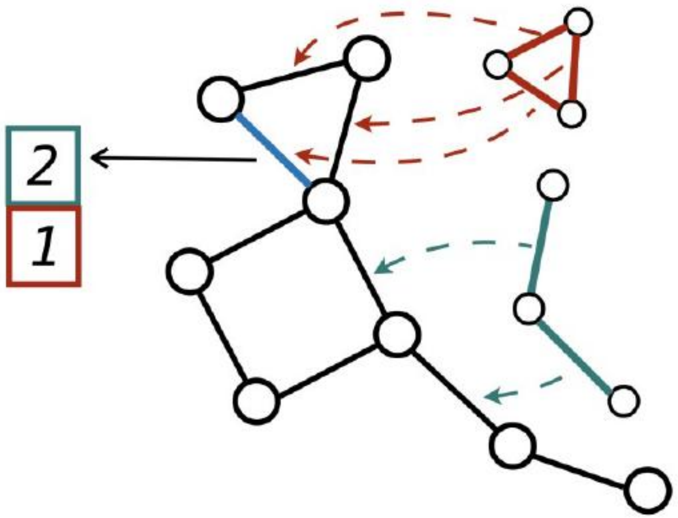
(a) Identical Features.　(b) Random Features.

[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[16]

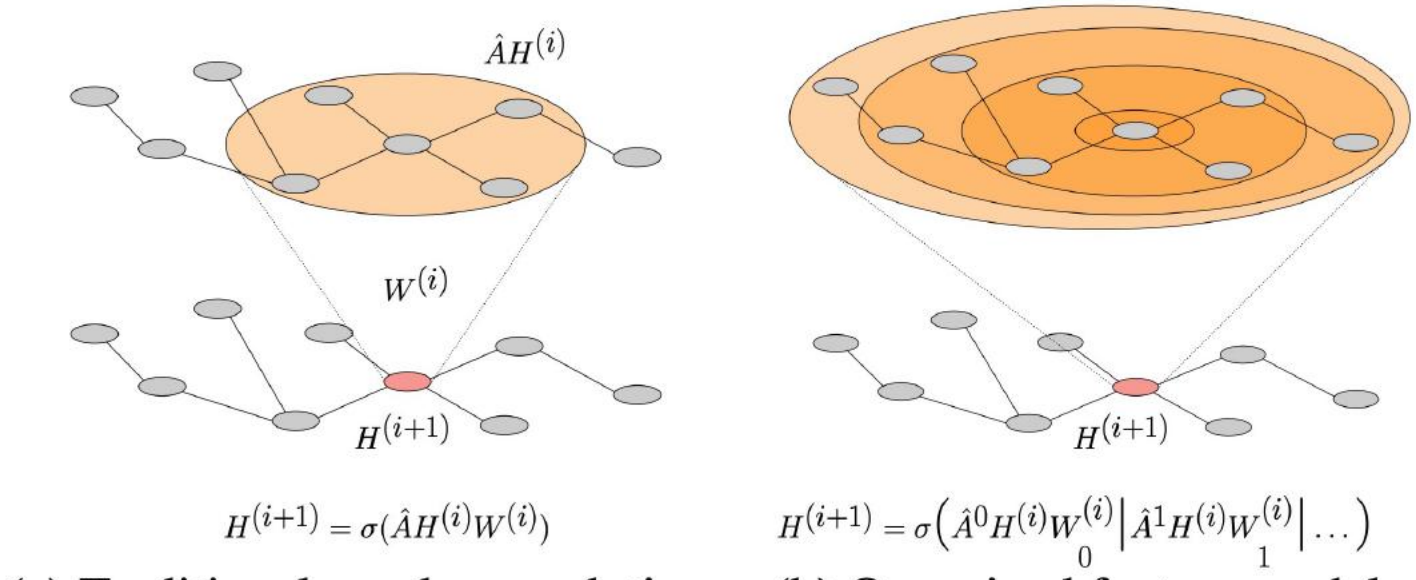




[17] Giorgos Bouritsas, et al. "Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting"

Aggregation strategy in GNNs

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

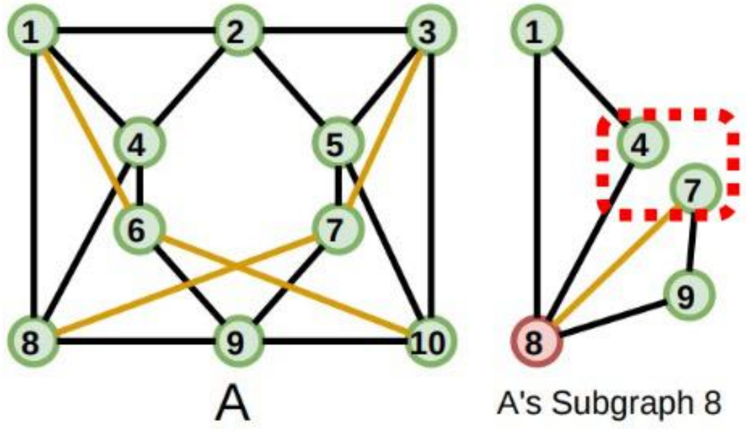


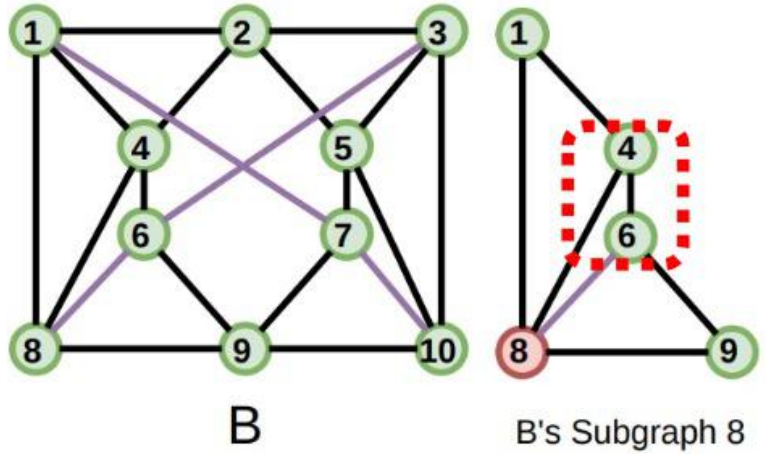
(a) Traditional graph convolution. (b) Our mixed feature model.

[18] Sami Abu-El-Haija, et al. "MixHop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing"

Aggregation strategy in GNNs

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





[19] Lingxiao Zhao, et al. "From Stars to Subgraphs: Uplifting Any GNN with Local Structure Awareness"

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.*

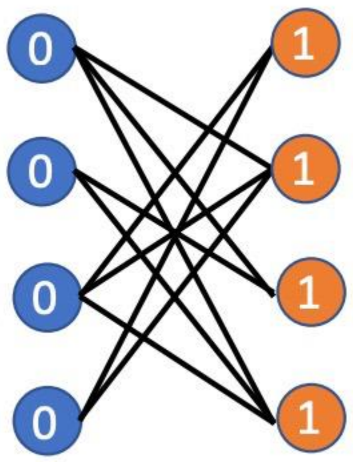
[20] Gabriele Corso, et al. "Principal Neighbourhood Aggregation for Graph Nets"

Aggregation strategy in GNNs

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

Aggregation strategy in GNNs

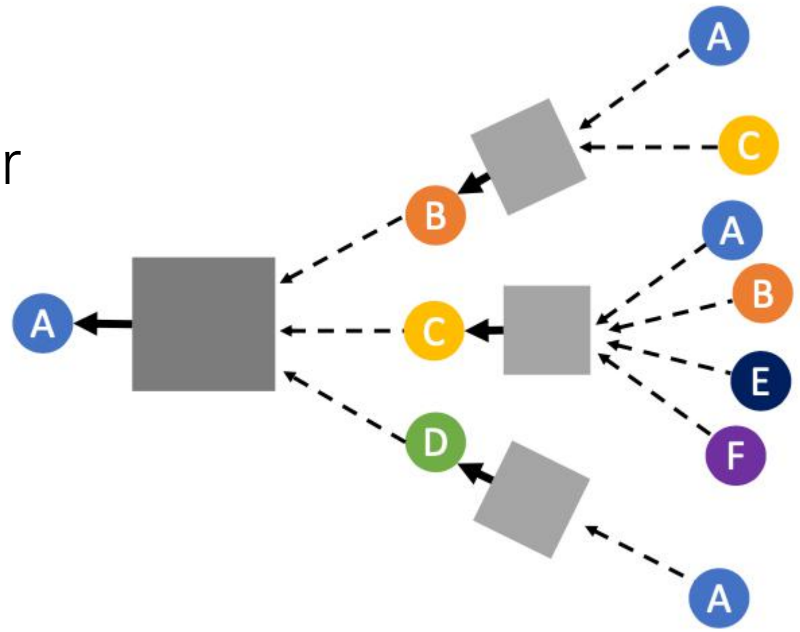
- Homophily assumption
  - Connected nodes are similar/related/informative
- How can we deal with heterophilous networks?[21,22]
  - Connected nodes have different class labels and dissimilar features



[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?"

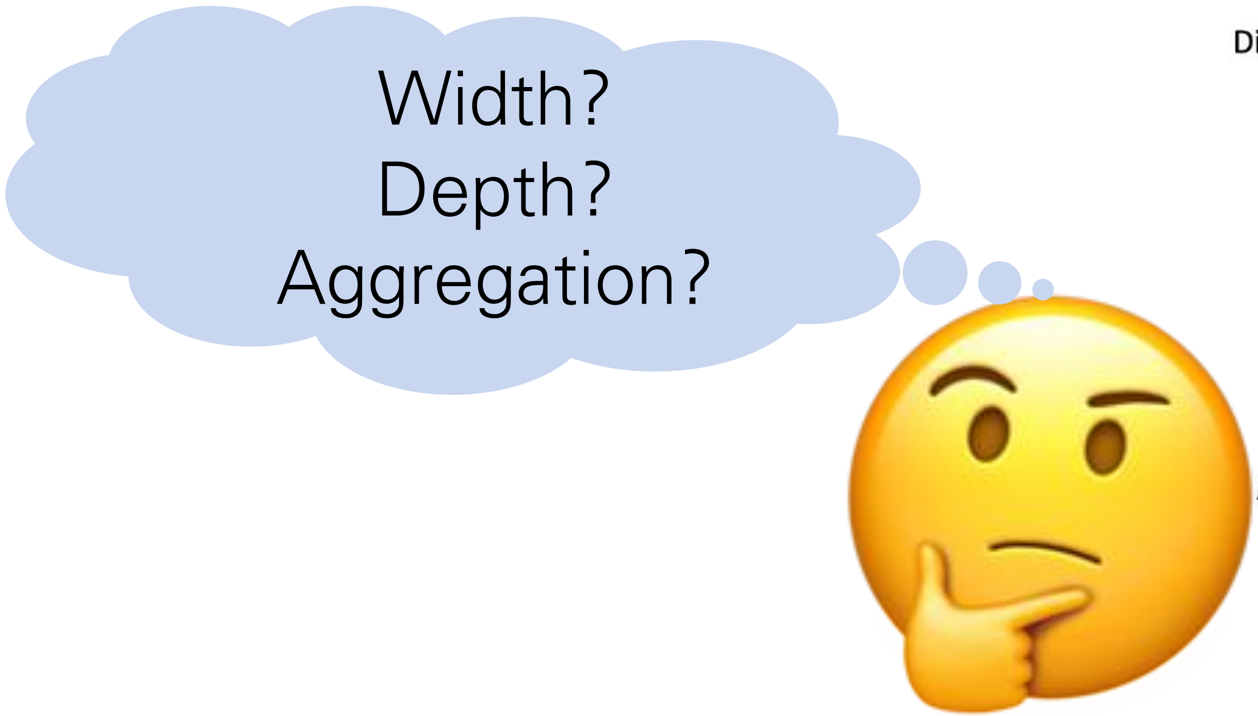
Graph Neural Network Architectures

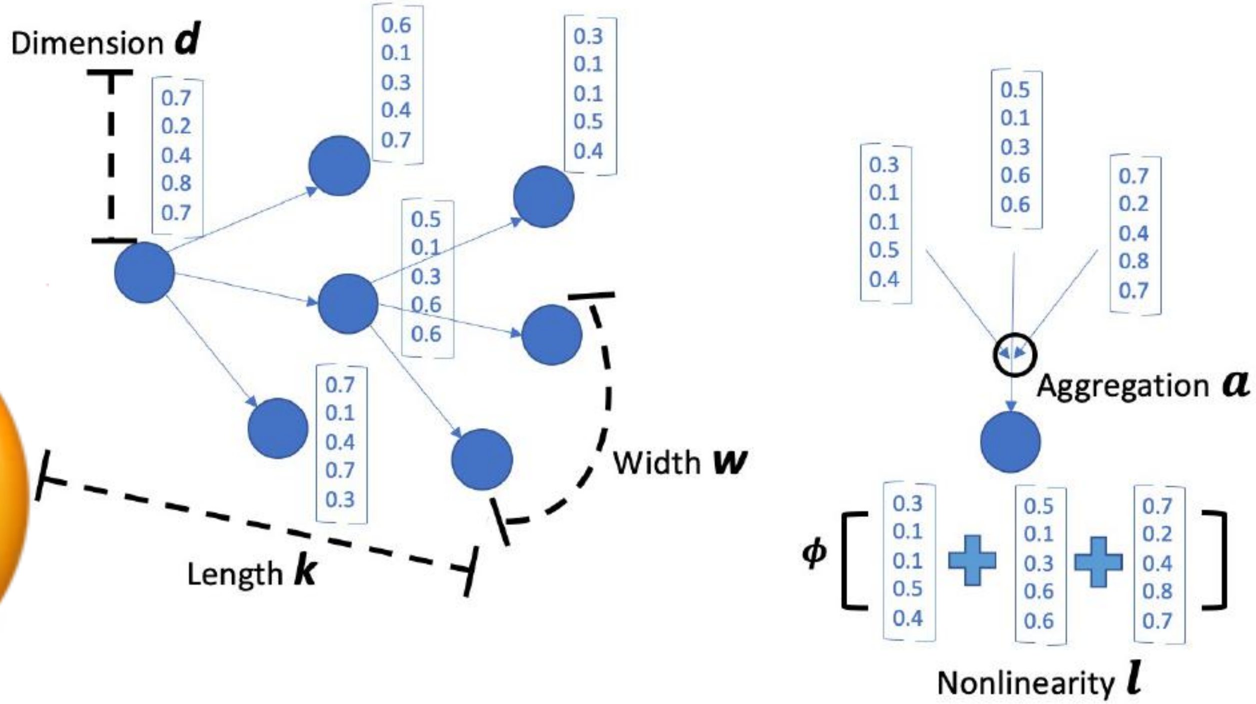
- Width
  - Which neighbors should we aggregate messages from?
- Depth
  - How many hops should we check?
- Aggregation
  - How should we aggregate messages from neighbor



Neural Architecture Search for GNNs

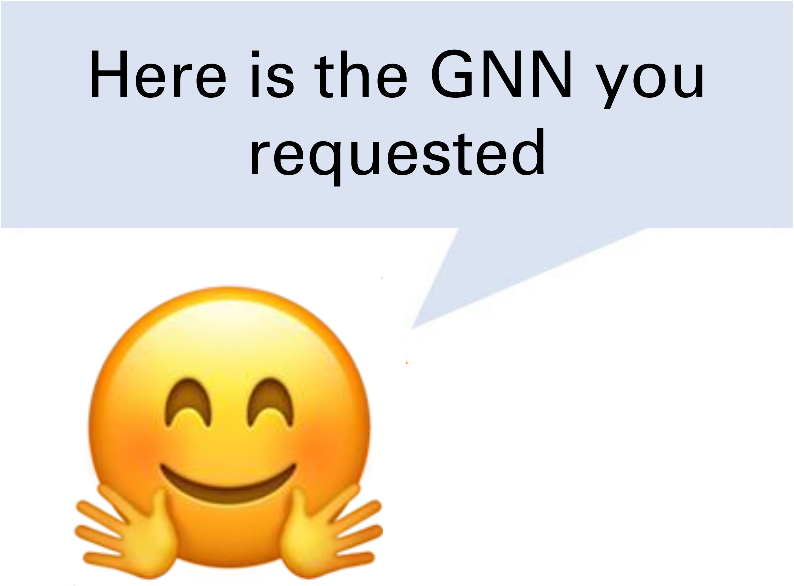
- Which width, depth, and aggregation strategy are proper for a given graph and task?





Neural Architecture Search for GNNs

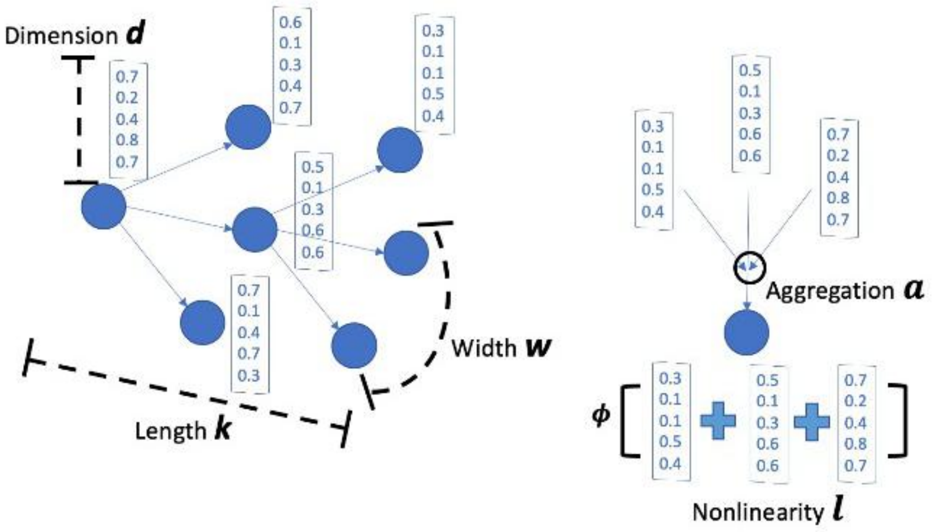
- Finding proper width, depth, and aggregation strategy for a given graph and task **automatically**[1,2,3]

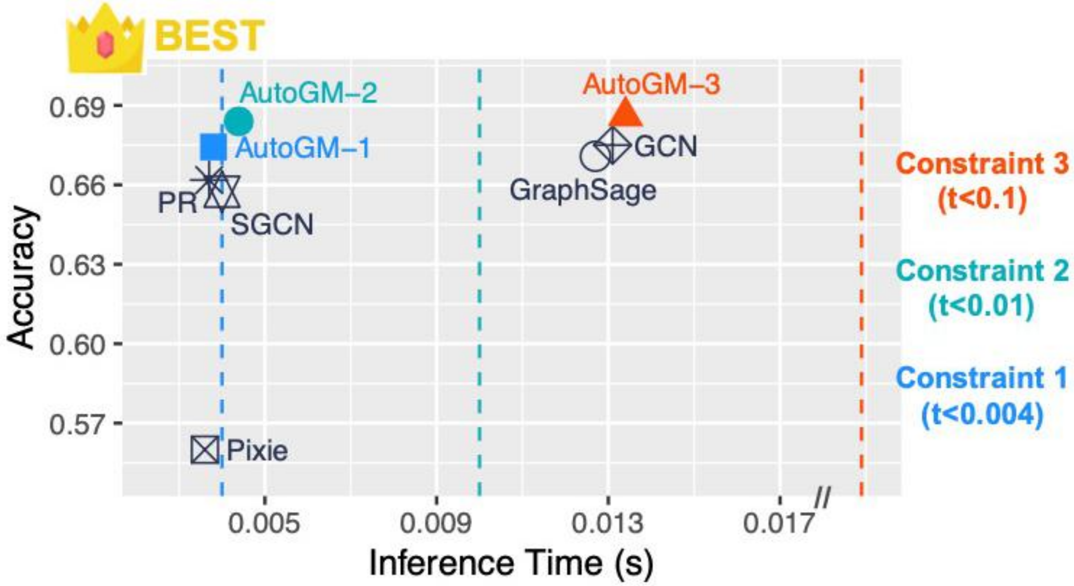


[23] Minji Yoon., et al. "Autonomous Graph Mining Algorithm Search with Best Speed/Accuracy Trade-off" [24] Kaixiong Zhou, et al. "Auto-GNN: Neural Architecture Search of Graph Neural Networks"
[25] Yang Gao, et al. "GraphNAS: Graph Neural Architecture Search with Reinforcement Learning"

Neural Architecture Search for GNNs

- AutoGM[23]





Step 1: define a hyperparameter space

Step 2: explore the space efficiently

[23] Minji Yoon., et al. "Autonomous Graph Mining Algorithm Search with Best Speed/Accuracy Trade-off"

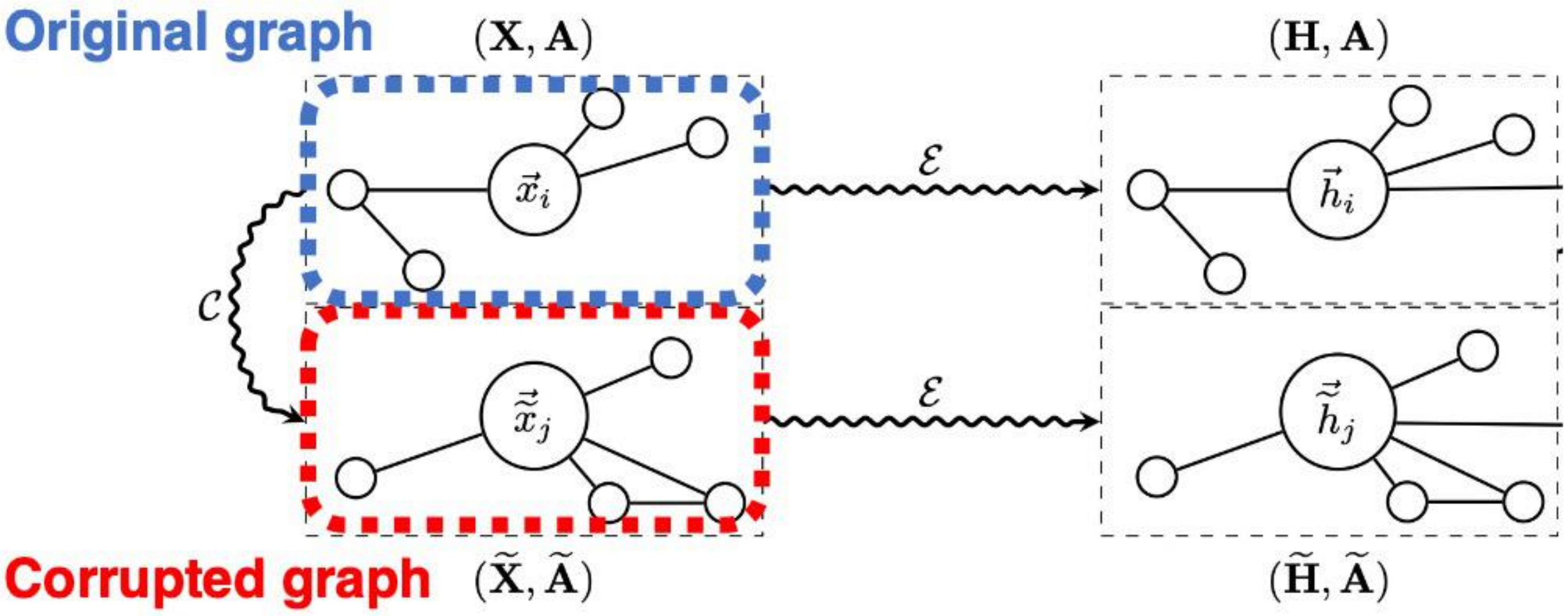
How to train GNNs?

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

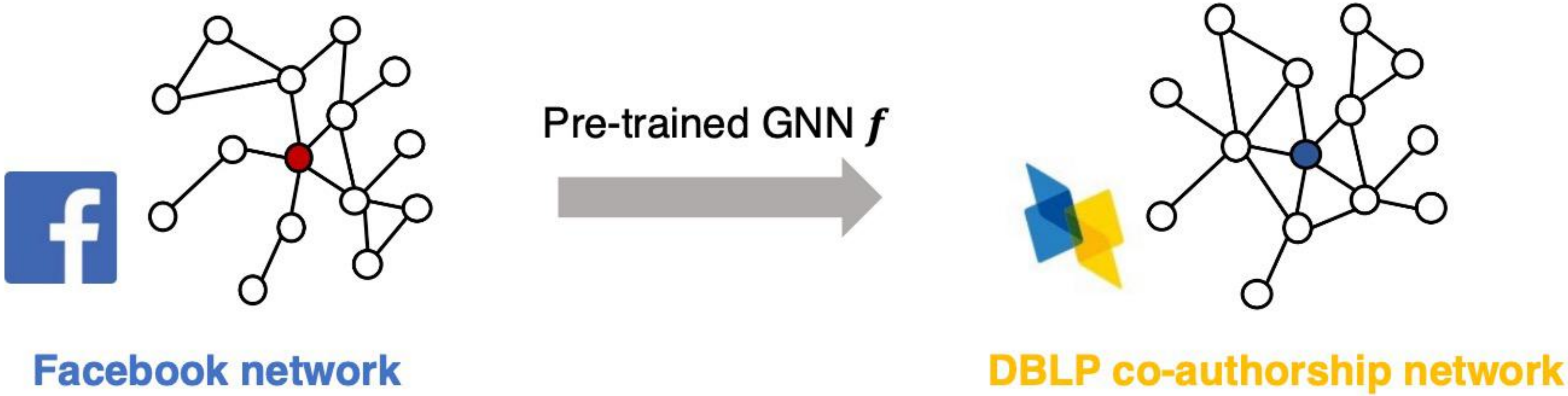
- Unsupervised learning[26]
  - Contrastive learning



[26] Petar Veličković., et al. "DEEP GRAPH INFOMAX"

How to train GNNs

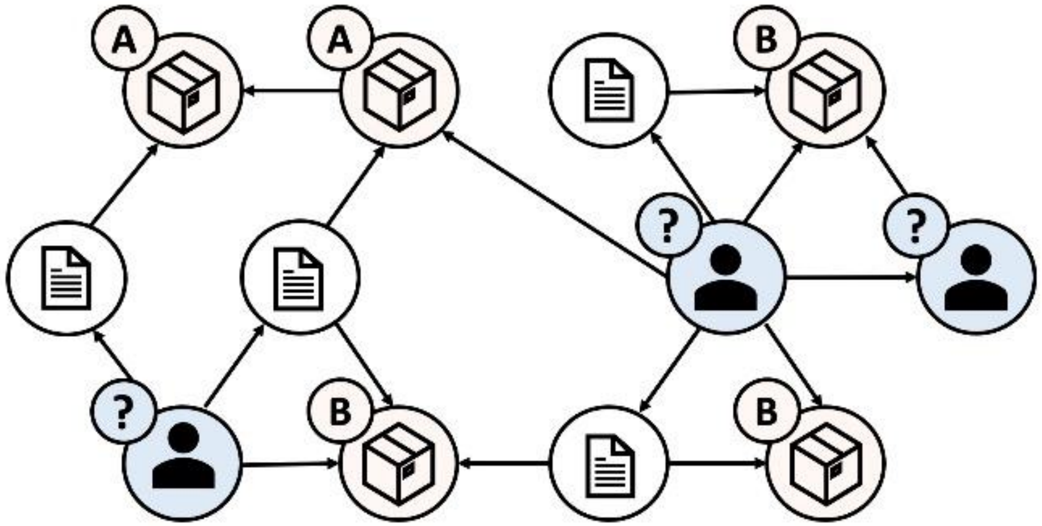
- Transfer learning
  - Transfer a pre-trained GNN model between graphs[27]



[27] Jiezhong Qiu, et al. "GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training"

How to train GNNs

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

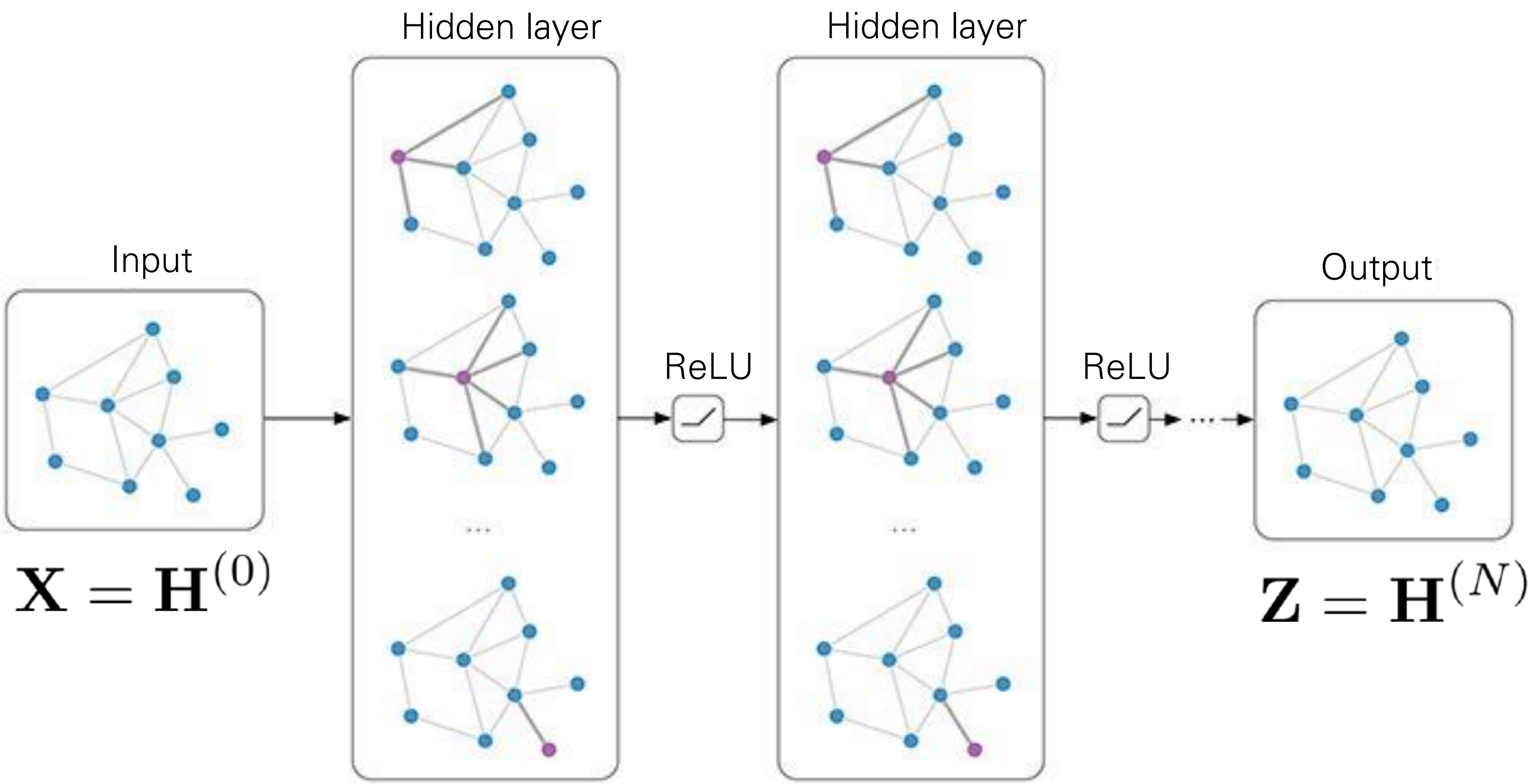


[28] Minji Yoon, et al. "Zero-shot Domain Adaptation of Heterogeneous Graphs via Knowledge Transfer Networks "

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 $\hat{\mathbf{A}}$



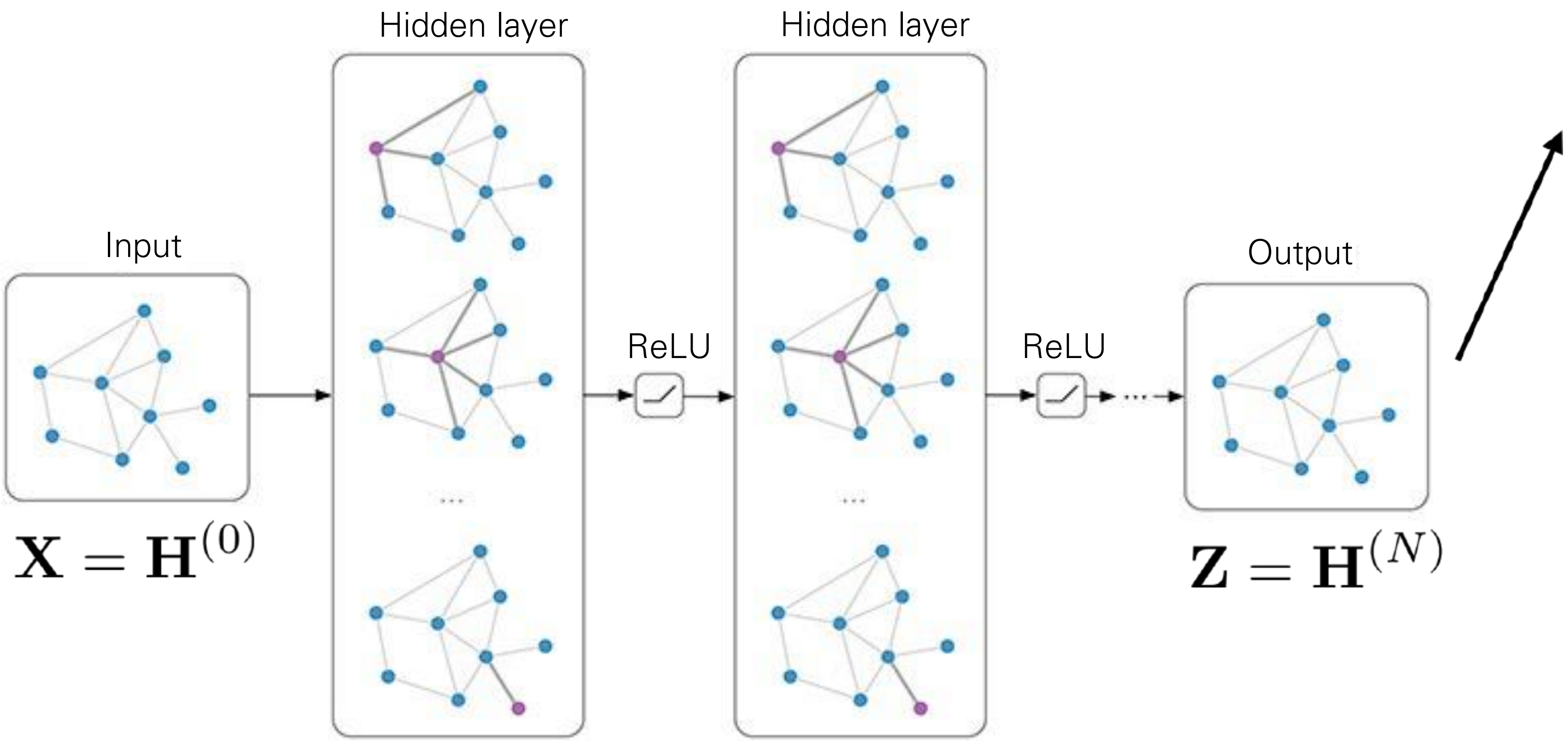
$\mathbf{X} = \mathbf{H}^{(0)}$

$\mathbf{Z} = \mathbf{H}^{(N)}$

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

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

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



$\mathbf{X} = \mathbf{H}^{(0)}$

$\mathbf{Z} = \mathbf{H}^{(N)}$

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

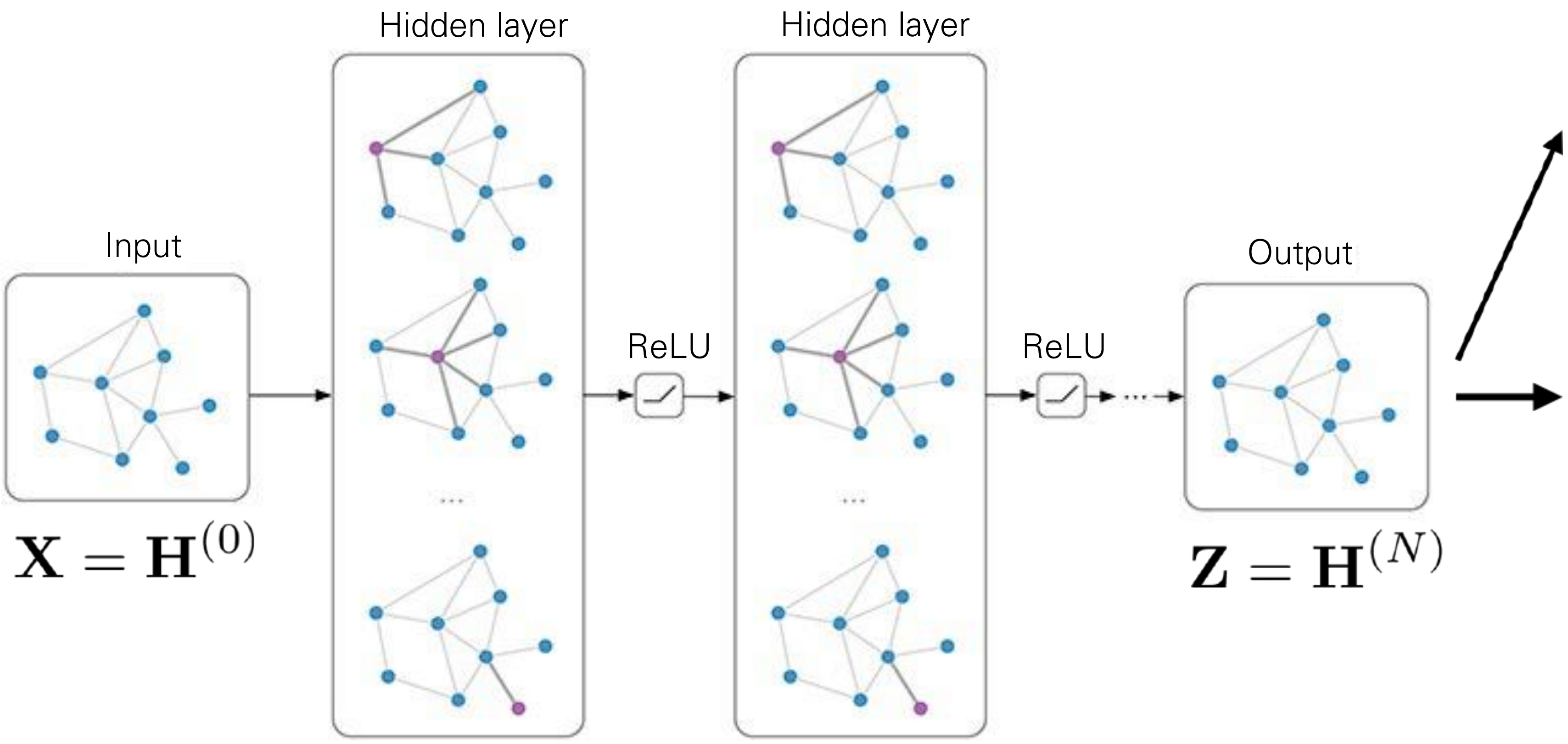
Node classification:

$$\mathrm{softmax}\left(\mathbf{z_n}\right)$$

e.g. Kipf & Welling (ICLR 2017)

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

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



$\mathbf{X} = \mathbf{H}^{(0)}$

$\mathbf{Z} = \mathbf{H}^{(N)}$

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

Node classification:

$$\mathrm{softmax}\left(\mathbf{z_n}\right)$$

e.g. Kipf & Welling (ICLR 2017)

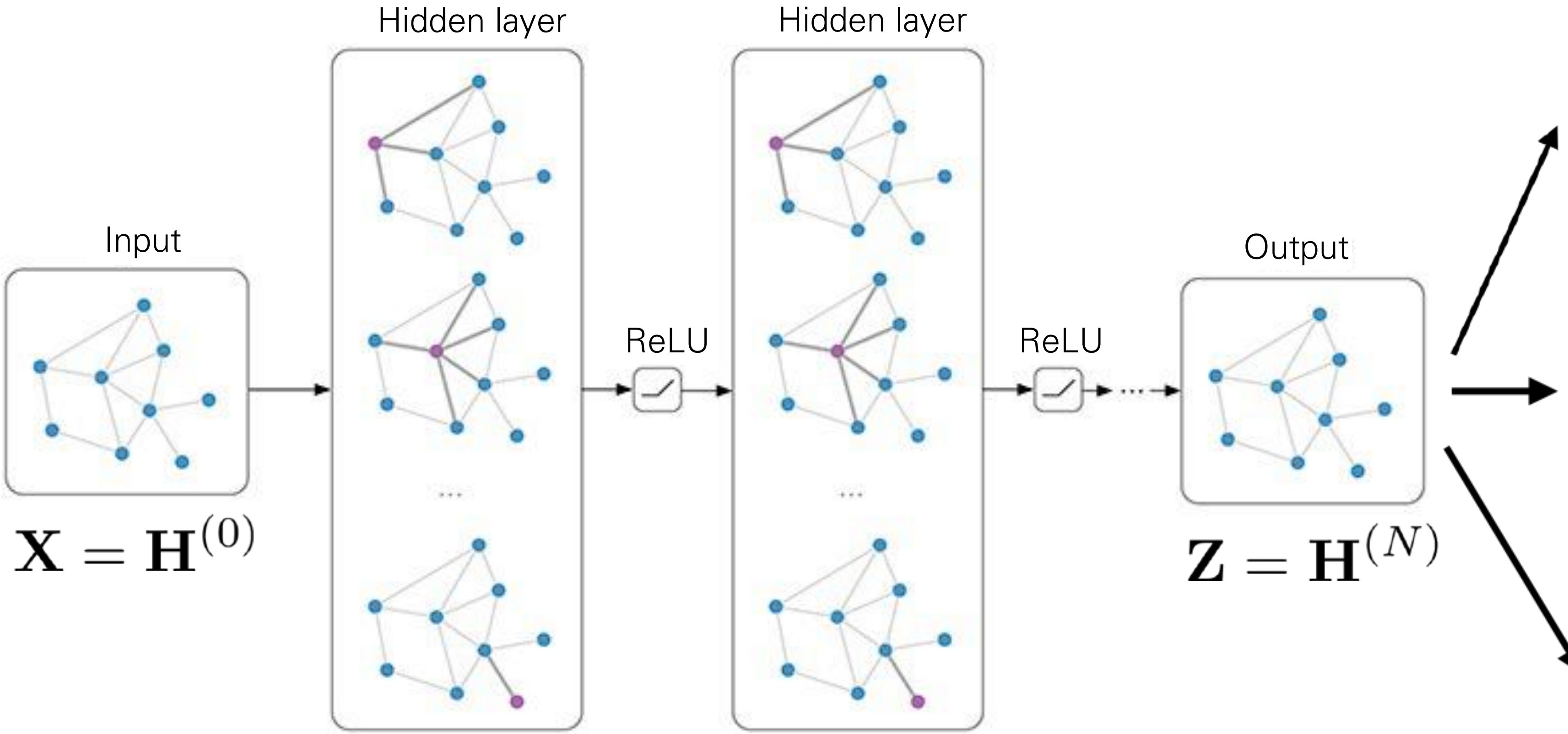
Graph classification:

$$\mathrm{softmax}\left(\sum_n \mathbf{z_n}\right)$$

e.g. Duvenaud et al. (NIPS 2015)

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

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



$\mathbf{X} = \mathbf{H}^{(0)}$

$\mathbf{Z} = \mathbf{H}^{(N)}$

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

Node classification:

$$\mathrm{softmax}\left(\mathbf{z_n}\right)$$

e.g. Kipf & Welling (ICLR 2017)

Graph classification:

$$\mathrm{softmax}\left(\sum_n \mathbf{z_n}\right)$$

e.g. Duvenaud et al. (NIPS 2015)

Link prediction:

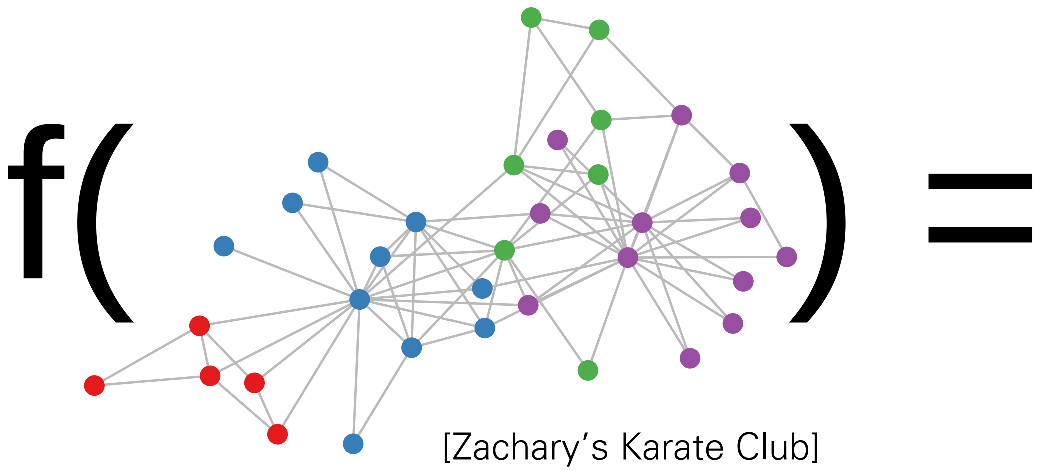
$$p\left(A_{ij}\right) = \sigma\left(\mathbf{z_i^T}\mathbf{z_j}\right)$$

Kipf & Welling (NIPS BDL 2016)
"Graph Auto-Encoders"

What do learned representations look like?

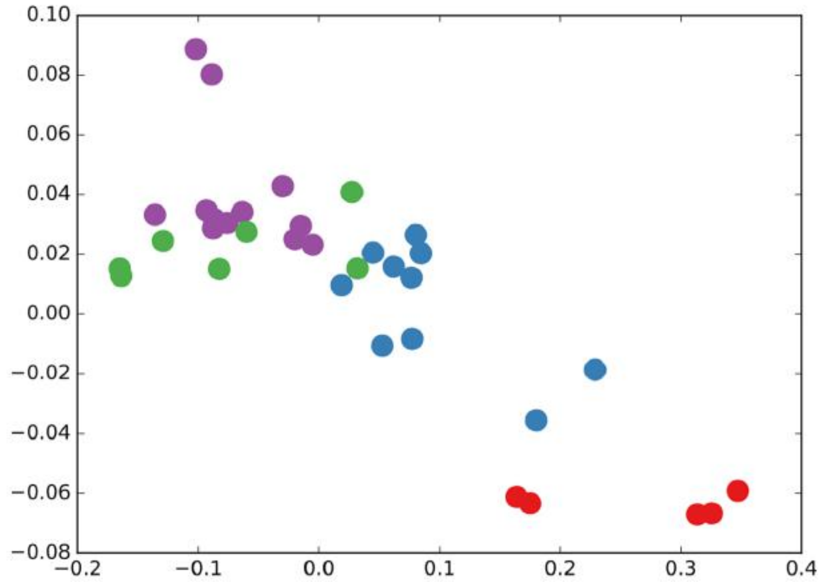
Forward pass through **untrained** 3-layer GCN model

Parameters initialized randomly



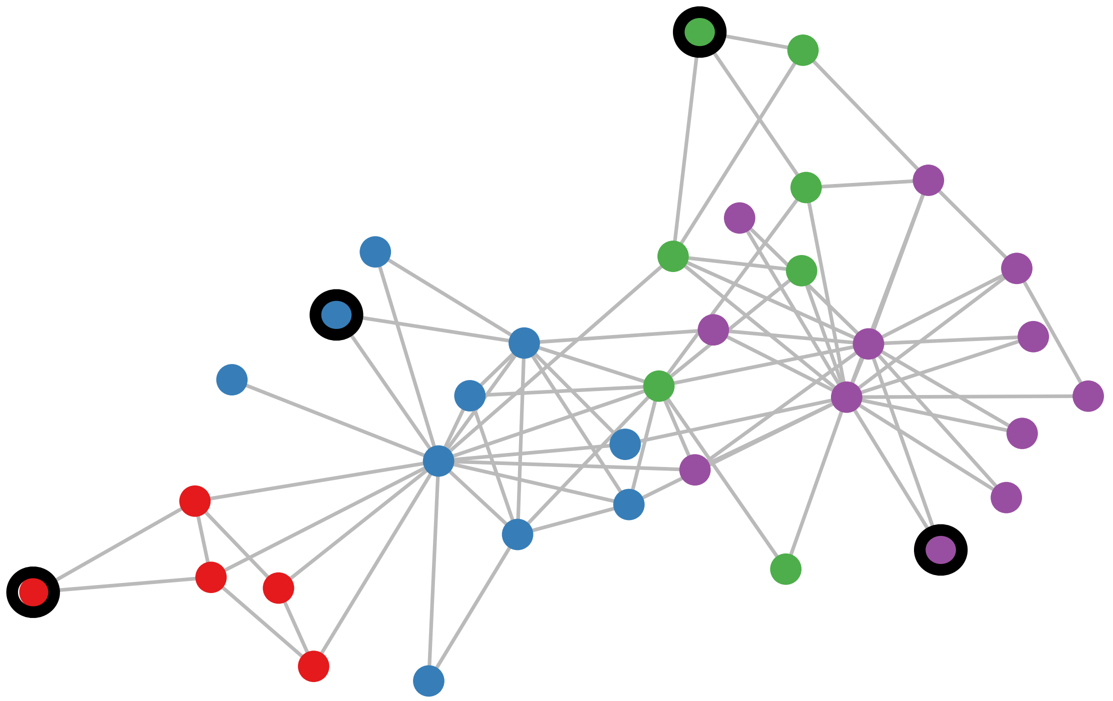
[Zachary's Karate Club]

2-dim output per node



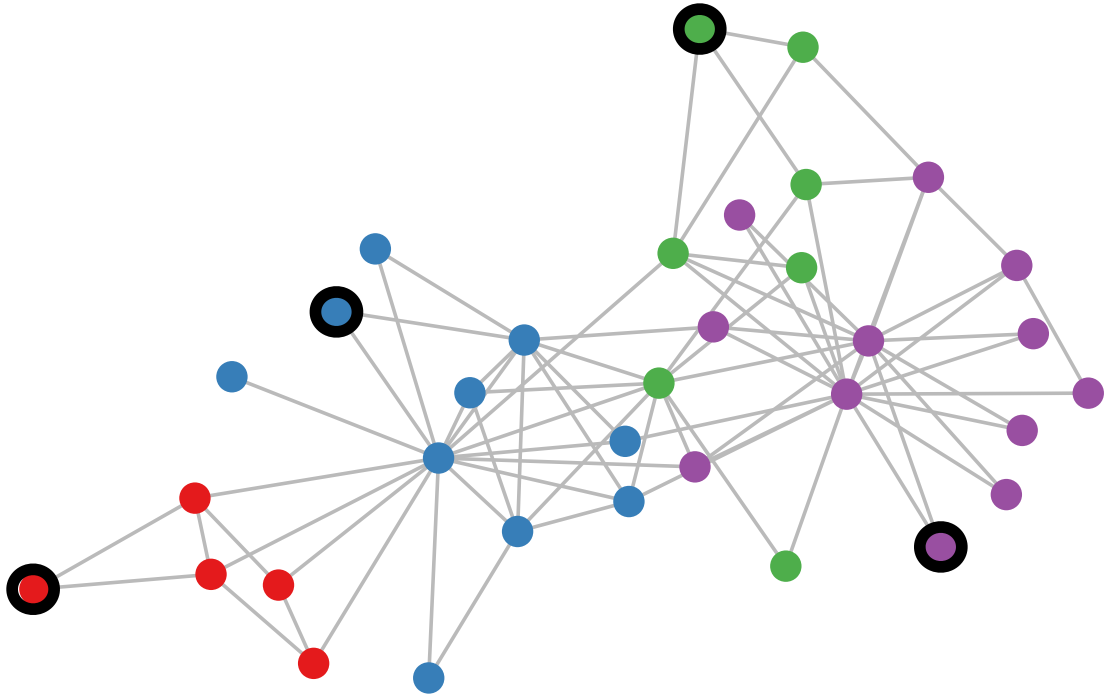
Semi-supervised classification on graphs

- **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:

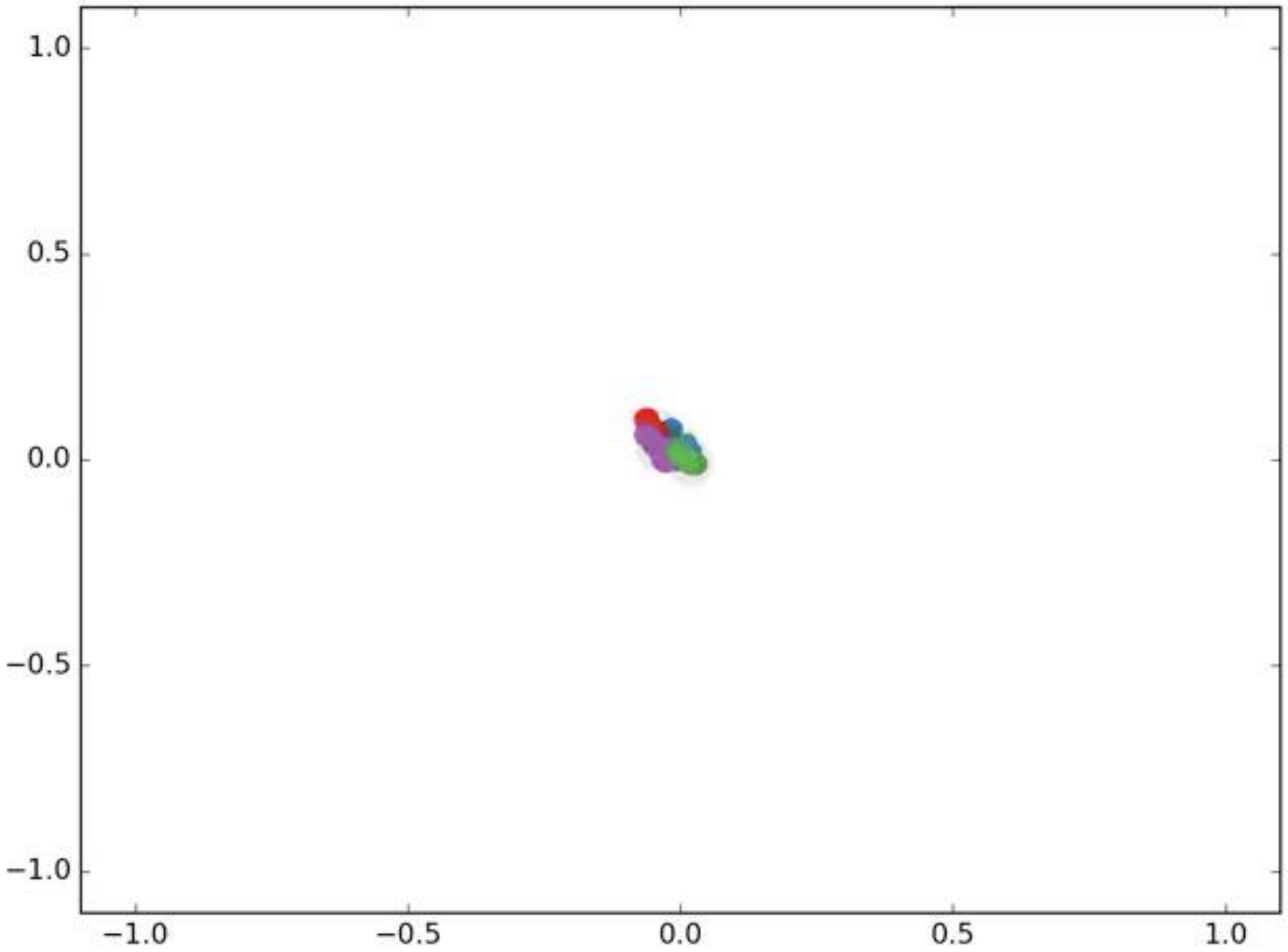
$$\mathcal{L} = -\sum_{l \in \mathcal{Y}_L} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf}$$

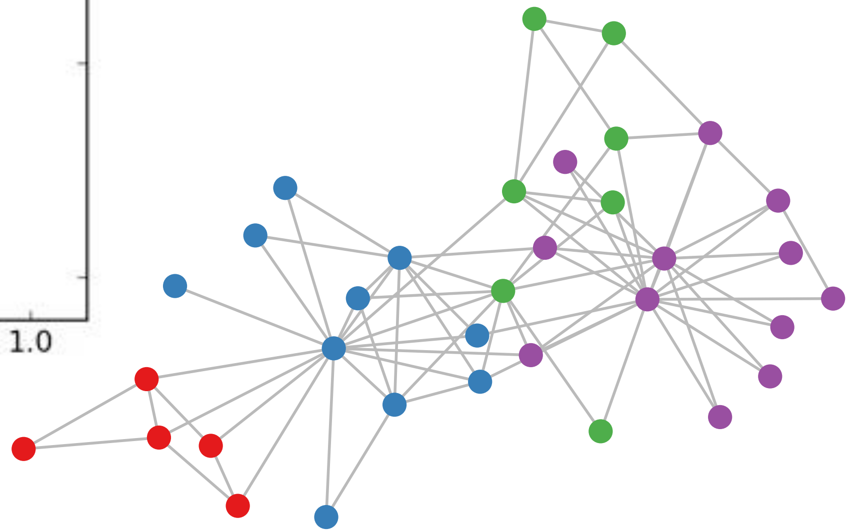
$\mathcal{Y}_L$ set of labeled node indices

$\mathbf{Y}$ label matrix

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

Toy example (semi-supervised learning)





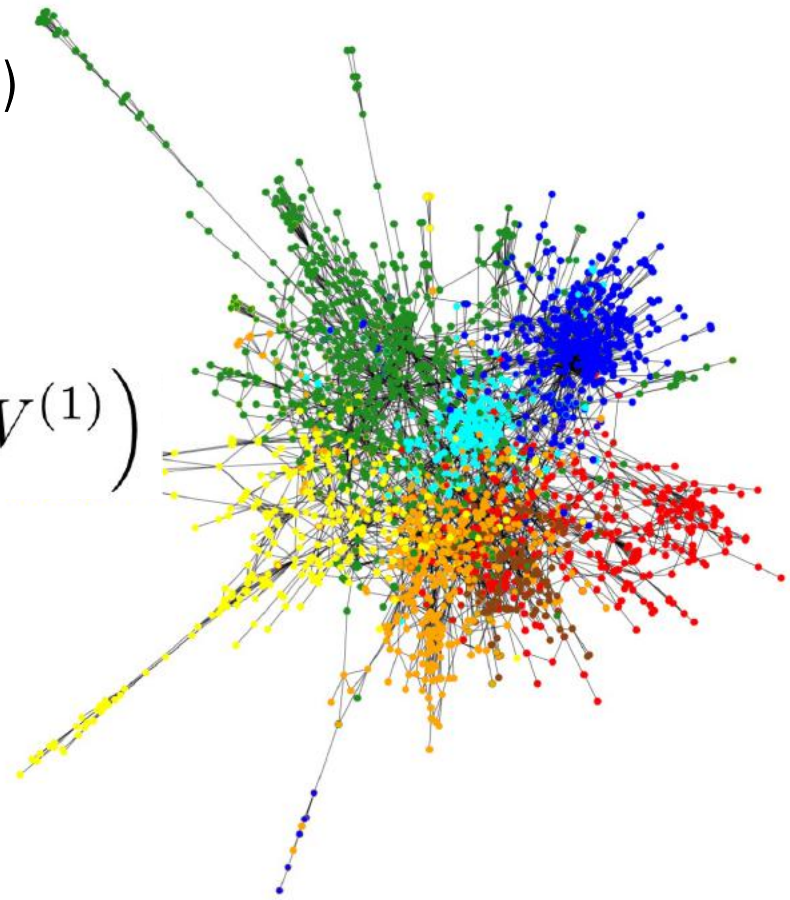
Application: Classification on citation networks

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

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



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

Kipf & Welling, Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017

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, ...)

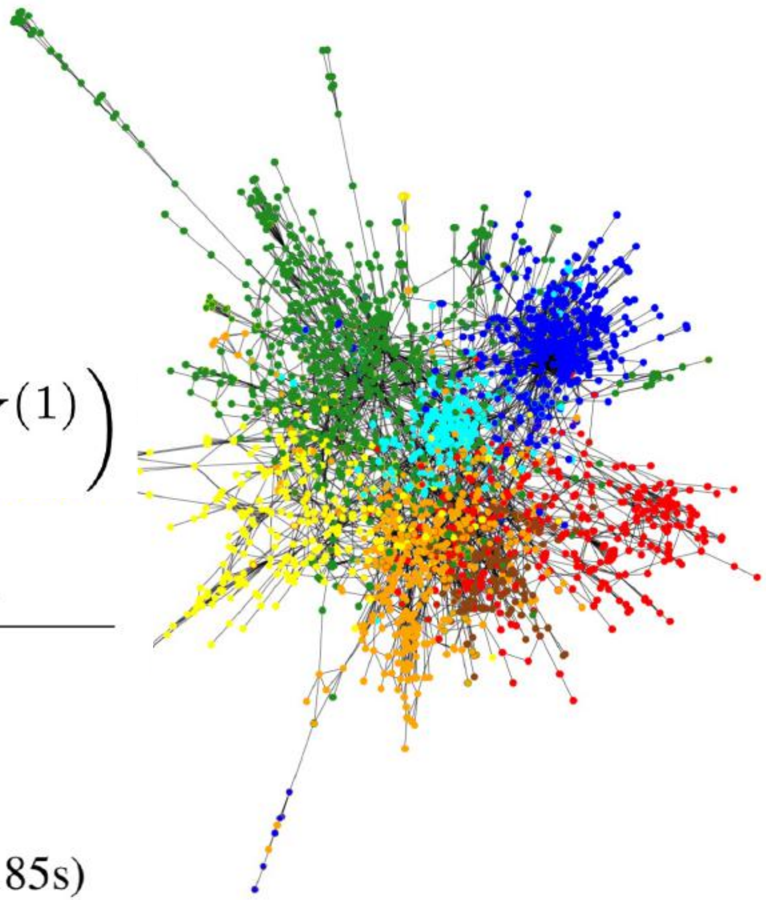
Model: 2-layer GCN

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

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
LP [27]	45.3	68.0	63.0	26.5
DeepWalk [18]	43.2	67.2	65.3	58.1
Planetoid* [25]	64.7 (26s)	75.7 (13s)	77.2 (25s)	61.9 (185s)
GCN (this paper)	**70.3** (7s)	**81.5** (4s)	**79.0** (38s)	**66.0** (48s)
GCN (rand. splits)	67.9 ± 0.5	80.1 ± 0.5	78.9 ± 0.7	58.4 ± 1.7

no input features



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

Kipf & Welling, Semi-Supervised Classification with Graph Convolutional Networks, ICLR 2017

Still many open problems..

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

Next Lecture:
Pretraining Language Models