On the Bottleneck of Graph Neural Networks and its Practical Implications

(ICLR ‘2021)

Uri Alon
Language Technologies Institute
Carnegie Mellon University

Eran Yahav
Technion

On the Bottleneck of Graph Neural Networks and its Practical Implications

(ICLR '2021)

Main Contribution: GNNs suffer from a **bottleneck** that causes **over-squashing** when trying to capture long-range interactions.

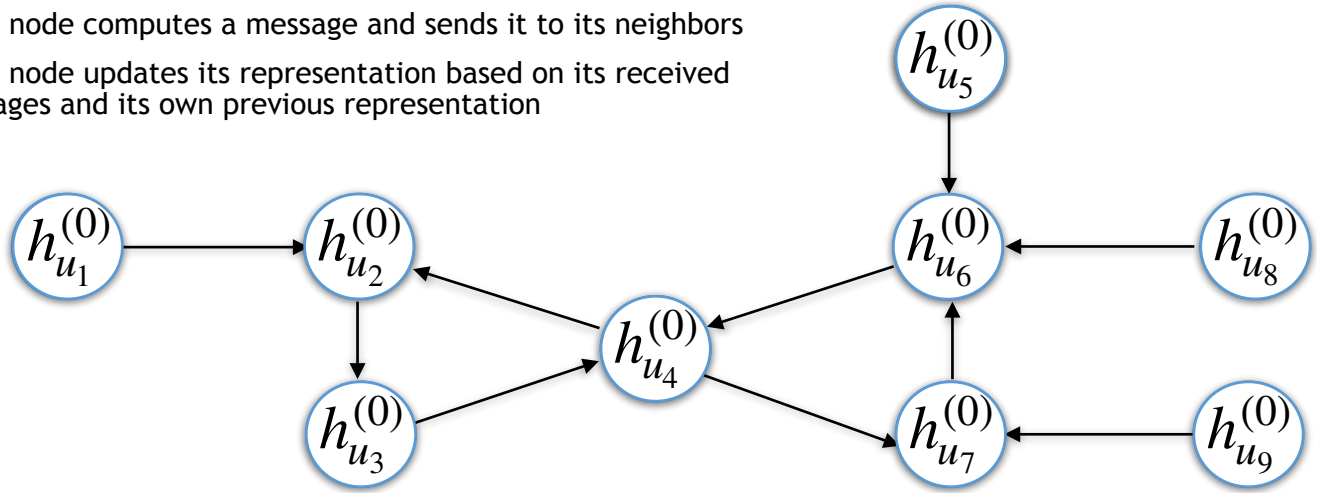
Uri Alon
Language Technologies Institute
Carnegie Mellon University

Eran Yahav
Technion

A GNN as a Message Passing Network

[Gilmer, ICML'2017]

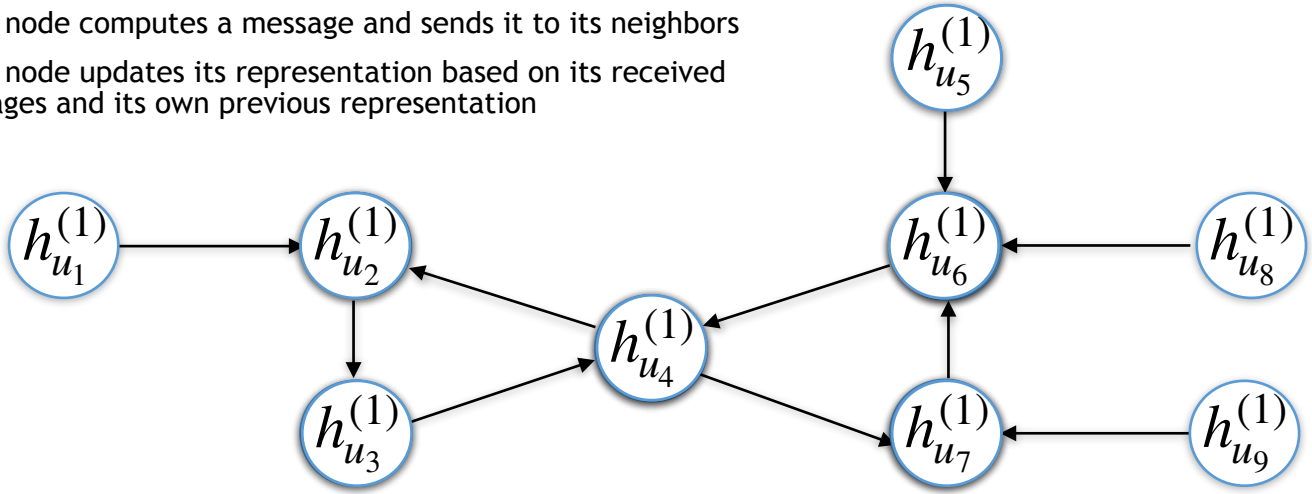
- Initial representations are embeddings or features
- At every message passing step (=layer):
  - Every node computes a message and sends it to its neighbors
  - Every node updates its representation based on its received messages and its own previous representation



A GNN as a Message Passing Network

[Gilmer, ICML'2017]

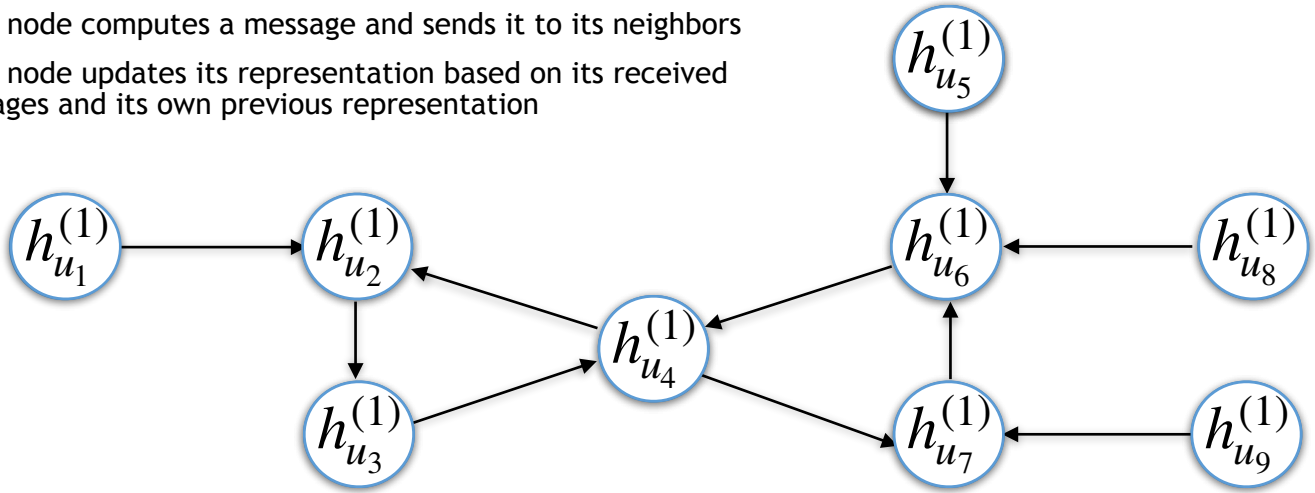
- Initial representations are embeddings or features
- At every message passing step (=layer):
  - Every node computes a message and sends it to its neighbors
  - Every node updates its representation based on its received messages and its own previous representation



A GNN as a Message Passing Network

[Gilmer, ICML'2017]

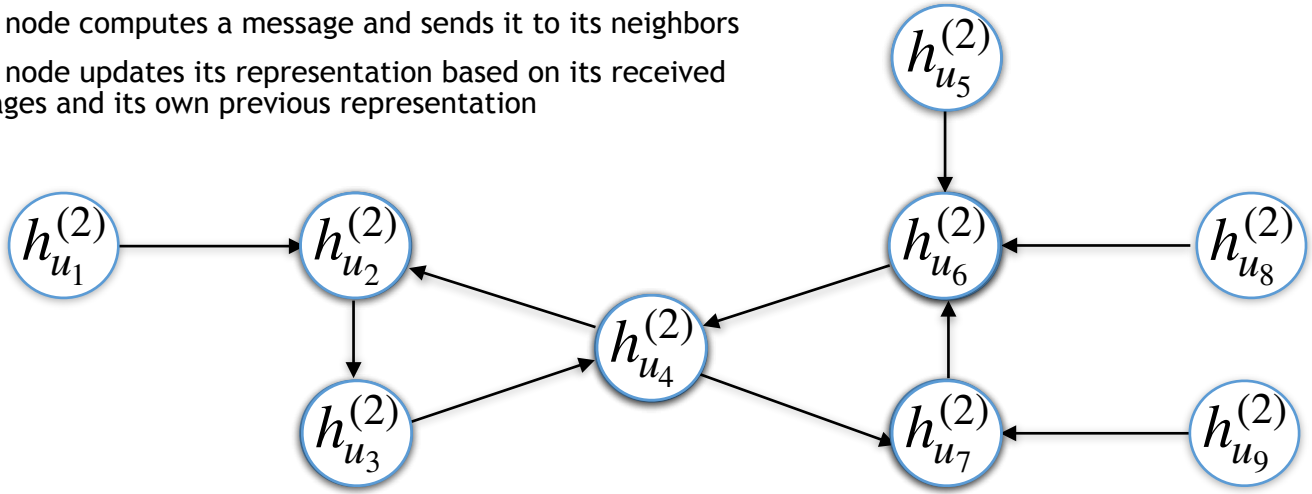
- Initial representations are embeddings or features
- At every message passing step (=layer):
  - Every node computes a message and sends it to its neighbors
  - Every node updates its representation based on its received messages and its own previous representation



A GNN as a Message Passing Network

[Gilmer, ICML'2017]

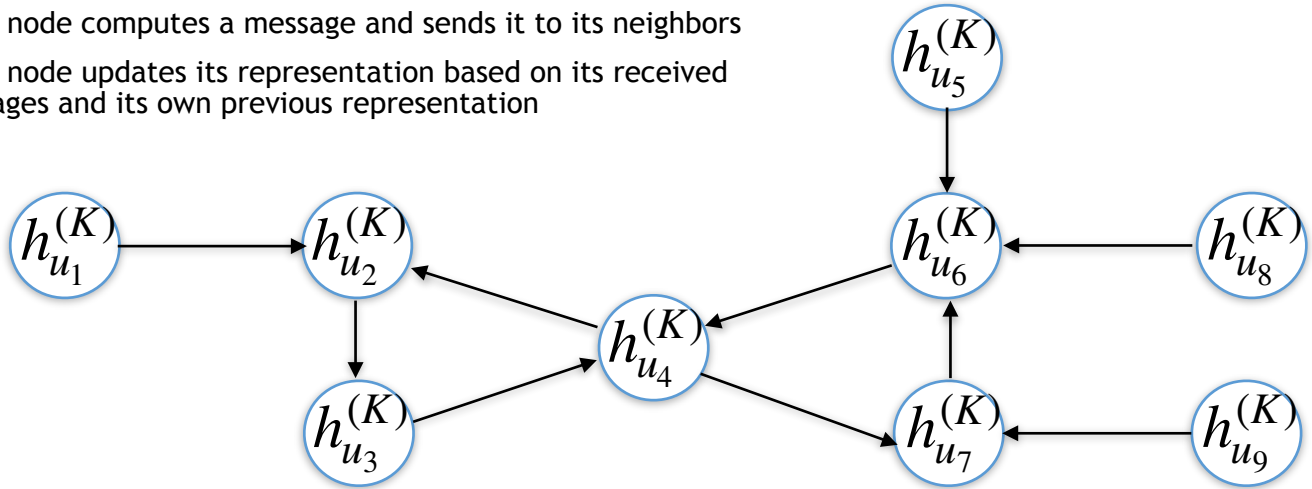
- Initial representations are embeddings or features
- At every message passing step (=layer):
  - Every node computes a message and sends it to its neighbors
  - Every node updates its representation based on its received messages and its own previous representation



A GNN as a Message Passing Network

[Gilmer, ICML'2017]

- Initial representations are embeddings or features
- At every message passing step (=layer):
  - Every node computes a message and sends it to its neighbors
  - Every node updates its representation based on its received messages and its own previous representation



- Given $\{h_u^{(K)} \mid u \in V\}$:
  - Node classification, graph classification, link prediction, sequence generation

GNN Types

- General: $\mathbf{h}_v^{(k)} = f_k\left(\mathbf{h}_v^{(k-1)}, \{\mathbf{h}_u^{(k-1)} \mid u \in \mathcal{N}_v\}\right) \qquad \mathcal{N}_v = \{u \in V \mid (u, v) \in E\}$

GNN Types

- General: $\mathbf{h}_v^{(k)} = f_k\left(\mathbf{h}_v^{(k-1)}, \{\mathbf{h}_u^{(k-1)} \mid u \in \mathcal{N}_v\}\right) \qquad \mathcal{N}_v = \{u \in V \mid (u, v) \in E\}$

GNN Types

- General:

$$\mathbf{h}_v^{(k)} = f_k\left(\mathbf{h}_v^{(k-1)}, \{\mathbf{h}_u^{(k-1)} \mid u \in \mathcal{N}_v\}\right) \qquad \mathcal{N}_v = \{u \in V \mid (u, v) \in E\}$$

GNN Types

- General:

$$\mathbf{h}_v^{(k)} = f_k\left(\mathbf{h}_v^{(k-1)}, \{\mathbf{h}_u^{(k-1)} \mid u \in \mathcal{N}_v\}\right) \qquad \mathcal{N}_v = \{u \in V \mid (u, v) \in E\}$$

GNN Types

- General:

$$\mathbf{h}_v^{(k)} = f_k\left(\mathbf{h}_v^{(k-1)}, \{\mathbf{h}_u^{(k-1)} \mid u \in \mathcal{N}_v\}\right) \qquad \mathcal{N}_v = \{u \in V \mid (u, v) \in E\}$$

GNN Types

- General:

$$\mathbf{h}_v^{(k)} = f_k\left(\mathbf{h}_v^{(k-1)}, \{\mathbf{h}_u^{(k-1)} \mid u \in \mathcal{N}_v\}\right) \qquad \mathcal{N}_v = \{u \in V \mid (u, v) \in E\}$$

- Graph convolutional network (GCN):
  (Kipf & Welling, ICLR'2017)

$$\mathbf{h}_v^{(k)} = ReLU\left(W^{(k)}\left(\frac{1}{c_v}\mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}_v} \frac{1}{c_{v,u}}\mathbf{h}_u^{(k-1)}\right)\right)$$

GNN Types

- General:

$$\mathbf{h}_v^{(k)} = f_k\left(\mathbf{h}_v^{(k-1)}, \{\mathbf{h}_u^{(k-1)} \mid u \in \mathcal{N}_v\}\right) \qquad \mathcal{N}_v = \{u \in V \mid (u,v) \in E\}$$

- Graph convolutional network (GCN):
  (Kipf & Welling, ICLR'2017)

$$\mathbf{h}_v^{(k)} = ReLU\left(W^{(k)}\left(\frac{1}{c_v}\mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}_v} \frac{1}{c_{v,u}}\mathbf{h}_u^{(k-1)}\right)\right)$$

GNN Types

- General:

$$\mathbf{h}_v^{(k)} = f_k\left(\mathbf{h}_v^{(k-1)}, \{\mathbf{h}_u^{(k-1)} \mid u \in \mathcal{N}_v\}\right) \qquad \mathcal{N}_v = \{u \in V \mid (u,v) \in E\}$$

- Graph convolutional network (GCN):
  (Kipf & Welling, ICLR'2017)

$$\mathbf{h}_v^{(k)} = ReLU\left(W^{(k)}\left(\frac{1}{c_v}\mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}_v} \frac{1}{c_{v,u}}\mathbf{h}_u^{(k-1)}\right)\right)$$

GNN Types

- General:

$$\mathbf{h}_v^{(k)} = f_k\left(\mathbf{h}_v^{(k-1)}, \{\mathbf{h}_u^{(k-1)} \mid u \in \mathcal{N}_v\}\right) \qquad \mathcal{N}_v = \{u \in V \mid (u,v) \in E\}$$

- Graph convolutional network (GCN):
  (Kipf & Welling, ICLR'2017)

$$\mathbf{h}_v^{(k)} = ReLU\left(W^{(k)}\left(\frac{1}{c_v}\mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}_v} \frac{1}{c_{v,u}}\mathbf{h}_u^{(k-1)}\right)\right)$$

GNN Types

- General:

$$\mathbf{h}_v^{(k)} = f_k\left(\mathbf{h}_v^{(k-1)}, \{\mathbf{h}_u^{(k-1)} \mid u \in \mathcal{N}_v\}\right) \qquad \mathcal{N}_v = \{u \in V \mid (u,v) \in E\}$$

- Graph convolutional network (GCN):
  (Kipf & Welling, ICLR'2017)

$$\mathbf{h}_v^{(k)} = ReLU\left(W^{(k)}\left(\frac{1}{c_v}\mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}_v} \frac{1}{c_{v,u}}\mathbf{h}_u^{(k-1)}\right)\right)$$

- Graph Attention Network (GAT):
  (Veličković et al., ICLR'2018)

$$\mathbf{h}_v^{(k)} = ReLU\left(MultiHeadAttention\left(\mathcal{N}_v \mid \mathbf{h}_v^{(k-1)}\right)\right)$$

GNN Types

- General:

$$\mathbf{h}_v^{(k)} = f_k\left(\mathbf{h}_v^{(k-1)}, \{\mathbf{h}_u^{(k-1)} \mid u \in \mathcal{N}_v\}\right) \qquad \mathcal{N}_v = \{u \in V \mid (u,v) \in E\}$$

- Graph convolutional network (GCN):
  (Kipf & Welling, ICLR'2017)

$$\mathbf{h}_v^{(k)} = ReLU\left(W^{(k)}\left(\frac{1}{c_v}\mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}_v} \frac{1}{c_{v,u}}\mathbf{h}_u^{(k-1)}\right)\right)$$

- Graph Attention Network (GAT):
  (Veličković et al., ICLR'2018)

$$\mathbf{h}_v^{(k)} = ReLU\left(MultiHeadAttention\left(\mathcal{N}_v \mid \mathbf{h}_v^{(k-1)}\right)\right)$$

- Gated Graph Neural Networks (GGNN):
  (Li et al., ICLR'2016)

$$\mathbf{h}_v^{(k)} = GRU\left(\mathbf{h}_v^{(k-1)}, W \cdot \sum_{u \in \mathcal{N}_v} \mathbf{h}_u^{(k-1)}\right)$$

GNN Types

- General:

$$\mathbf{h}_v^{(k)} = f_k\left(\mathbf{h}_v^{(k-1)}, \{\mathbf{h}_u^{(k-1)} \mid u \in \mathcal{N}_v\}\right) \qquad \mathcal{N}_v = \{u \in V \mid (u,v) \in E\}$$

- Graph convolutional network (GCN):
  (Kipf & Welling, ICLR'2017)

$$\mathbf{h}_v^{(k)} = ReLU\left(W^{(k)}\left(\frac{1}{c_v}\mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}_v} \frac{1}{c_{v,u}}\mathbf{h}_u^{(k-1)}\right)\right)$$

- Graph Attention Network (GAT):
  (Veličković et al., ICLR'2018)

$$\mathbf{h}_v^{(k)} = ReLU\left(MultiHeadAttention\left(\mathcal{N}_v \mid \mathbf{h}_v^{(k-1)}\right)\right)$$

- Gated Graph Neural Networks (GGNN):
  (Li et al., ICLR'2016)

$$\mathbf{h}_v^{(k)} = GRU\left(\mathbf{h}_v^{(k-1)}, W \cdot \sum_{u \in \mathcal{N}_v} \mathbf{h}_u^{(k-1)}\right)$$

GNN Types

- General:

$$\mathbf{h}_v^{(k)} = f_k\left(\mathbf{h}_v^{(k-1)}, \{\mathbf{h}_u^{(k-1)} \mid u \in \mathcal{N}_v\}\right) \qquad \mathcal{N}_v = \{u \in V \mid (u,v) \in E\}$$

- Graph convolutional network (GCN):
  (Kipf & Welling, ICLR'2017)

$$\mathbf{h}_v^{(k)} = ReLU\left(W^{(k)}\left(\frac{1}{c_v}\mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}_v} \frac{1}{c_{v,u}}\mathbf{h}_u^{(k-1)}\right)\right)$$

- Graph Attention Network (GAT):
  (Veličković et al., ICLR'2018)

$$\mathbf{h}_v^{(k)} = ReLU\left(MultiHeadAttention\left(\mathcal{N}_v \mid \mathbf{h}_v^{(k-1)}\right)\right)$$

- Gated Graph Neural Networks (GGNN):
  (Li et al., ICLR'2016)

$$\mathbf{h}_v^{(k)} = GRU\left(\mathbf{h}_v^{(k-1)}, W \cdot \sum_{u \in \mathcal{N}_v} \mathbf{h}_u^{(k-1)}\right)$$

GNN Types

- General:

$$\mathbf{h}_v^{(k)} = f_k\left(\mathbf{h}_v^{(k-1)}, \{\mathbf{h}_u^{(k-1)} \mid u \in \mathcal{N}_v\}\right) \qquad \mathcal{N}_v = \{u \in V \mid (u,v) \in E\}$$

- Graph convolutional network (GCN):
  (Kipf & Welling, ICLR'2017)

$$\mathbf{h}_v^{(k)} = ReLU\left(W^{(k)}\left(\frac{1}{c_v}\mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}_v} \frac{1}{c_{v,u}}\mathbf{h}_u^{(k-1)}\right)\right)$$

- Graph Attention Network (GAT):
  (Veličković et al., ICLR'2018)

$$\mathbf{h}_v^{(k)} = ReLU\left(MultiHeadAttention\left(\mathcal{N}_v \mid \mathbf{h}_v^{(k-1)}\right)\right)$$

- Gated Graph Neural Networks (GGNN):
  (Li et al., ICLR'2016)

$$\mathbf{h}_v^{(k)} = GRU\left(\mathbf{h}_v^{(k-1)}, W \cdot \sum_{u \in \mathcal{N}_v} \mathbf{h}_u^{(k-1)}\right)$$

- Graph Isomorphism Network (GIN):
  (Xu et al., ICLR'2019)

$$\mathbf{h}_v^{(k)} = MLP^{(k)}\left(\left(1 + \epsilon^{(k)}\right)\mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}_v} \mathbf{h}_u^{(k-1)}\right)$$

GNN Types

- General:

$$\mathbf{h}_v^{(k)} = f_k\left(\mathbf{h}_v^{(k-1)}, \{\mathbf{h}_u^{(k-1)} \mid u \in \mathcal{N}_v\}\right) \qquad \mathcal{N}_v = \{u \in V \mid (u,v) \in E\}$$

- Graph convolutional network (GCN):
  (Kipf & Welling, ICLR'2017)

$$\mathbf{h}_v^{(k)} = ReLU\left(W^{(k)}\left(\frac{1}{c_v}\mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}_v} \frac{1}{c_{v,u}}\mathbf{h}_u^{(k-1)}\right)\right)$$

- Graph Attention Network (GAT):
  (Veličković et al., ICLR'2018)

$$\mathbf{h}_v^{(k)} = ReLU\left(MultiHeadAttention\left(\mathcal{N}_v \mid \mathbf{h}_v^{(k-1)}\right)\right)$$

- Gated Graph Neural Networks (GGNN):
  (Li et al., ICLR'2016)

$$\mathbf{h}_v^{(k)} = GRU\left(\mathbf{h}_v^{(k-1)}, W \cdot \sum_{u \in \mathcal{N}_v} \mathbf{h}_u^{(k-1)}\right)$$

- Graph Isomorphism Network (GIN):
  (Xu et al., ICLR'2019)

$$\mathbf{h}_v^{(k)} = MLP^{(k)}\left(\left(1 + \epsilon^{(k)}\right)\mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}_v} \mathbf{h}_u^{(k-1)}\right)$$

GNN Types

- General:

$$\mathbf{h}_v^{(k)} = f_k\left(\mathbf{h}_v^{(k-1)}, \{\mathbf{h}_u^{(k-1)} \mid u \in \mathcal{N}_v\}\right) \qquad \mathcal{N}_v = \{u \in V \mid (u,v) \in E\}$$

- Graph convolutional network (GCN):
  (Kipf & Welling, ICLR'2017)

$$\mathbf{h}_v^{(k)} = ReLU\left(W^{(k)}\left(\frac{1}{c_v}\mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}_v} \frac{1}{c_{v,u}}\mathbf{h}_u^{(k-1)}\right)\right)$$

- Graph Attention Network (GAT):
  (Veličković et al., ICLR'2018)

$$\mathbf{h}_v^{(k)} = ReLU\left(MultiHeadAttention\left(\mathcal{N}_v \mid \mathbf{h}_v^{(k-1)}\right)\right)$$

- Gated Graph Neural Networks (GGNN):
  (Li et al., ICLR'2016)

$$\mathbf{h}_v^{(k)} = GRU\left(\mathbf{h}_v^{(k-1)}, W \cdot \sum_{u \in \mathcal{N}_v} \mathbf{h}_u^{(k-1)}\right)$$

- Graph Isomorphism Network (GIN):
  (Xu et al., ICLR'2019)

$$\mathbf{h}_v^{(k)} = MLP^{(k)}\left(\left(1 + \epsilon^{(k)}\right)\mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}_v} \mathbf{h}_u^{(k-1)}\right)$$

GNN Types

- General:

$$\mathbf{h}_v^{(k)} = f_k\left(\mathbf{h}_v^{(k-1)}, \{\mathbf{h}_u^{(k-1)} \mid u \in \mathcal{N}_v\}\right) \qquad \mathcal{N}_v = \{u \in V \mid (u,v) \in E\}$$

- Graph convolutional network (GCN):
  (Kipf & Welling, ICLR'2017)

$$\mathbf{h}_v^{(k)} = ReLU\left(W^{(k)}\left(\frac{1}{c_v}\mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}_v} \frac{1}{c_{v,u}}\mathbf{h}_u^{(k-1)}\right)\right)$$

- Graph Attention Network (GAT):
  (Veličković et al., ICLR'2018)

$$\mathbf{h}_v^{(k)} = ReLU\left(MultiHeadAttention\left(\mathcal{N}_v \mid \mathbf{h}_v^{(k-1)}\right)\right)$$

- Gated Graph Neural Networks (GGNN):
  (Li et al., ICLR'2016)

$$\mathbf{h}_v^{(k)} = GRU\left(\mathbf{h}_v^{(k-1)}, W \cdot \sum_{u \in \mathcal{N}_v} \mathbf{h}_u^{(k-1)}\right)$$

- Graph Isomorphism Network (GIN):
  (Xu et al., ICLR'2019)

$$\mathbf{h}_v^{(k)} = MLP^{(k)}\left(\left(1 + \epsilon^{(k)}\right)\mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}_v} \mathbf{h}_u^{(k-1)}\right)$$

GNN Types

- General:

$$\mathbf{h}_v^{(k)} = f_k\left(\mathbf{h}_v^{(k-1)}, \{\mathbf{h}_u^{(k-1)} \mid u \in \mathcal{N}_v\}\right) \qquad \mathcal{N}_v = \{u \in V \mid (u,v) \in E\}$$

- Graph convolutional network (GCN):
  (Kipf & Welling, ICLR'2017)

$$\mathbf{h}_v^{(k)} = ReLU\left(W^{(k)}\left(\frac{1}{c_v}\mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}_v} \frac{1}{c_{v,u}}\mathbf{h}_u^{(k-1)}\right)\right)$$

- Graph Attention Network (GAT):
  (Veličković et al., ICLR'2018)

$$\mathbf{h}_v^{(k)} = ReLU\left(MultiHeadAttention\left(\mathcal{N}_v \mid \mathbf{h}_v^{(k-1)}\right)\right)$$

- Gated Graph Neural Networks (GGNN):
  (Li et al., ICLR'2016)

$$\mathbf{h}_v^{(k)} = GRU\left(\mathbf{h}_v^{(k-1)}, W \cdot \sum_{u \in \mathcal{N}_v} \mathbf{h}_u^{(k-1)}\right)$$

- Graph Isomorphism Network (GIN):
  (Xu et al., ICLR'2019)

$$\mathbf{h}_v^{(k)} = MLP^{(k)}\left(\left(1 + \epsilon^{(k)}\right)\mathbf{h}_v^{(k-1)} + \sum_{u \in \mathcal{N}_v} \mathbf{h}_u^{(k-1)}\right)$$

What are graph neural networks good for?

- GNNs are good for **short-range** tasks:

What are graph neural networks good for?

- GNNs are good for **short-range** tasks:
  - Paper subject classification (Cora/Citeseer/Pubmed, Sen et al., 2008)

What are graph neural networks good for?

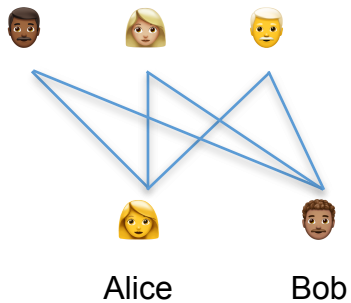
- GNNs are good for **short-range** tasks:
  - Paper subject classification (Cora/Citeseer/Pubmed, Sen et al., 2008)
  - Friendship/collaboration prediction (Open Graph Benchmark, Hu et al. 2020):

What are graph neural networks good for?

- GNNs are good for **short-range** tasks:
  - Paper subject classification (Cora/Citeseer/Pubmed, Sen et al., 2008)
  - Friendship/collaboration prediction (Open Graph Benchmark, Hu et al. 2020):

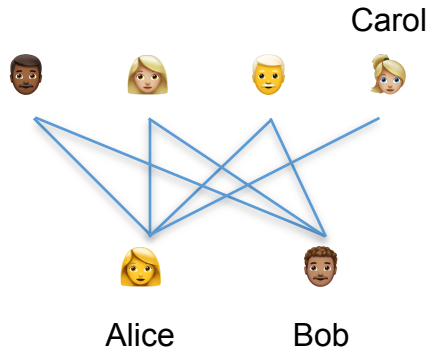
What are graph neural networks good for?

- GNNs are good for **short-range** tasks:
  - Paper subject classification (Cora/Citeseer/Pubmed, Sen et al., 2008)
  - Friendship/collaboration prediction (Open Graph Benchmark, Hu et al. 2020):



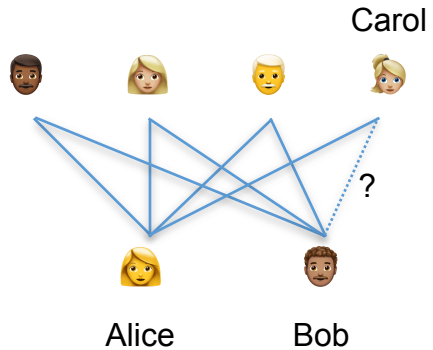
What are graph neural networks good for?

- GNNs are good for **short-range** tasks:
  - Paper subject classification (Cora/Citeseer/Pubmed, Sen et al., 2008)
  - Friendship/collaboration prediction (Open Graph Benchmark, Hu et al. 2020):



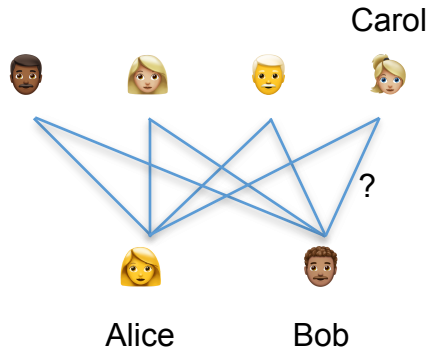
What are graph neural networks good for?

- GNNs are good for **short-range** tasks:
  - Paper subject classification (Cora/Citeseer/Pubmed, Sen et al., 2008)
  - Friendship/collaboration prediction (Open Graph Benchmark, Hu et al. 2020):



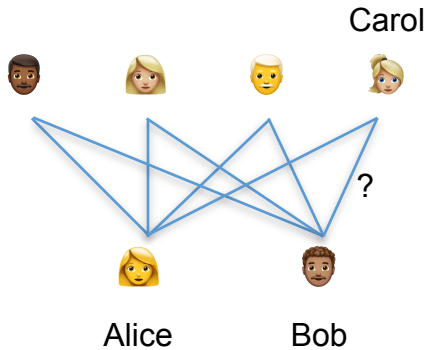
What are graph neural networks good for?

- GNNs are good for **short-range** tasks:
  - Paper subject classification (Cora/Citeseer/Pubmed, Sen et al., 2008)
  - Friendship/collaboration prediction (Open Graph Benchmark, Hu et al. 2020):



What are graph neural networks good for?

- GNNs are good for **short-range** tasks:
  - Paper subject classification (Cora/Citeseer/Pubmed, Sen et al., 2008)
  - Friendship/collaboration prediction (Open Graph Benchmark, Hu et al. 2020):

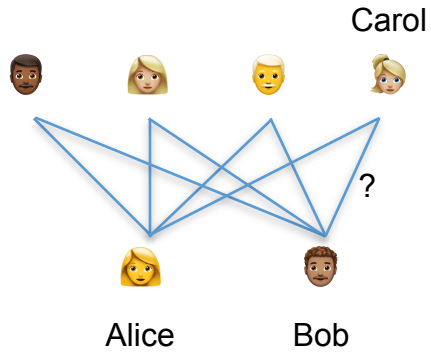


Very local property,

requires only 2-3 message-passing steps

What are graph neural networks good for?

- GNNs are good for **short-range** tasks:
  - Paper subject classification (Cora/Citeseer/Pubmed, Sen et al., 2008)
  - Friendship/collaboration prediction (Open Graph Benchmark, Hu et al. 2020):



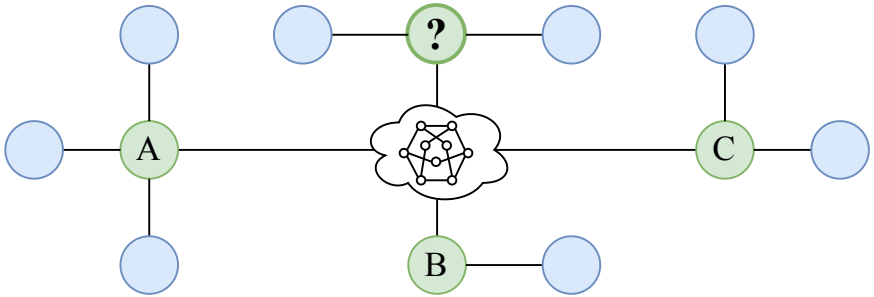
Very local property,

requires only 2-3 message-passing steps

- This work: but not that good for **long-range** tasks (4+ edges away interaction)

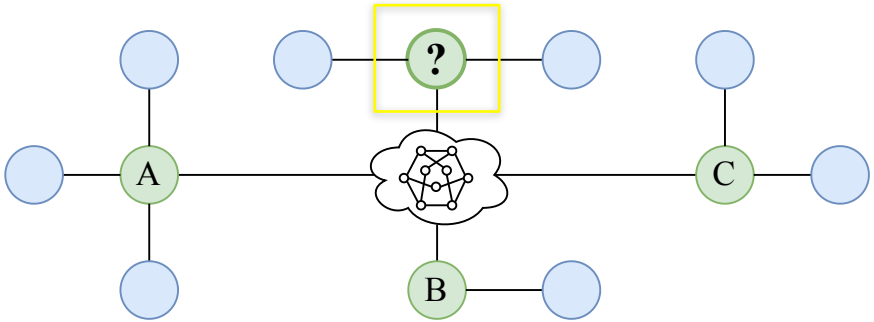
The NeighborsMatch problem

- Assume that we wish to predict a label for the target node



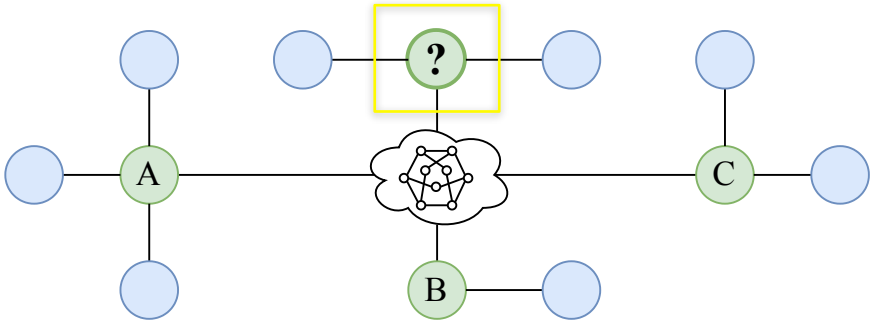
The NeighborsMatch problem

- Assume that we wish to predict a label for the target node



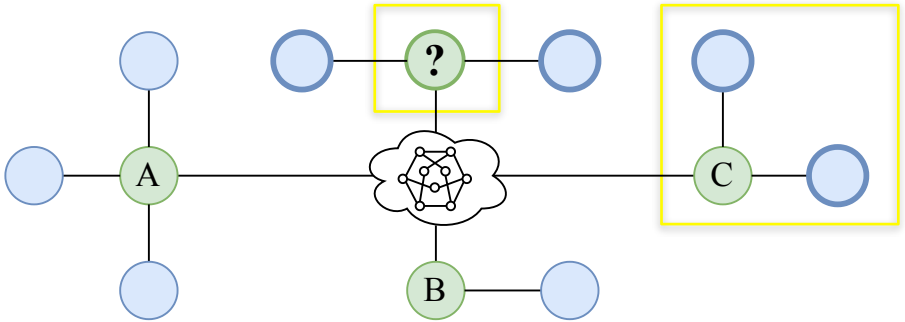
The NeighborsMatch problem

- Assume that we wish to predict a label for the target node
- The correct label is the label of the green node that has the same number of blue neighbors as the target node, in the same graph



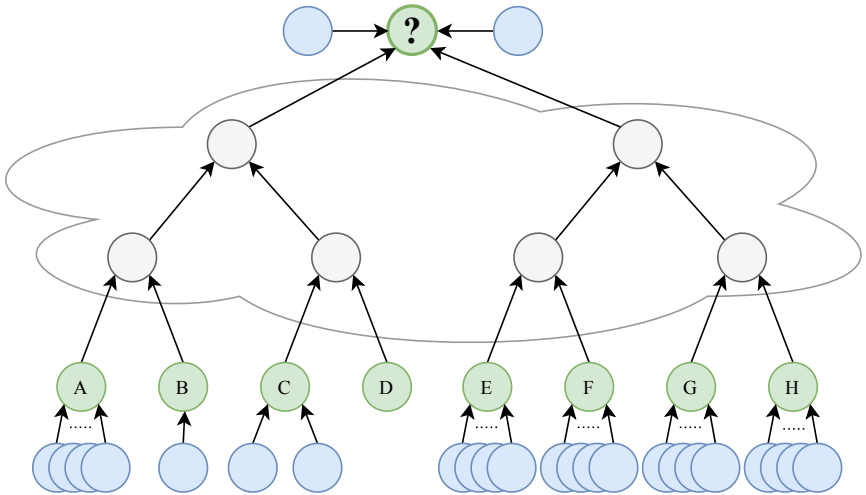
The NeighborsMatch problem

- Assume that we wish to predict a label for the target node
- The correct label is the label of the green node that has the same number of blue neighbors as the target node, in the same graph
  - In this example – **C**



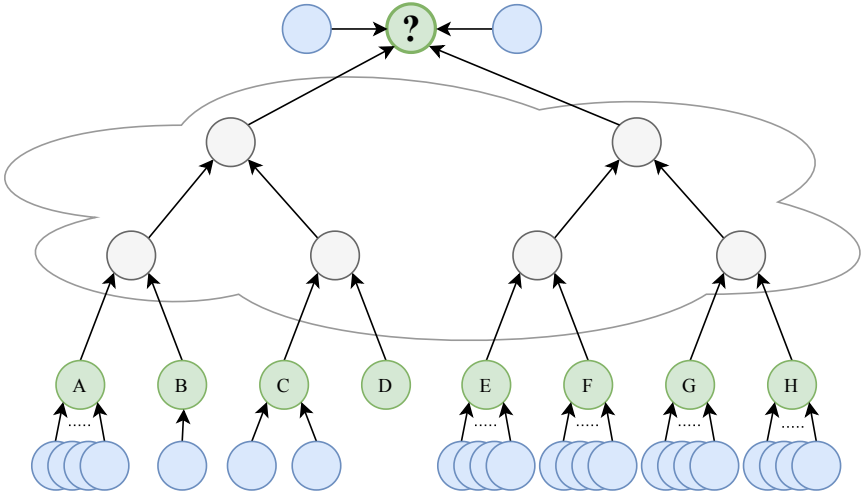
The GNN Bottleneck

From the perspective of the target node, the rest of the graph may look like a tree, where the target node itself is the root



The GNN Bottleneck

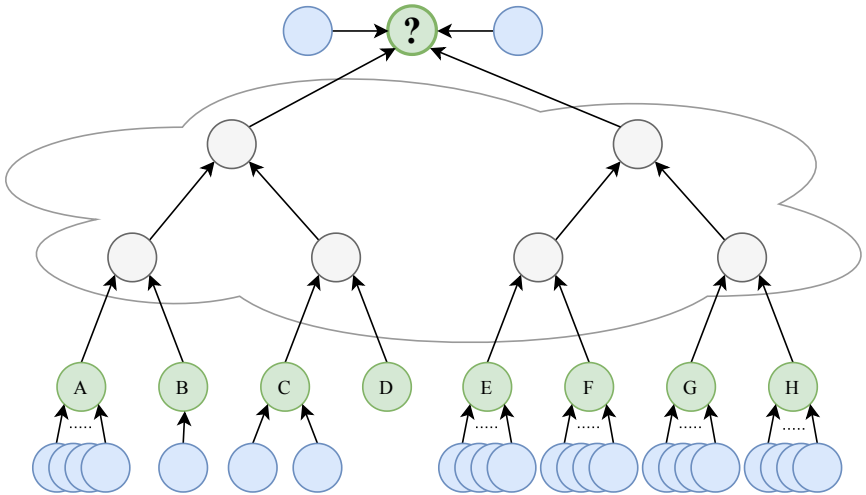
From the perspective of the target node, the rest of the graph may look like a tree, where the target node itself is the root



We need: $K > distance\left(C, target\right)$

The GNN Bottleneck

From the perspective of the target node, the rest of the graph may look like a tree, where the target node itself is the root



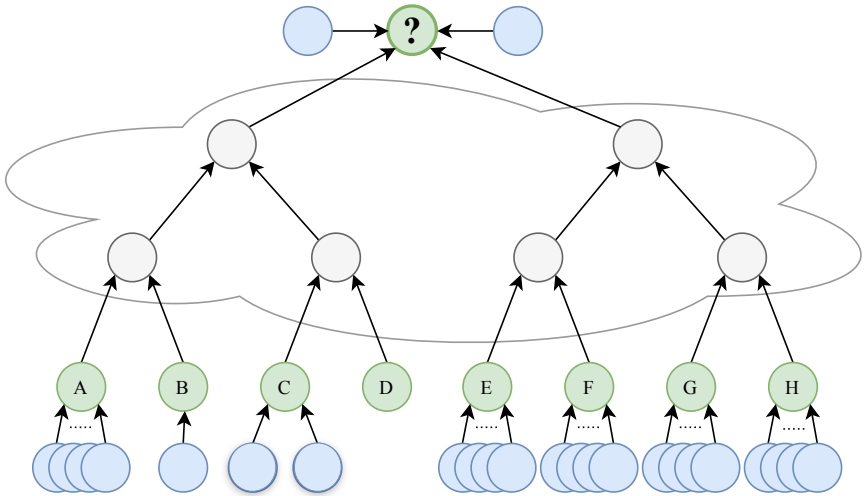
We need: $K > distance\left(C, target\right)$

In this case, we need at least $K \geq 4$ GNN layers for the information to reach the target node

The GNN Bottleneck

From the perspective of the target node, the rest of the graph may look like a tree, where the target node itself is the root

t=0



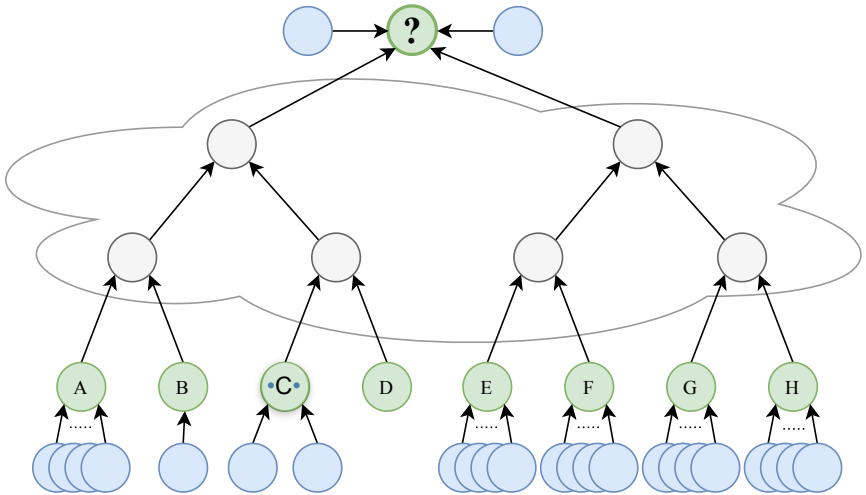
We need: $K > distance\left(C, target\right)$

In this case, we need at least $K \geq 4$ GNN layers for the information to reach the target node

The GNN Bottleneck

From the perspective of the target node, the rest of the graph may look like a tree, where the target node itself is the root

t=1



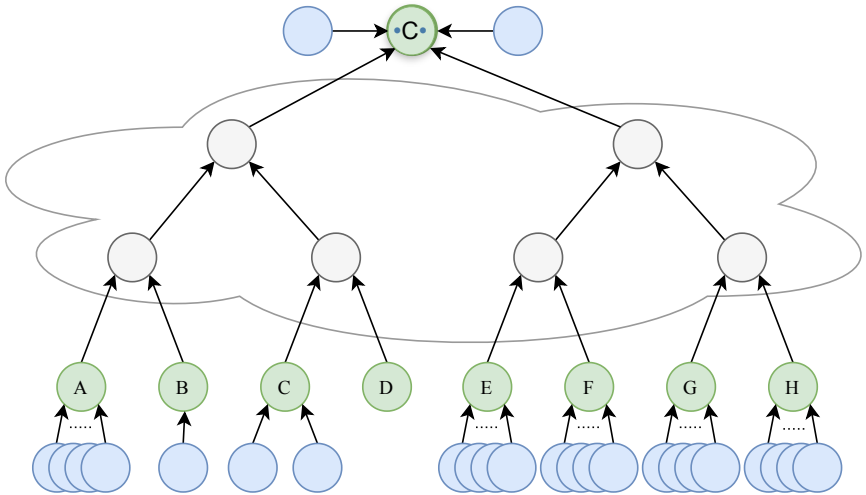
We need: $K > distance\left(C, target\right)$

In this case, we need at least $K \geq 4$ GNN layers for the information to reach the target node

The GNN Bottleneck

From the perspective of the target node, the rest of the graph may look like a tree, where the target node itself is the root

t=4

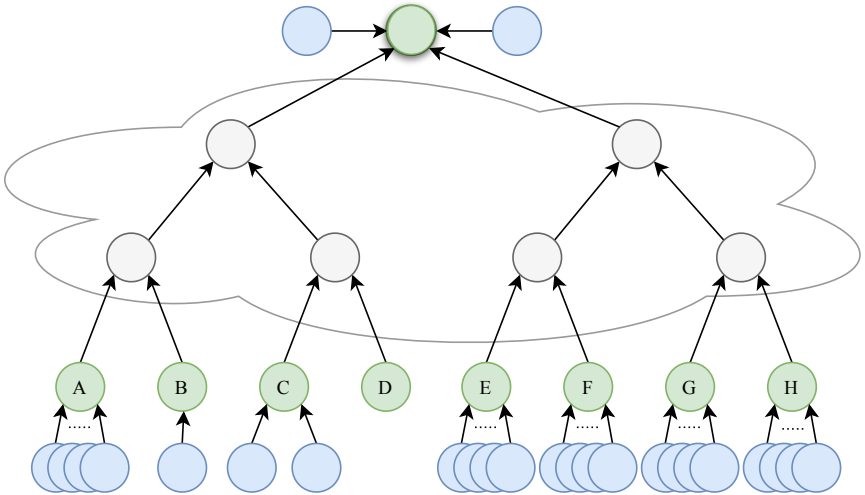


We need: $K > distance\left(C, target\right)$

In this case, we need at least $K \geq 4$ GNN layers for the information to reach the target node

The GNN Bottleneck

From the perspective of the target node, the rest of the graph may look like a tree, where the target node itself is the root

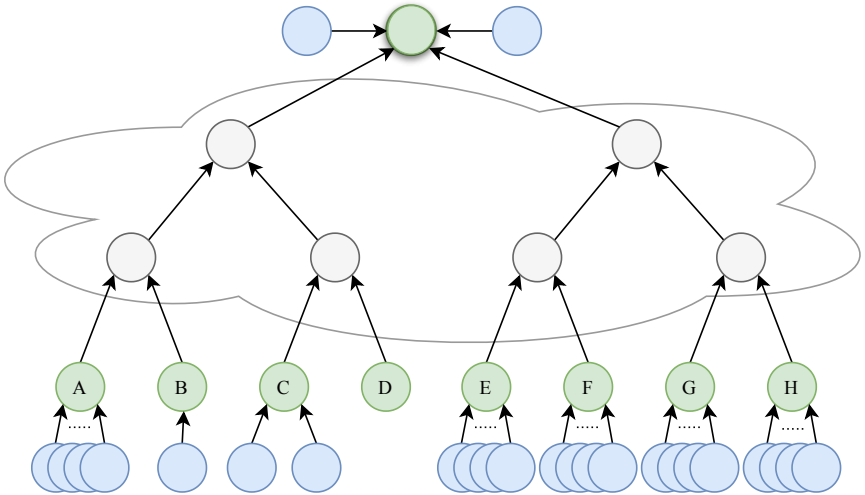


We need: $K > distance\left(C, target\right)$

In this case, we need at least $K \geq 4$ GNN layers for the information to reach the target node

The GNN Bottleneck

From the perspective of the target node, the rest of the graph may look like a tree, where the target node itself is the root



We need: $K > distance\left(C, target\right)$

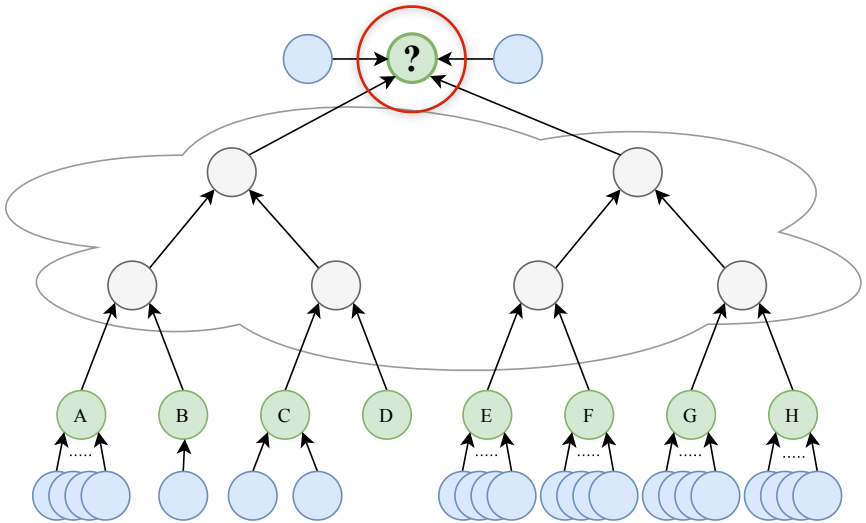
In this case, we need at least $K \geq 4$ GNN layers for the information to reach the target node

However, the receptive field of the target node grows **exponentially** with the number of layers

The GNN Bottleneck

From the perspective of the target node, the rest of the graph may look like a tree, where the target node itself is the root

t=0



We need: $K > distance\left(C, target\right)$

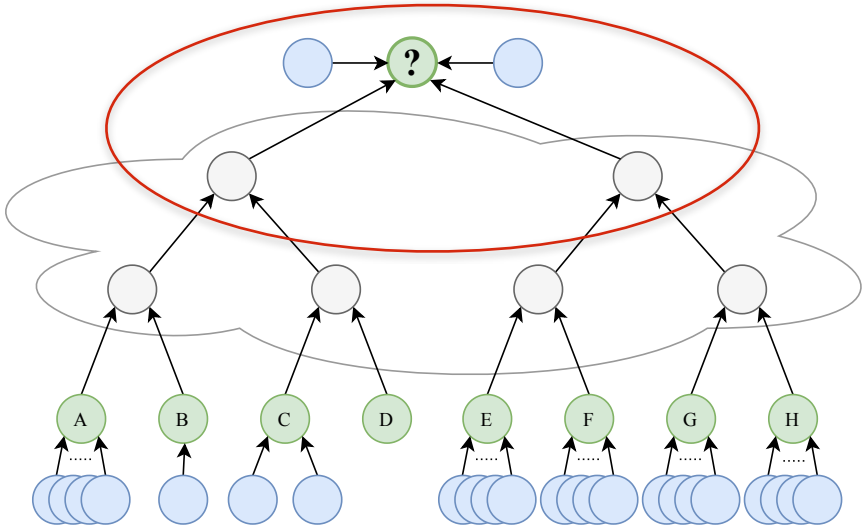
In this case, we need at least $K \geq 4$ GNN layers for the information to reach the target node

However, the receptive field of the target node grows **exponentially** with the number of layers

The GNN Bottleneck

From the perspective of the target node, the rest of the graph may look like a tree, where the target node itself is the root

t=1



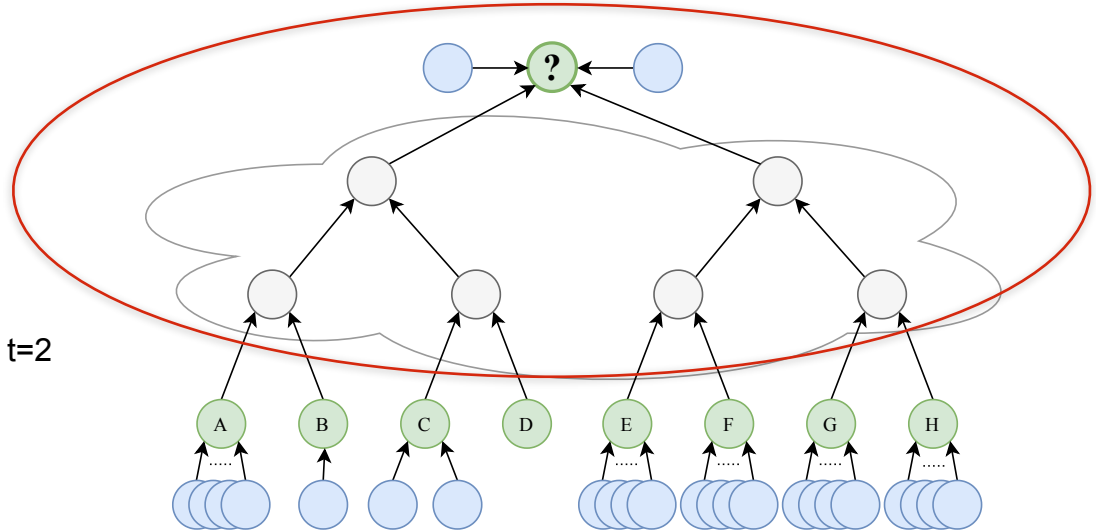
We need: $K > distance\left(C, target\right)$

In this case, we need at least $K \geq 4$ GNN layers for the information to reach the target node

However, the receptive field of the target node grows **exponentially** with the number of layers

The GNN Bottleneck

From the perspective of the target node, the rest of the graph may look like a tree, where the target node itself is the root



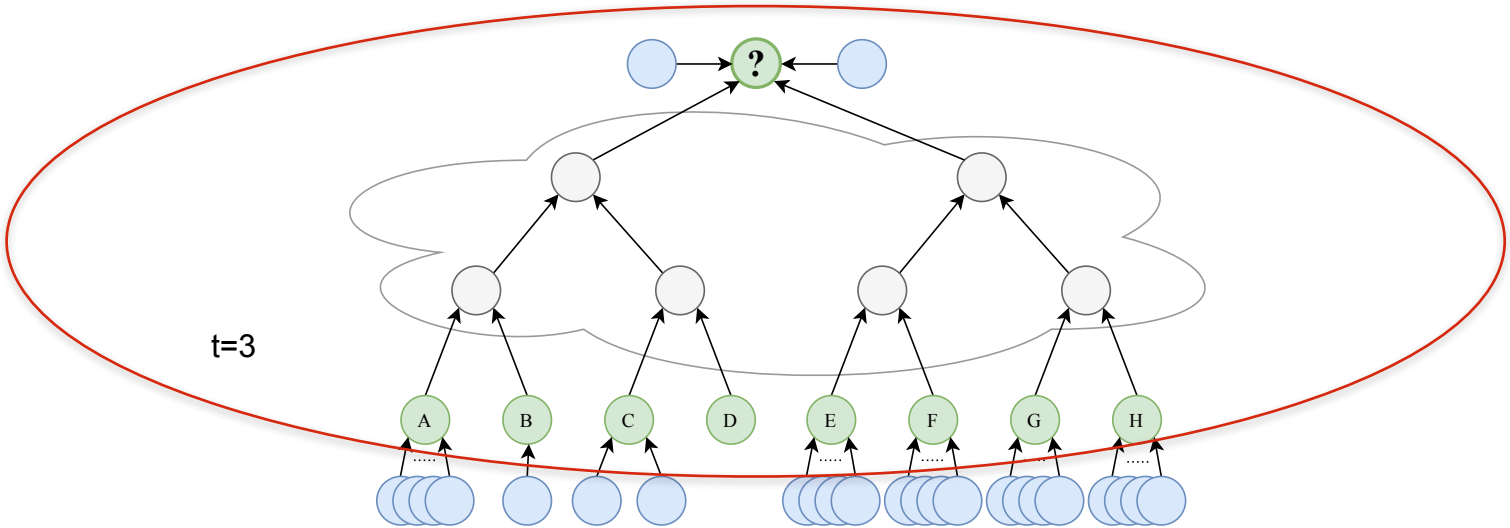
We need: $K > distance\left(C, target\right)$

In this case, we need at least $K \geq 4$ GNN layers for the information to reach the target node

However, the receptive field of the target node grows **exponentially** with the number of layers

The GNN Bottleneck

From the perspective of the target node, the rest of the graph may look like a tree, where the target node itself is the root



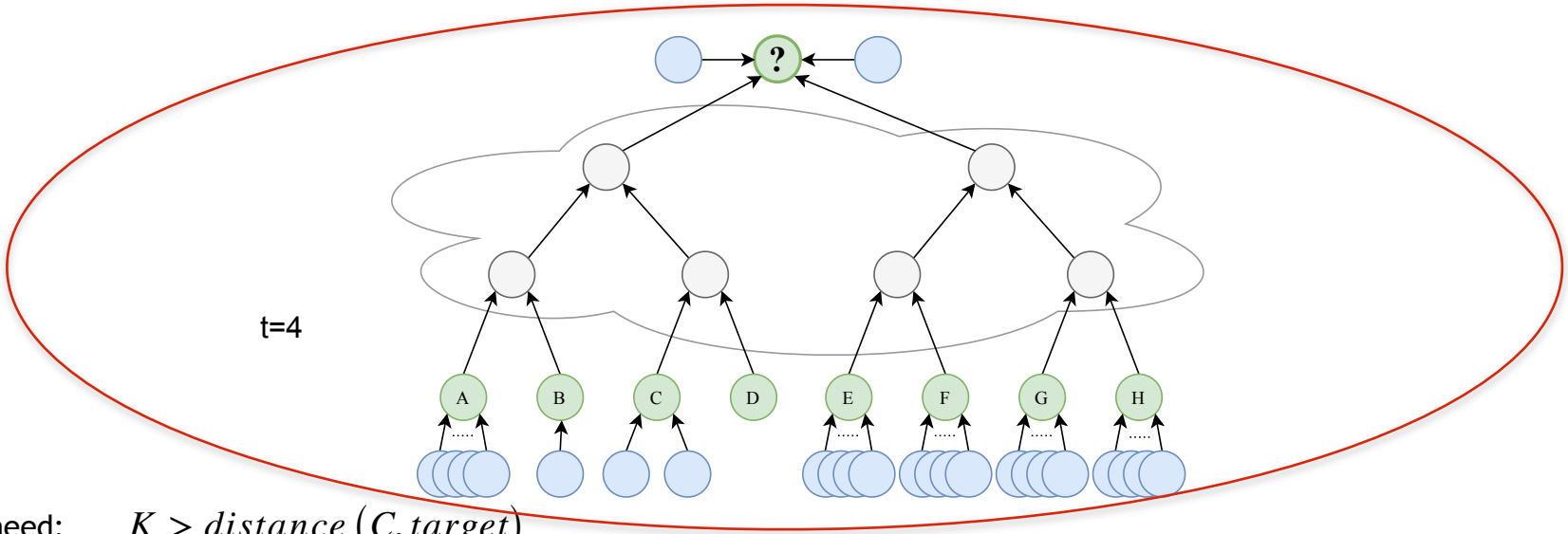
We need: $K > distance\left(C, target\right)$

In this case, we need at least $K \geq 4$ GNN layers for the information to reach the target node

However, the receptive field of the target node grows **exponentially** with the number of layers

The GNN Bottleneck

From the perspective of the target node, the rest of the graph may look like a tree, where the target node itself is the root



We need: $K > distance\left(C, target\right)$

In this case, we need at least $K \geq 4$ GNN layers for the information to reach the target node

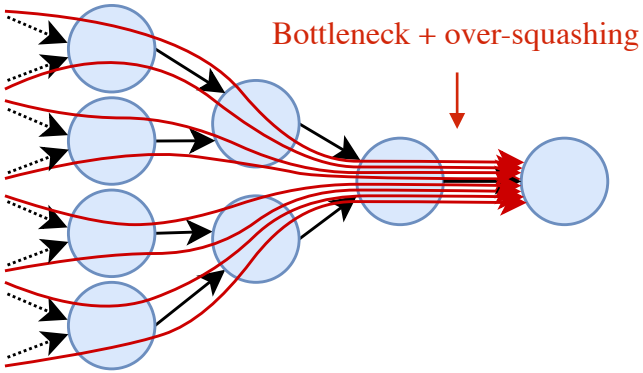
However, the receptive field of the target node grows **exponentially** with the number of layers

Over-squashing

To flow a message to a distance of **4**, we need to squash $\mathbf{O}\left(\mathbf{degree}^{\mathbf{4}}\right)$ messages into a single node representation (the representation of the target node).

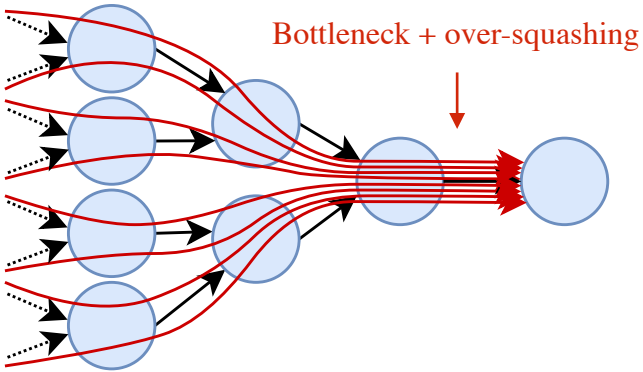
Over-squashing

To flow a message to a distance of 4, we need to squash $\mathbf{O}\left(\mathbf{degree}^{4}\right)$ messages into a single node representation (the representation of the target node).



Over-squashing

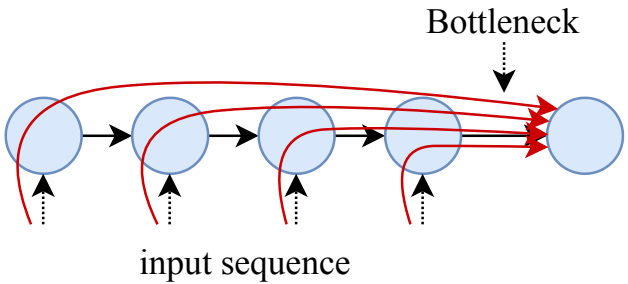
To flow a message to a distance of 4, we need to squash $\mathbf{O}\left(\mathbf{degree}^{4}\right)$ messages into a single node representation (the representation of the target node).



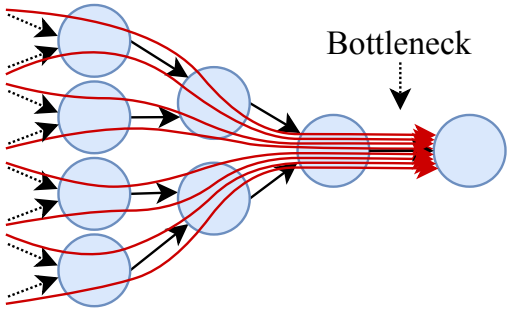
An exponential amount of information is squashed into a fixed-size vector.

Over-squashing

Actually, this is similar to the bottleneck of recurrent sequential models (before attention), except that the receptive field in RNNs grows **linearly**, while in GNNs it grows **exponentially**



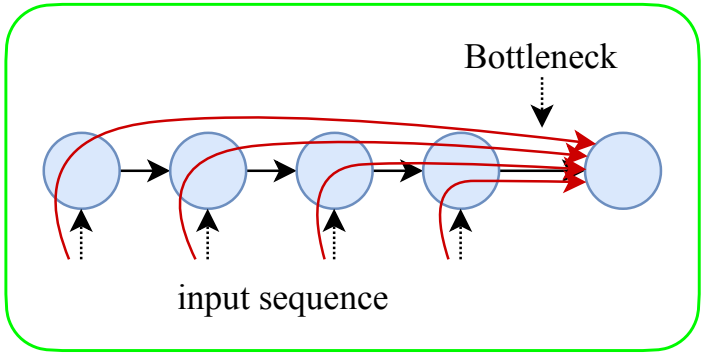
RNNs



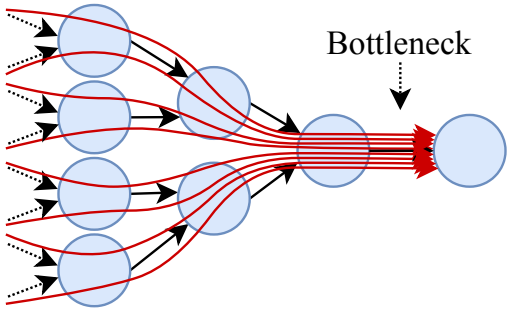
GNNs

Over-squashing

Actually, this is similar to the bottleneck of recurrent sequential models (before attention), except that the receptive field in RNNs grows **linearly**, while in GNNs it grows **exponentially**



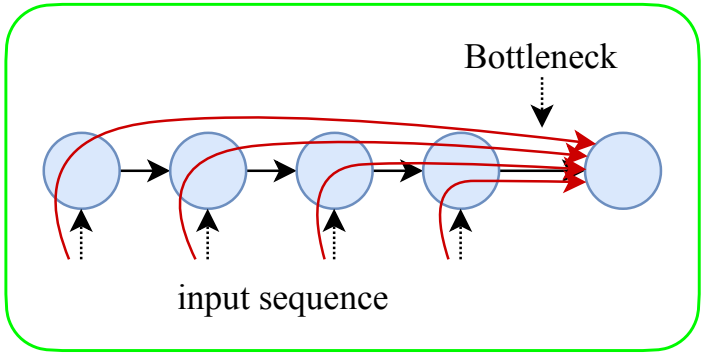
RNNs



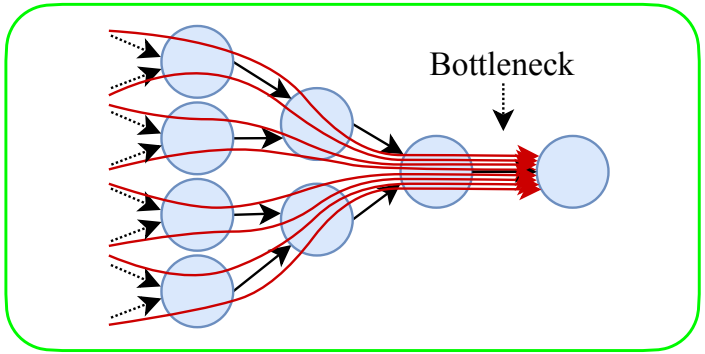
GNNs

Over-squashing

Actually, this is similar to the bottleneck of recurrent sequential models (before attention), except that the receptive field in RNNs grows **linearly**, while in GNNs it grows **exponentially**



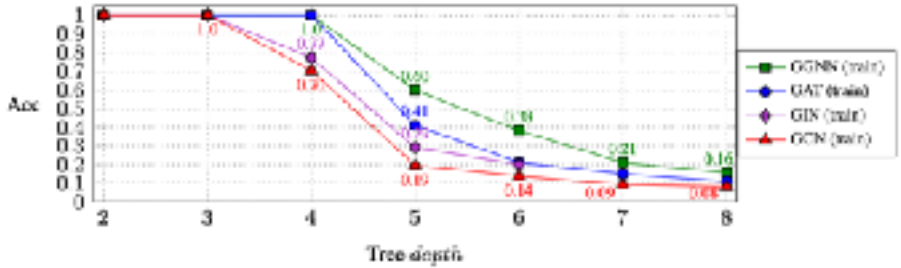
RNNs



GNNs

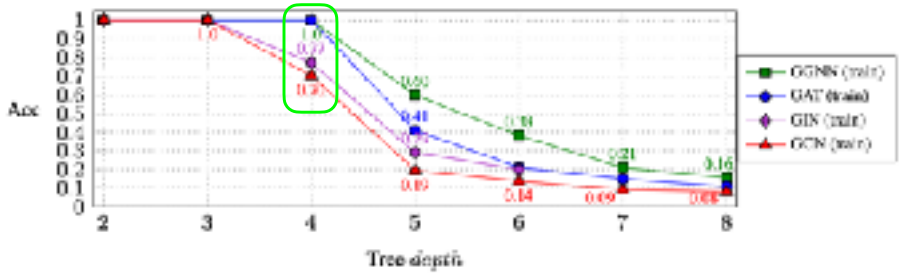
Over-squashing prevents GNNs from fitting the training data

- At depth=4, some GNNs cannot even reach 100% **training** accuracy



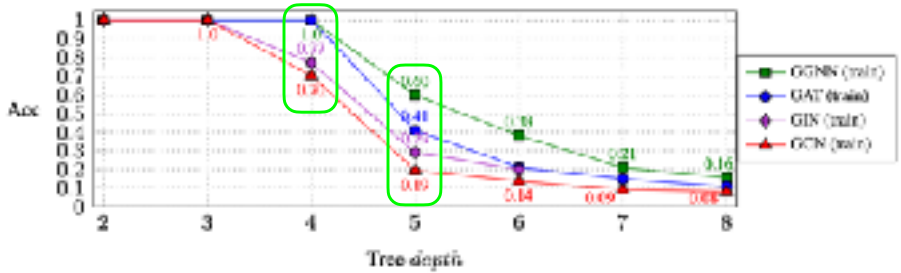
Over-squashing prevents GNNs from fitting the training data

- At depth=4, some GNNs cannot even reach 100% **training** accuracy



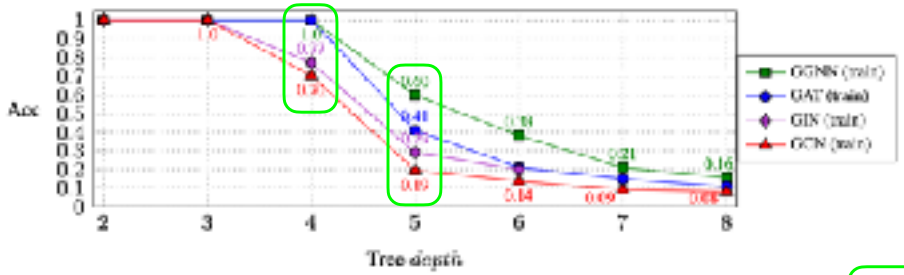
Over-squashing prevents GNNs from fitting the training data

- At depth=4, some GNNs cannot even reach 100% **training** accuracy



Over-squashing prevents GNNs from fitting the training data

- At depth=4, some GNNs cannot even reach 100% **training** accuracy

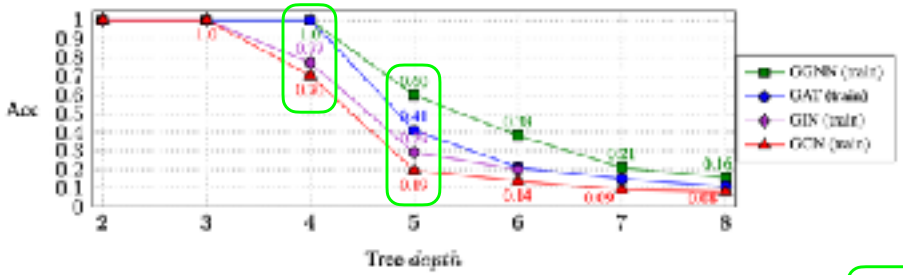


- (In the paper:) combinatorially, to fit the dataset, the dimension d must satisfy:

$$2^{32 \cdot d} > \frac{\left(2^{depth}\right)!}{2^{2^{depth}-1}}$$

Over-squashing prevents GNNs from fitting the training data

- At depth=4, some GNNs cannot even reach 100% **training** accuracy

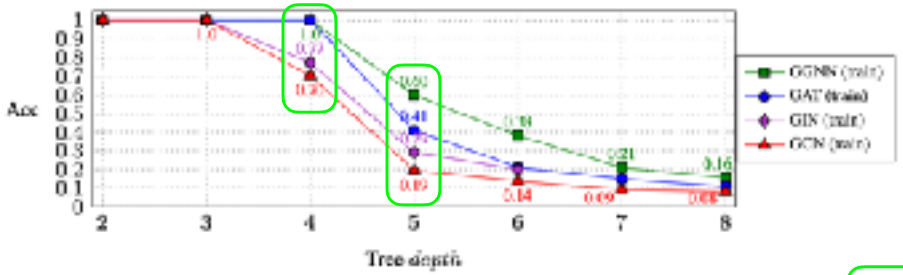


- (In the paper:) combinatorially, to fit the dataset, the dimension d must satisfy:
  - Such that there will be enough bits to express all different training examples

$$2^{32 \cdot d} > \frac{\left(2^{depth}\right)!}{2^{2^{depth}-1}}$$

Over-squashing prevents GNNs from fitting the training data

- At depth=4, some GNNs cannot even reach 100% **training** accuracy

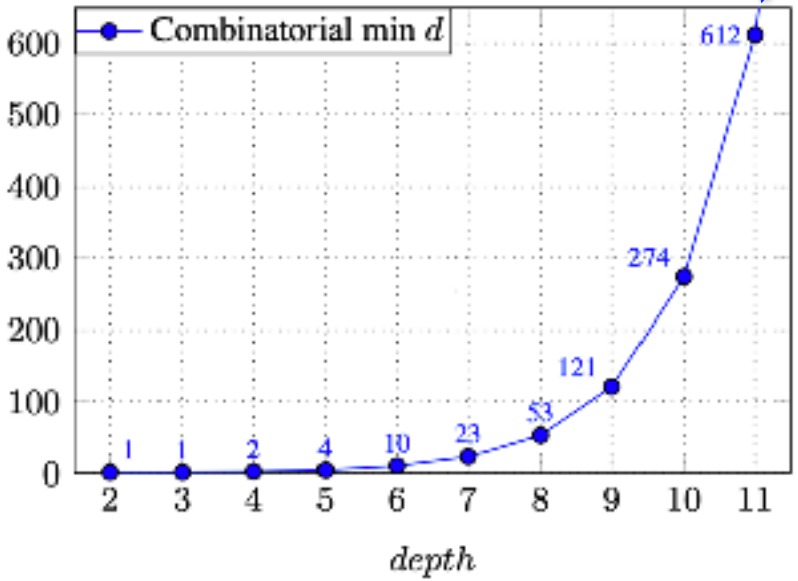


- (In the paper:) combinatorially, to fit the dataset, the dimension d must satisfy:
  - Such that there will be enough bits to express all different training examples
  - Otherwise, pigeonhole principle: some different examples will result in the same vector representation.

$$2^{32 \cdot d} > \frac{\left(2^{depth}\right)!}{2^{2^{depth}-1}}$$

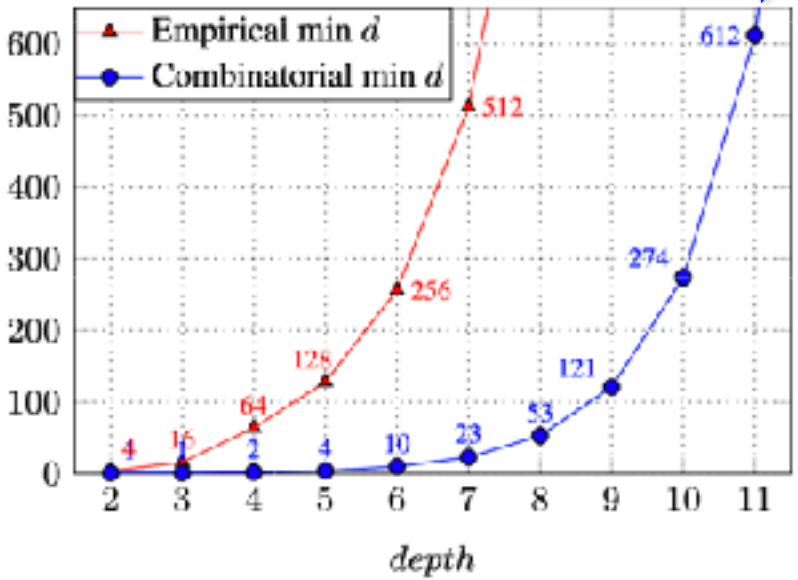
How long is “long-range”?

$$2^{32 \cdot d} > \frac{\left(2^{depth}\right)!}{2^{2^{depth}-1}}$$



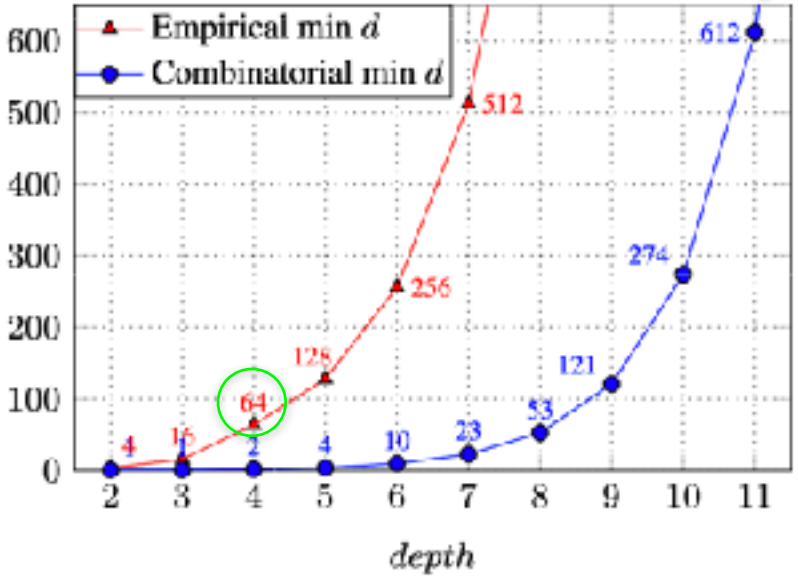
How long is “long-range”?

$$2^{32 \cdot d} > \frac{\left(2^{depth}\right)!}{2^{2^{depth}-1}}$$



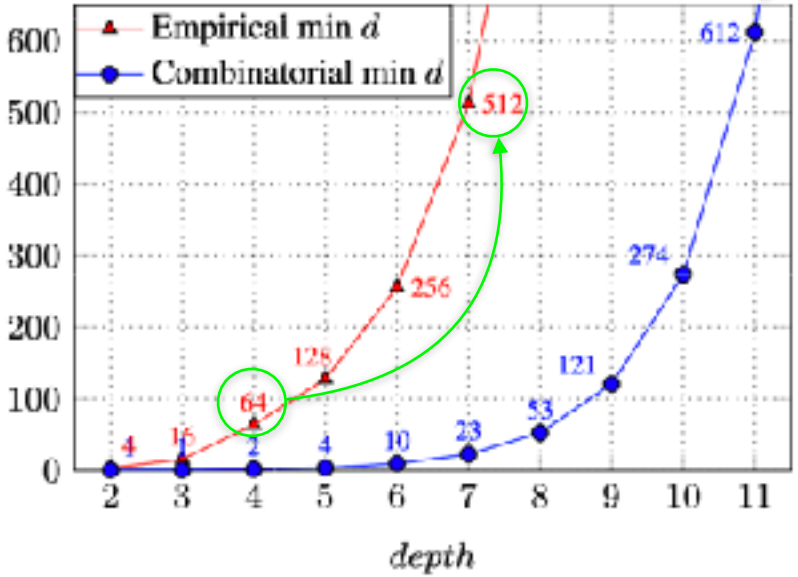
How long is “long-range”?

$$2^{32 \cdot d} > \frac{\left(2^{depth}\right)!}{2^{2^{depth}-1}}$$



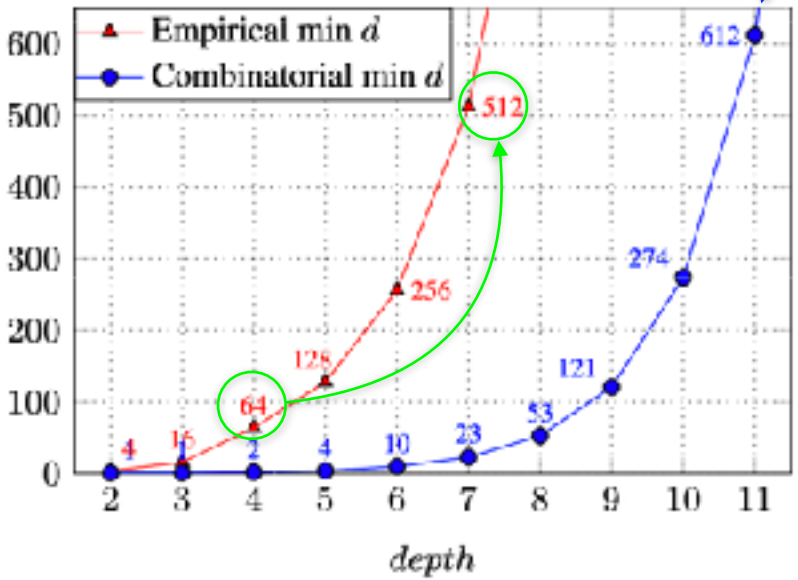
How long is “long-range”?

$$2^{32 \cdot d} > \frac{\left(2^{depth}\right)!}{2^{2^{depth}-1}}$$



How long is “long-range”?

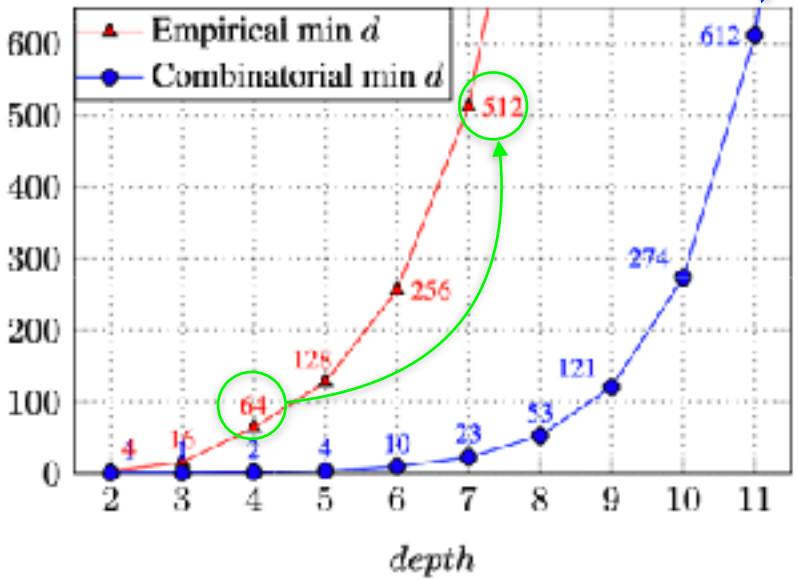
$$2^{32 \cdot d} > \frac{\left(2^{depth}\right)!}{2^{2^{depth}-1}}$$



< **2x** increase in message range costs **8x** increase in dimension

How long is “long-range”?

$$2^{32 \cdot d} > \frac{\left(2^{depth}\right)!}{2^{2^{depth}-1}}$$

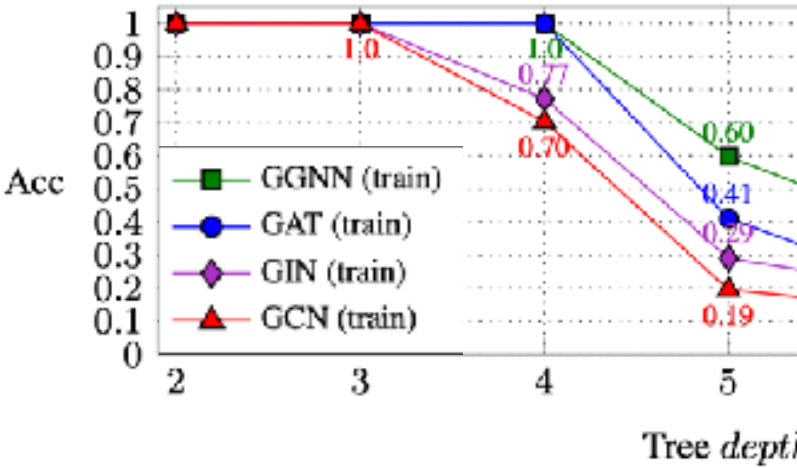


< **2x** increase in message range costs **8x** increase in dimension

Over-squashing is only partially treatable by increasing dimensions

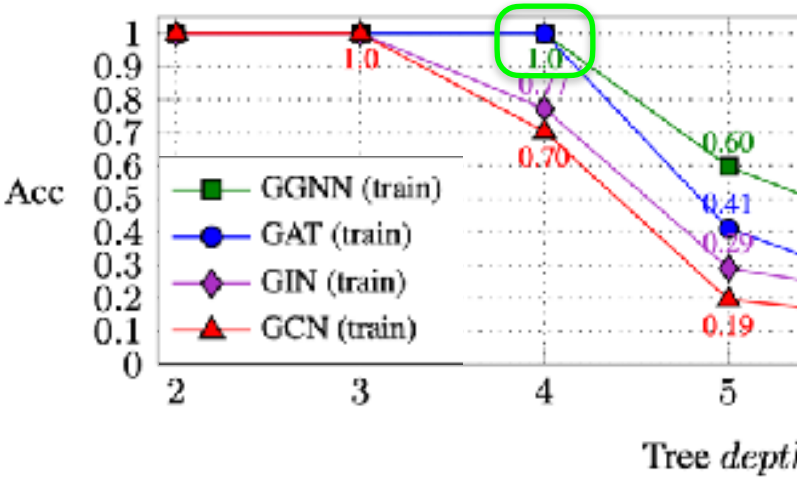
Different GNNs are affected by the bottleneck differently

- **GCN** and **GIN** suffer from over-squashing **more** than **GAT** and **GGNN**.



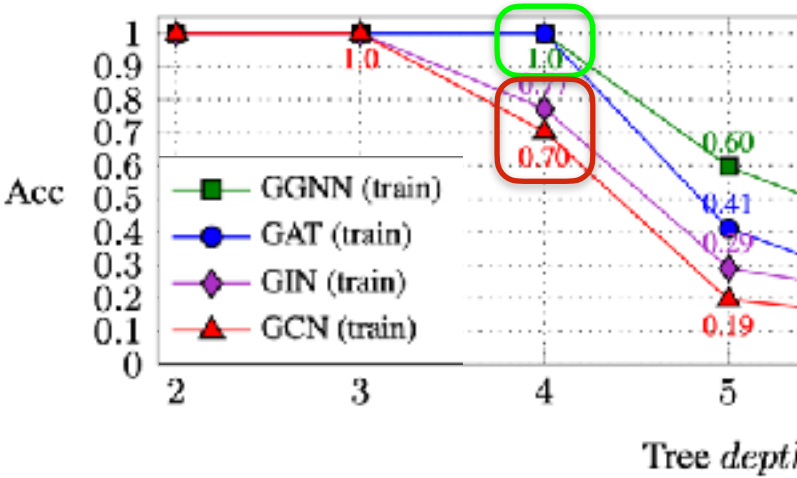
Different GNNs are affected by the bottleneck differently

- **GCN** and **GIN** suffer from over-squashing **more** than **GAT** and **GGNN**.



Different GNNs are affected by the bottleneck differently

- **GCN** and **GIN** suffer from over-squashing **more** than **GAT** and **GGNN**.

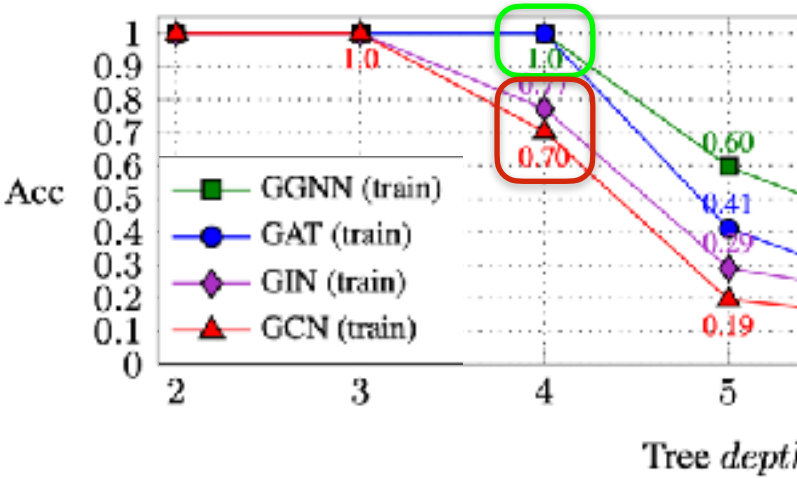


Different GNNs are affected by the bottleneck differently

- **GCN** and **GIN** suffer from over-squashing **more** than **GAT** and **GGNN**.

- **GCN** $$\mathbf{h}_v^{(k)} = ReLU\left(W^{(k)}\left(\frac{1}{c_v}\mathbf{h}_v^{(k-1)} + \sum_{u\in\mathcal{N}_v}\frac{1}{c_{v,u}}\mathbf{h}_u^{(k-1)}\right)\right)$$

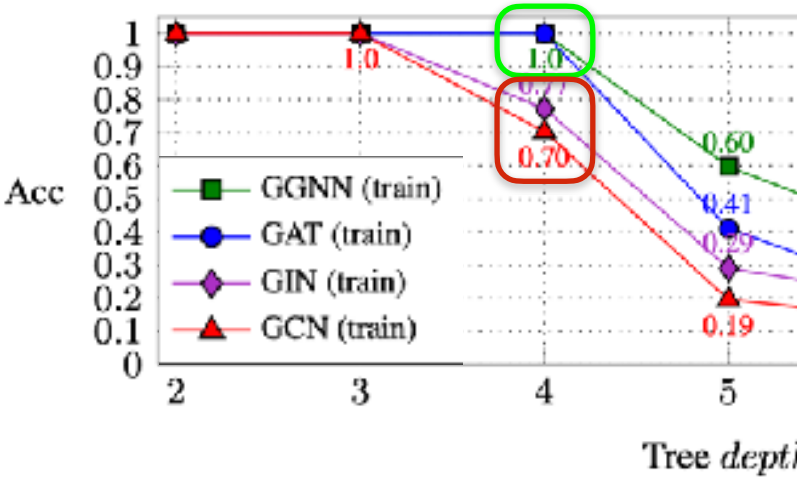
- **GIN** $$\mathbf{h}_v^{(k)} = MLP^{(k)}\left(\left(1+\epsilon^{(k)}\right)\mathbf{h}_v^{(k-1)} + \sum_{u\in\mathcal{N}_v}\mathbf{h}_u^{(k-1)}\right)$$



Different GNNs are affected by the bottleneck differently

- **GCN** and **GIN** suffer from over-squashing **more** than **GAT** and **GGNN**.

- **GCN** $\mathbf{h}_v^{(k)} = ReLU\left(W^{(k)}\left(\frac{1}{c_v}\mathbf{h}_v^{(k-1)} + \sum_{u\in\mathcal{N}_v}\frac{1}{c_{v,u}}\mathbf{h}_u^{(k-1)}\right)\right)$
- **GIN** $\mathbf{h}_v^{(k)} = MLP^{(k)}\left(\left(1+\epsilon^{(k)}\right)\mathbf{h}_v^{(k-1)} + \sum_{u\in\mathcal{N}_v}\mathbf{h}_u^{(k-1)}\right)$
- **GGNN** $\mathbf{h}_v^{(k)} = GRU\left(\mathbf{h}_v^{(k-1)}, \sum_{u\in\mathcal{N}_v} W_{neighbor}\mathbf{h}_u^{(k-1)}\right)$



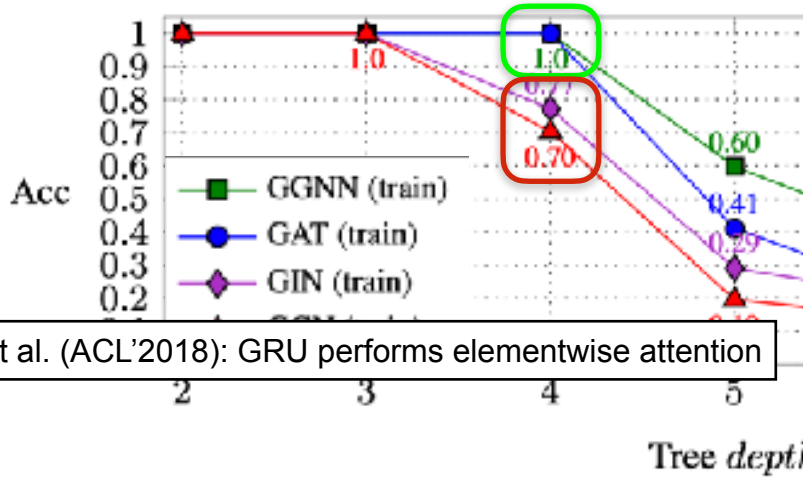
Different GNNs are affected by the bottleneck differently

- **GCN** and **GIN** suffer from over-squashing **more** than **GAT** and **GGNN**.

- **GCN** $\mathbf{h}_v^{(k)} = ReLU\left(W^{(k)}\left(\frac{1}{c_v}\mathbf{h}_v^{(k-1)} + \sum_{u\in\mathcal{N}_v}\frac{1}{c_{v,u}}\mathbf{h}_u^{(k-1)}\right)\right)$

- **GIN** $\mathbf{h}_v^{(k)} = MLP^{(k)}\left(\left(1+\epsilon^{(k)}\right)\mathbf{h}_v^{(k-1)} + \sum_{u\in\mathcal{N}_v}\mathbf{h}_u^{(k-1)}\right)$

- **GGNN** $\mathbf{h}_v^{(k)} = GRU\left(\mathbf{h}_v^{(k-1)}, \sum_{u\in\mathcal{N}_v} W_{neighbor}\mathbf{h}_u^{(k-1)}\right)$

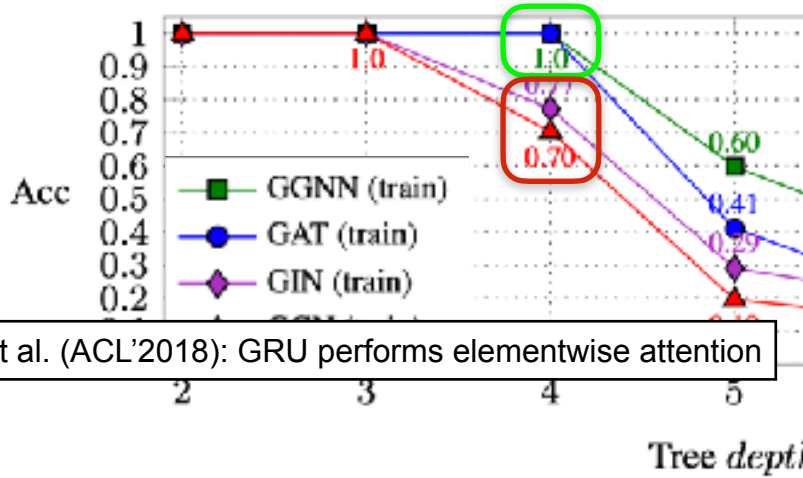


Levy et al. (ACL'2018): GRU performs elementwise attention

Different GNNs are affected by the bottleneck differently

- **GCN** and **GIN** suffer from over-squashing **more** than **GAT** and **GGNN**.

- **GCN** $\mathbf{h}_v^{(k)} = ReLU\left(W^{(k)}\left(\frac{1}{c_v}\mathbf{h}_v^{(k-1)} + \sum_{u\in\mathcal{N}_v}\frac{1}{c_{v,u}}\mathbf{h}_u^{(k-1)}\right)\right)$
- **GIN** $\mathbf{h}_v^{(k)} = MLP^{(k)}\left(\left(1+\epsilon^{(k)}\right)\mathbf{h}_v^{(k-1)} + \sum_{u\in\mathcal{N}_v}\mathbf{h}_u^{(k-1)}\right)$
- **GGNN** $\mathbf{h}_v^{(k)} = GRU\left(\mathbf{h}_v^{(k-1)}, \sum_{u\in\mathcal{N}_v} W_{neighbor}\mathbf{h}_u^{(k-1)}\right)$
- **GAT** $\mathbf{h}_v^{(k)} = ReLU\left(MultiHeadAttention\left(\mathcal{N}_v \mid \mathbf{h}_v^{(k-1)}\right)\right)$

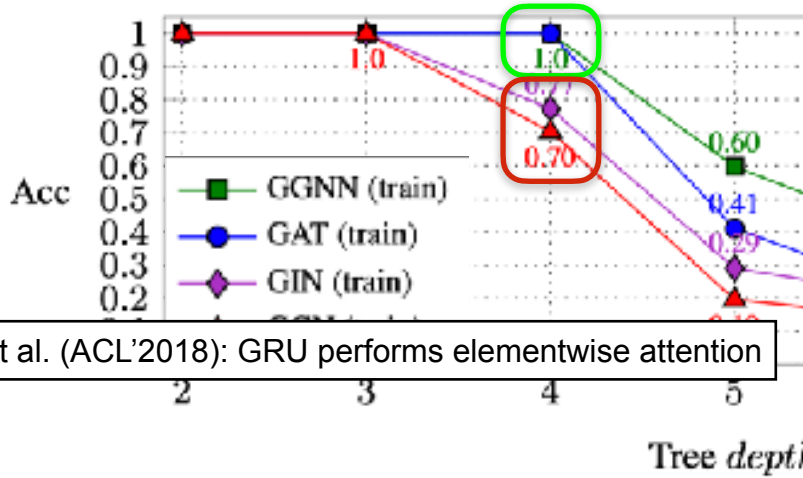


Levy et al. (ACL'2018): GRU performs elementwise attention

Different GNNs are affected by the bottleneck differently

- **GCN** and **GIN** suffer from over-squashing **more** than **GAT** and **GGNN**.

- **GCN** $\mathbf{h}_v^{(k)} = ReLU\left(W^{(k)}\left(\frac{1}{c_v}\mathbf{h}_v^{(k-1)} + \sum_{u\in\mathcal{N}_v}\frac{1}{c_{v,u}}\mathbf{h}_u^{(k-1)}\right)\right)$
- **GIN** $\mathbf{h}_v^{(k)} = MLP^{(k)}\left(\left(1+\epsilon^{(k)}\right)\mathbf{h}_v^{(k-1)} + \sum_{u\in\mathcal{N}_v}\mathbf{h}_u^{(k-1)}\right)$
- **GGNN** $\mathbf{h}_v^{(k)} = GRU\left(\mathbf{h}_v^{(k-1)}, \sum_{u\in\mathcal{N}_v} W_{neighbor}\mathbf{h}_u^{(k-1)}\right)$
- **GAT** $\mathbf{h}_v^{(k)} = ReLU\left(MultiHeadAttention\left(\mathcal{N}_v \mid \mathbf{h}_v^{(k-1)}\right)\right)$

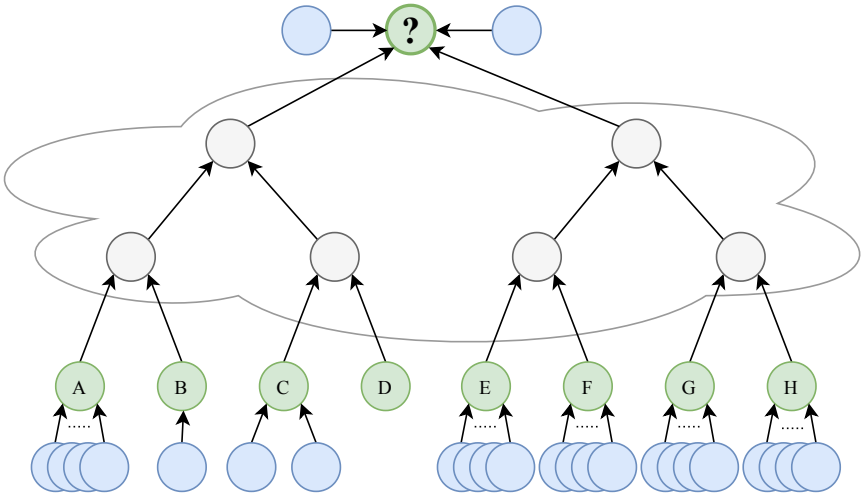


Levy et al. (ACL'2018): GRU performs elementwise attention

The reason is the multihead, not the attention
("How Attentive are Graph Attention Networks?", 2021)

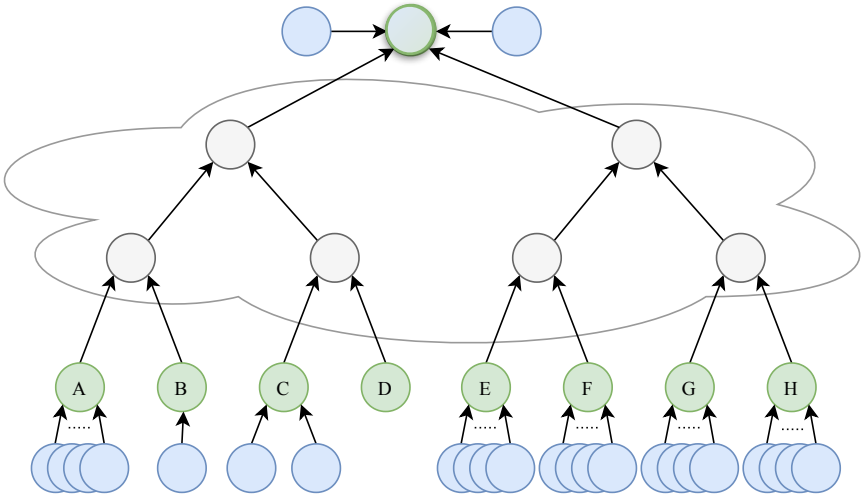
Different GNNs are affected by the bottleneck differently

- At the **t=4**, all messages reach the target node



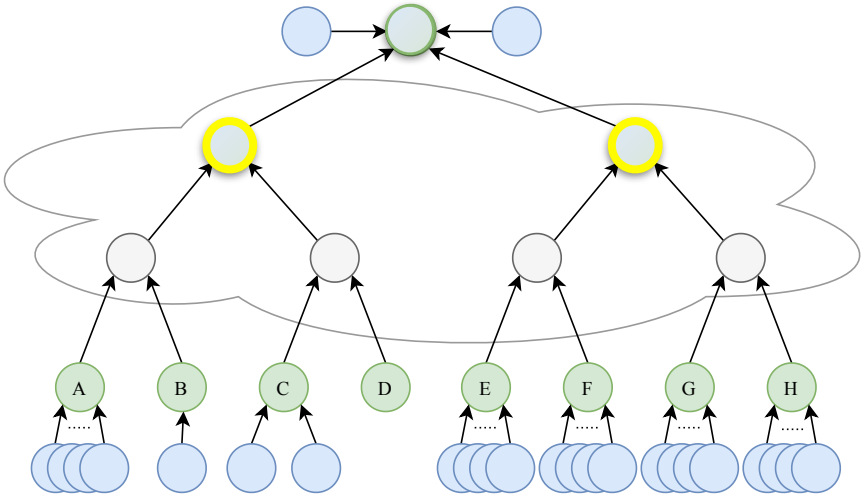
Different GNNs are affected by the bottleneck differently

- At the **t=4**, all messages reach the target node



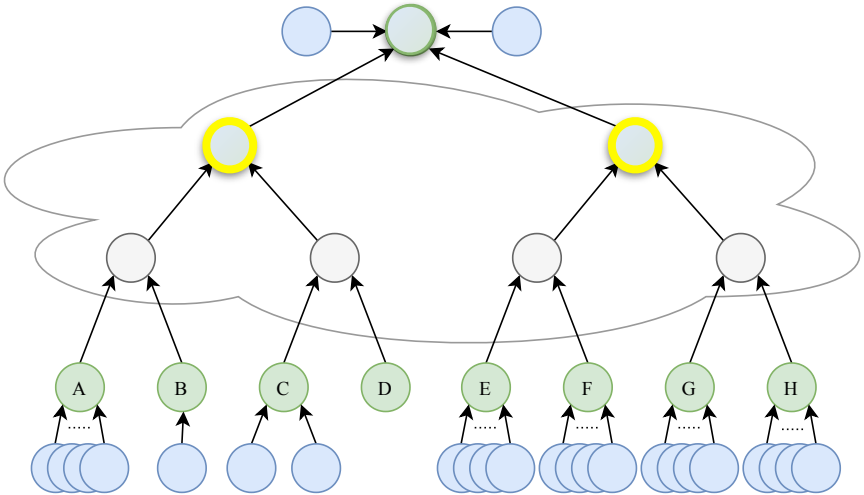
Different GNNs are affected by the bottleneck differently

- At the **t=4**, all messages reach the target node
- One of the branches is irrelevant, because the solution is in the other branch



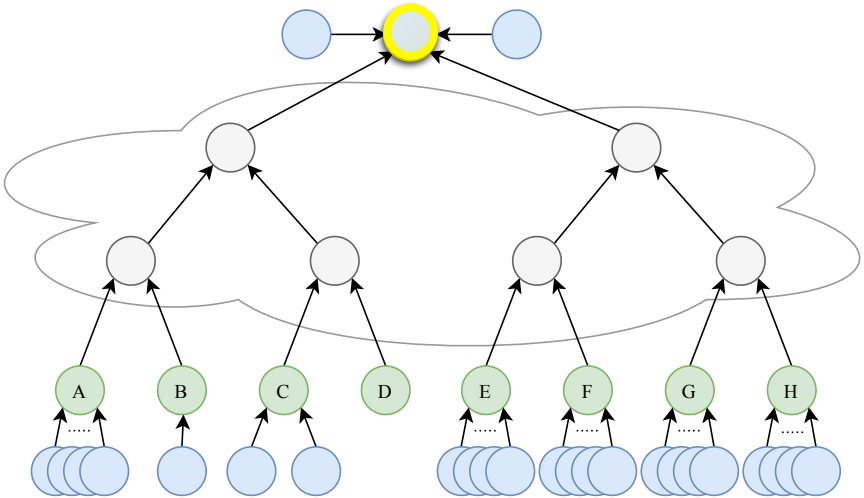
Different GNNs are affected by the bottleneck differently

- At the **t=4**, all messages reach the target node
- One of the branches is irrelevant, because the solution is in the other branch
- GIN and GCN **sum both** incoming edges



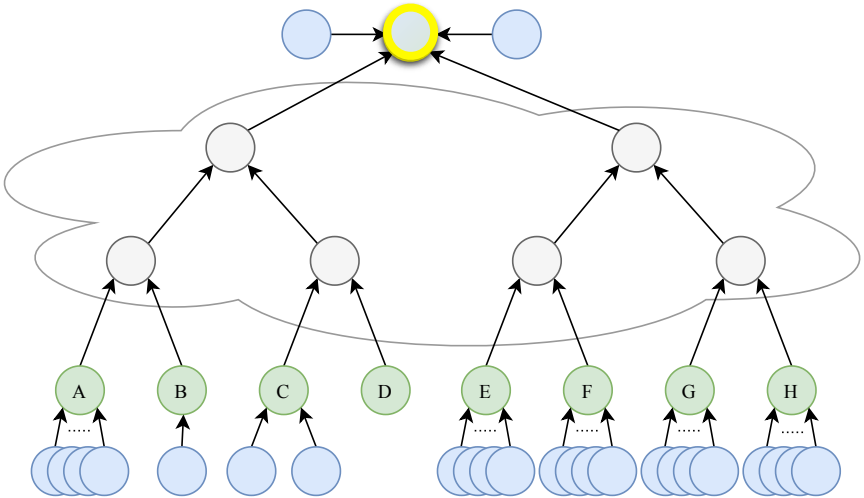
Different GNNs are affected by the bottleneck differently

- At the **t=4**, all messages reach the target node
- One of the branches is irrelevant, because the solution is in the other branch
- GIN and GCN **sum both** incoming edges



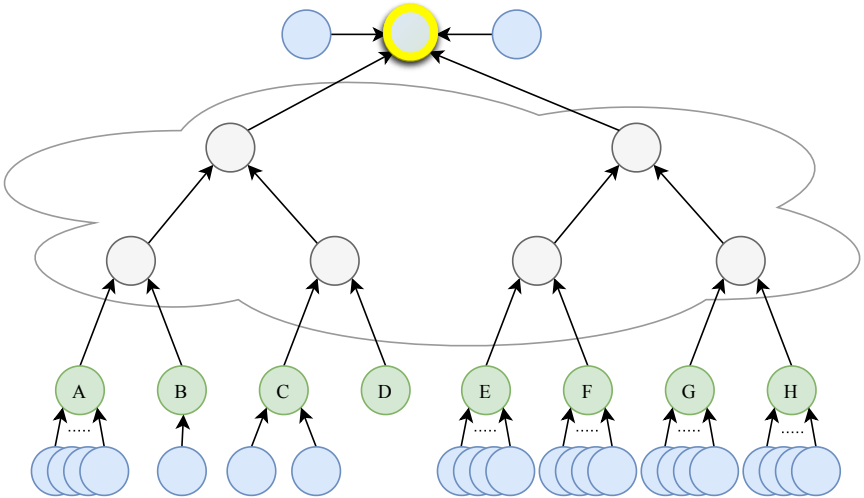
Different GNNs are affected by the bottleneck differently

- At the **t=4**, all messages reach the target node
- One of the branches is irrelevant, because the solution is in the other branch
- GIN and GCN **sum both** incoming edges
- GAT and GGNN can select to absorb a single edge.



Different GNNs are affected by the bottleneck differently

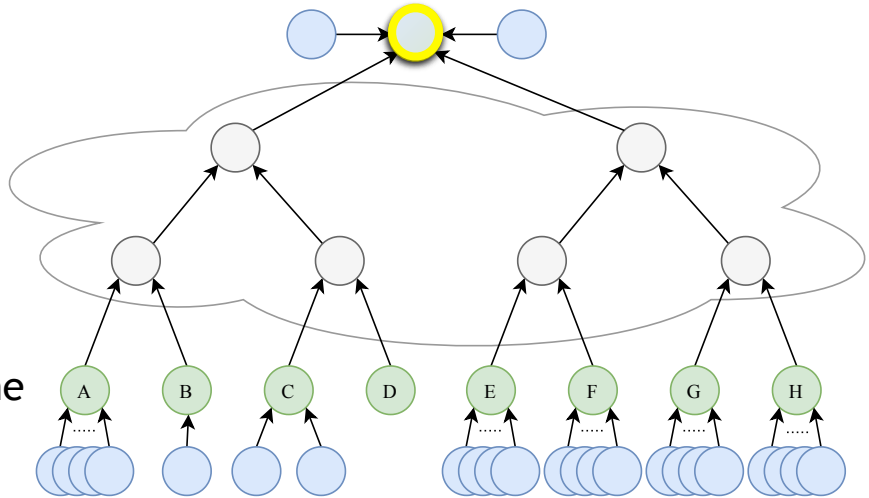
- At the **t=4**, all messages reach the target node
- One of the branches is irrelevant, because the solution is in the other branch
- GIN and GCN **sum both** incoming edges
- GAT and GGNN can select to absorb a single edge.



Different GNNs are affected by the bottleneck differently

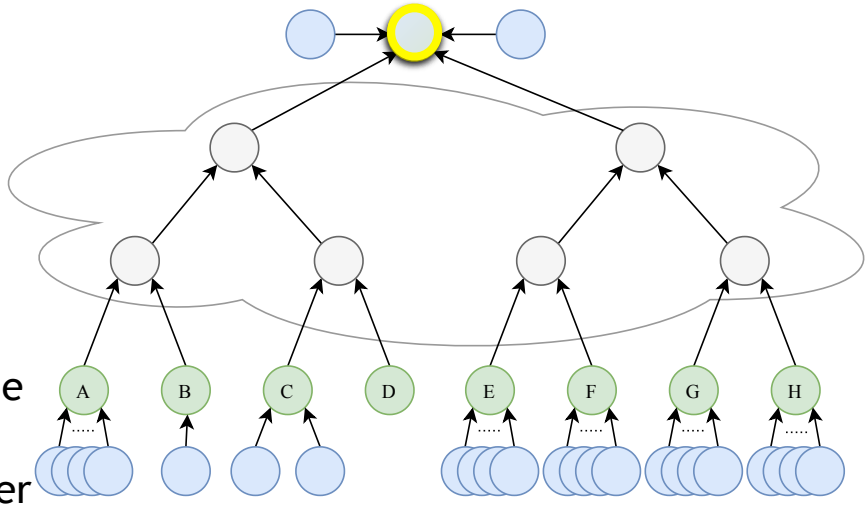
- At the **t=4**, all messages reach the target node
- One of the branches is irrelevant, because the solution is in the other branch
- GIN and GCN **sum both** incoming edges
- GAT and GGNN can select to absorb a single edge.

  ⇒ The effective bottleneck is only **half** of the effect on GIN and GCN



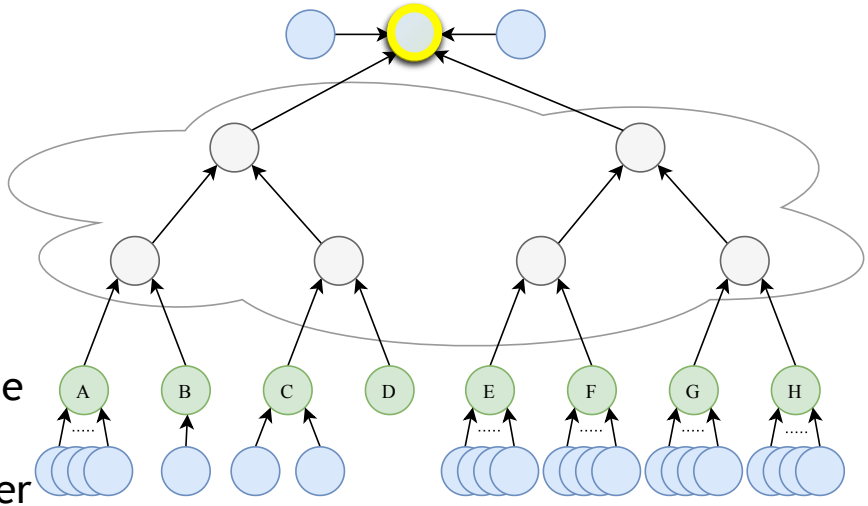
Different GNNs are affected by the bottleneck differently

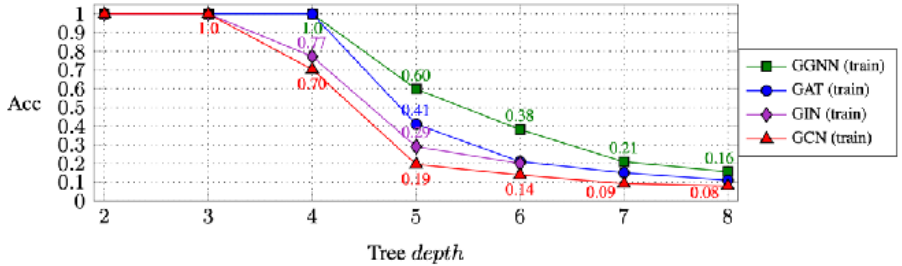
- At the **t=4**, all messages reach the target node
- One of the branches is irrelevant, because the solution is in the other branch
- GIN and GCN **sum both** incoming edges
- GAT and GGNN can select to absorb a single edge.
  - ⇒ The effective bottleneck is only **half** of the effect on GIN and GCN
  - ⇒ Thus, they succeed in a depth that is larger by 1



Different GNNs are affected by the bottleneck differently

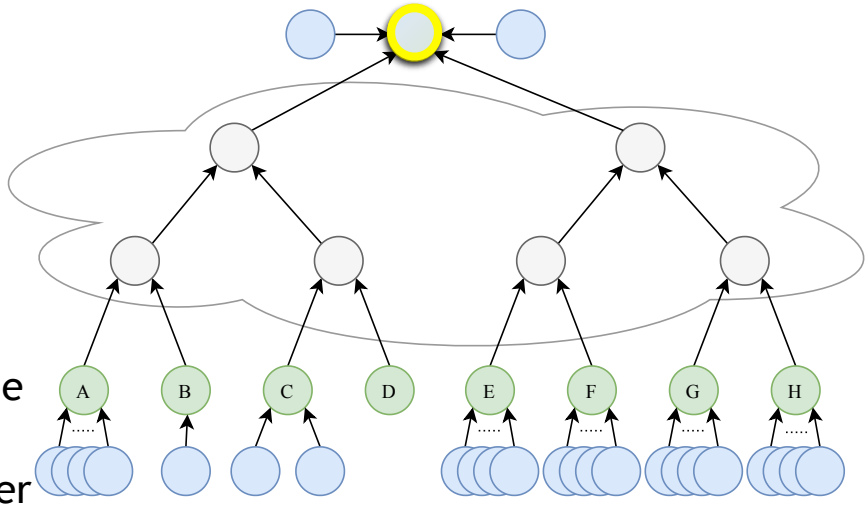
- At the **t=4**, all messages reach the target node
- One of the branches is irrelevant, because the solution is in the other branch
- GIN and GCN **sum both** incoming edges
- GAT and GGNN can select to absorb a single edge.
  - ⇒ The effective bottleneck is only **half** of the effect on GIN and GCN
  - ⇒ Thus, they succeed in a depth that is larger by 1

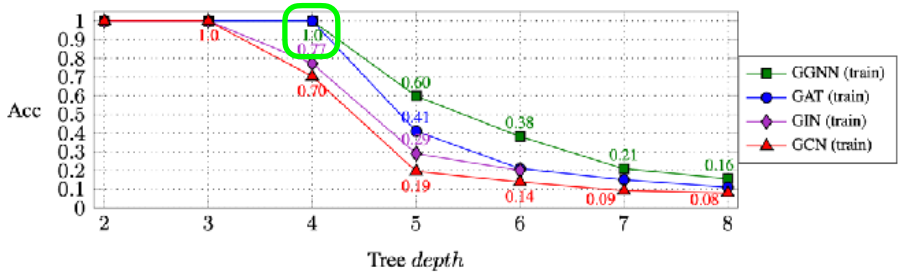




Different GNNs are affected by the bottleneck differently

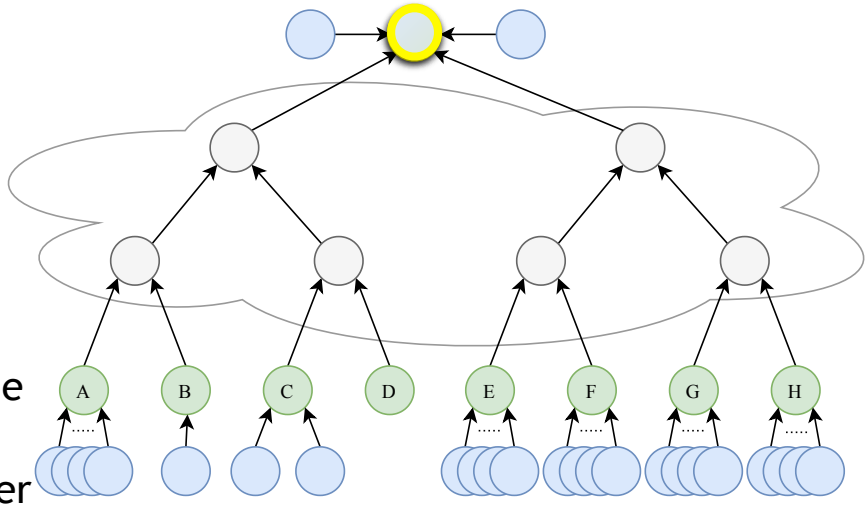
- At the **t=4**, all messages reach the target node
- One of the branches is irrelevant, because the solution is in the other branch
- GIN and GCN **sum both** incoming edges
- GAT and GGNN can select to absorb a single edge.
  - ⇒ The effective bottleneck is only **half** of the effect on GIN and GCN
  - ⇒ Thus, they succeed in a depth that is larger by 1

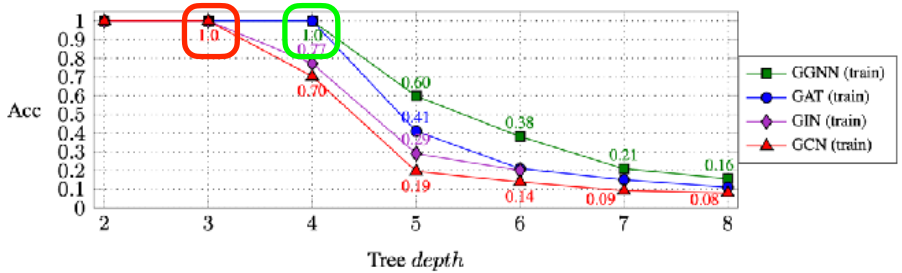




Different GNNs are affected by the bottleneck differently

- At the **t=4**, all messages reach the target node
- One of the branches is irrelevant, because the solution is in the other branch
- GIN and GCN **sum both** incoming edges
- GAT and GGNN can select to absorb a single edge.
  - ⇒ The effective bottleneck is only **half** of the effect on GIN and GCN
  - ⇒ Thus, they succeed in a depth that is larger by 1





Does the bottleneck affect real-life models?

- Off-the-shelf, state-of-the-art models, trained by others

Does the bottleneck affect real-life models?

- Off-the-shelf, state-of-the-art models, trained by others
- To break the bottleneck:
  - We (modified the original implementations and) made the last GNN layer fully-adjacent (**FA**) - every node has an edge to all other nodes

Does the bottleneck affect real-life models?

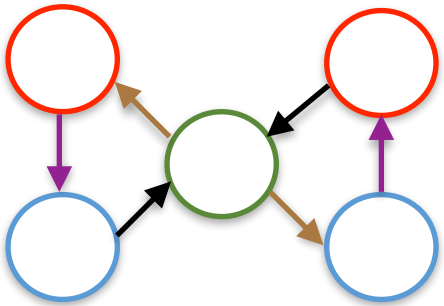
- Off-the-shelf, state-of-the-art models, trained by others
- To break the bottleneck:
  - We (modified the original implementations and) made the last GNN layer fully-adjacent (**FA**) - every node has an edge to all other nodes
  - Re-trained without adding weights, without **any** hyperparameter tuning

Does the bottleneck affect real-life models?

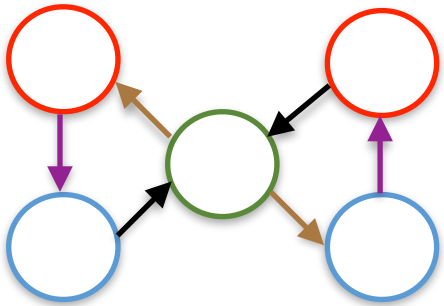
- Off-the-shelf, state-of-the-art models, trained by others
- To break the bottleneck:
  - We (modified the original implementations and) made the last GNN layer fully-adjacent (**FA**) - every node has an edge to all other nodes
  - Re-trained without adding weights, without **any** hyperparameter tuning
- The most trivial idea, just to show that the bottleneck affects SoTA models

Does the bottleneck affect real-life models?

- Off-the-shelf, state-of-the-art models, trained by others
- To break the bottleneck:
  - We (modified the original implementations and) made the last GNN layer fully-adjacent (**FA**) - every node has an edge to all other nodes
  - Re-trained without adding weights, without **any** hyperparameter tuning
- The most trivial idea, just to show that the bottleneck affects SoTA models



t=0

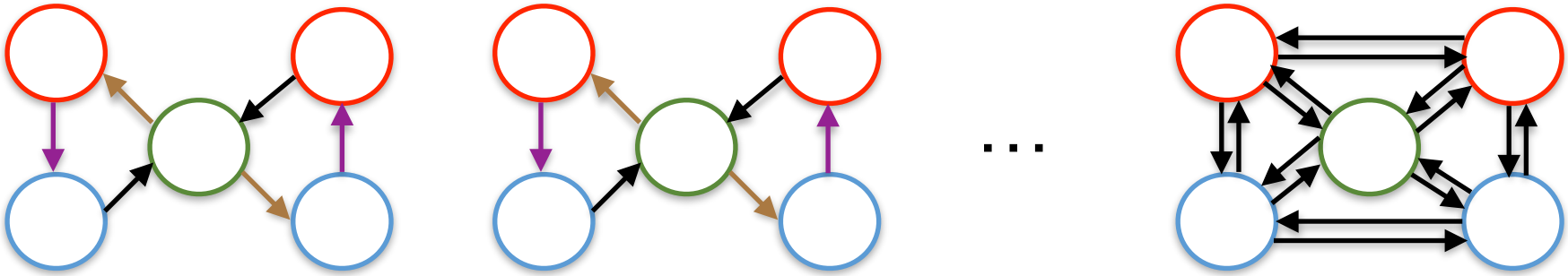


t=1

...

Does the bottleneck affect real-life models?

- Off-the-shelf, state-of-the-art models, trained by others
- To break the bottleneck:
  - We (modified the original implementations and) made the last GNN layer fully-adjacent (**FA**) - every node has an edge to all other nodes
  - Re-trained without adding weights, without **any** hyperparameter tuning
- The most trivial idea, just to show that the bottleneck affects SoTA models



Real-life Models

- A simple modification improves state-of-the-art GNNs without any tuning:

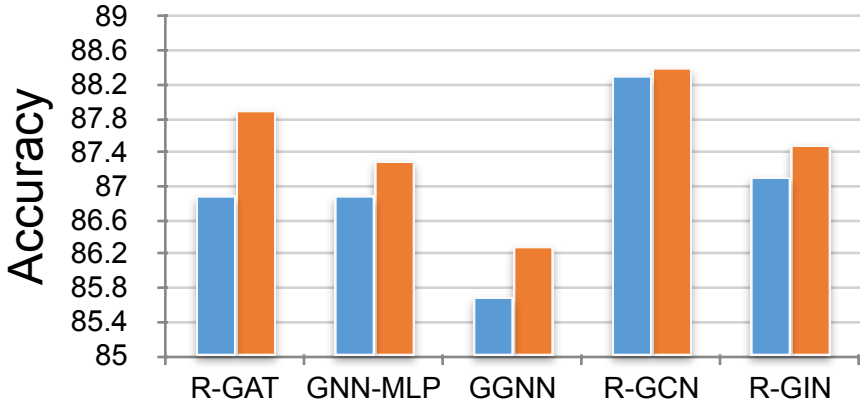
Real-life Models

- A simple modification improves state-of-the-art GNNs without any tuning:
  - +1% accuracy increase in Variable Misuse
  - -40% error reduction in predicting quantum chemical properties of molecules (“QM9”)
  - -5% error reduction in classifying whether a biochemical compound contains anti-cancer activity (“NCI1”)
  - -12% error reduction in classifying enzymes (“ENZYMES”)

VarMisuse
(accuracy, higher = better)

VarMisuse
(accuracy, higher = better)

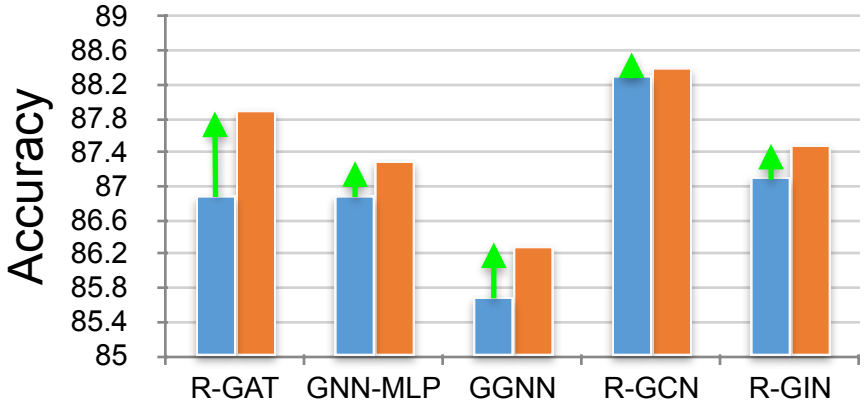
SeenProj test set



Base (Brockschmidt, ICML'2020)
+FA

VarMisuse
(accuracy, higher = better)

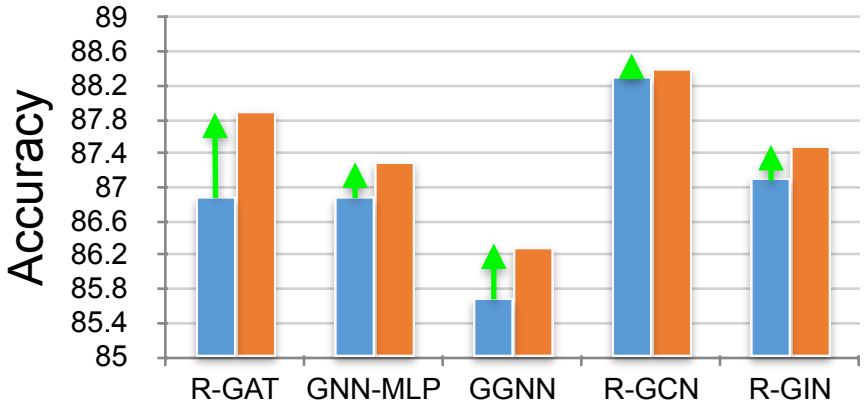
SeenProj test set



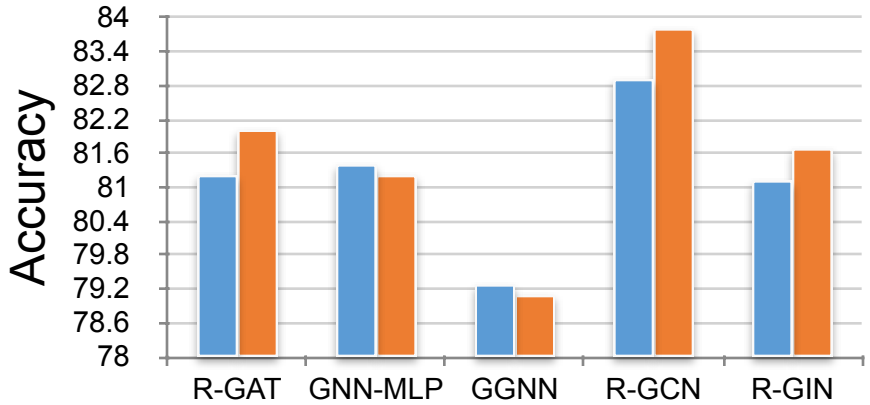
Base (Brockschmidt, ICML'2020)
+FA

VarMisuse
(accuracy, higher = better)

SeenProj test set



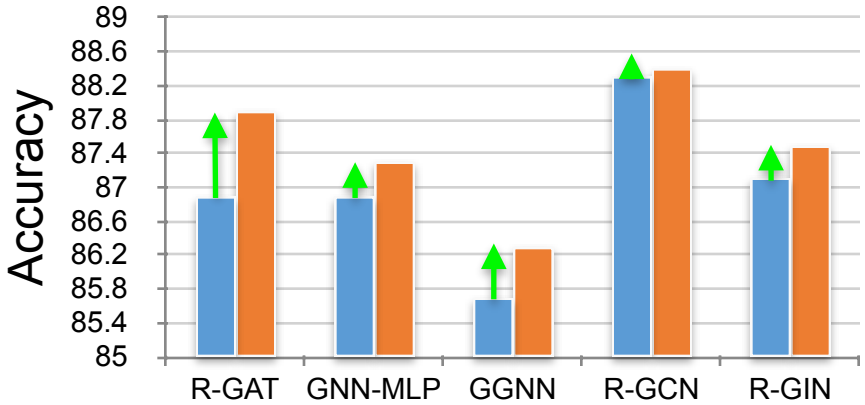
UnseenProj test set



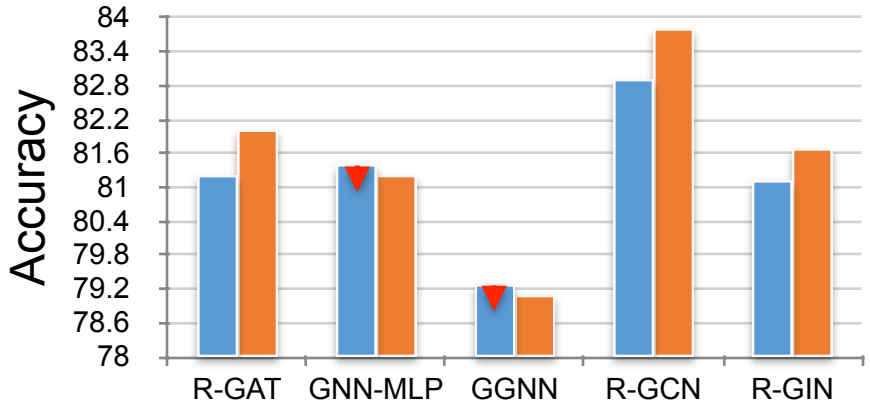
Base (Brockschmidt, ICML'2020)
+FA

VarMisuse
(accuracy, higher = better)

SeenProj test set



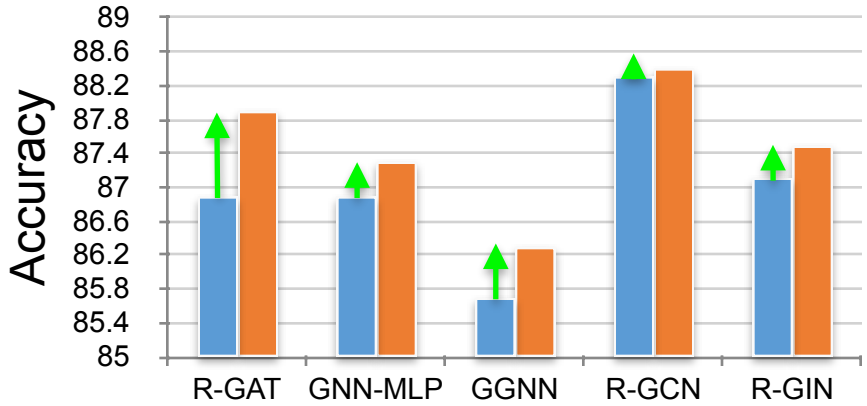
UnseenProj test set



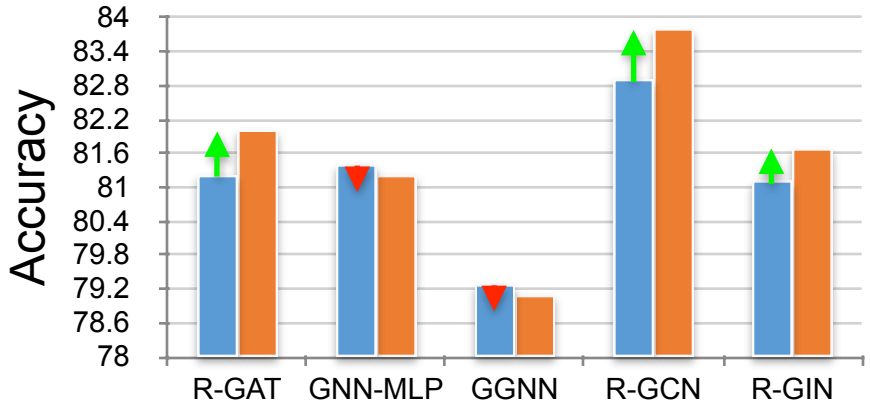
Base (Brockschmidt, ICML'2020)
+FA

VarMisuse
(accuracy, higher = better)

SeenProj test set



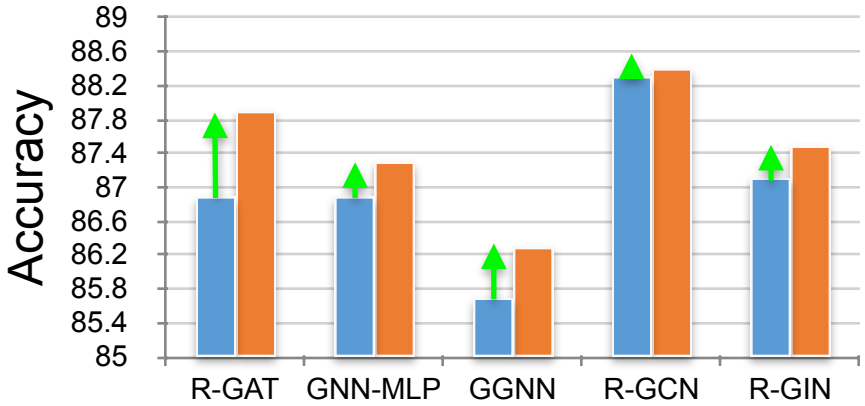
UnseenProj test set



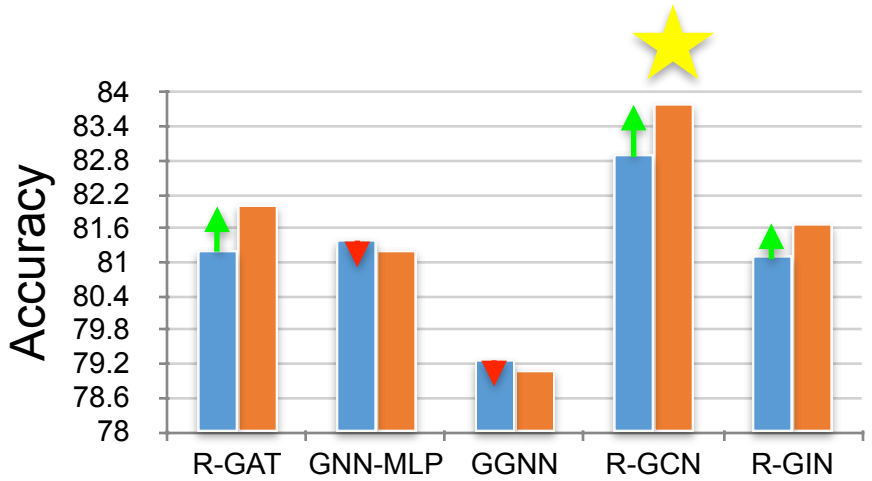
Base (Brockschmidt, ICML'2020)
+FA

VarMisuse
(accuracy, higher = better)

SeenProj test set



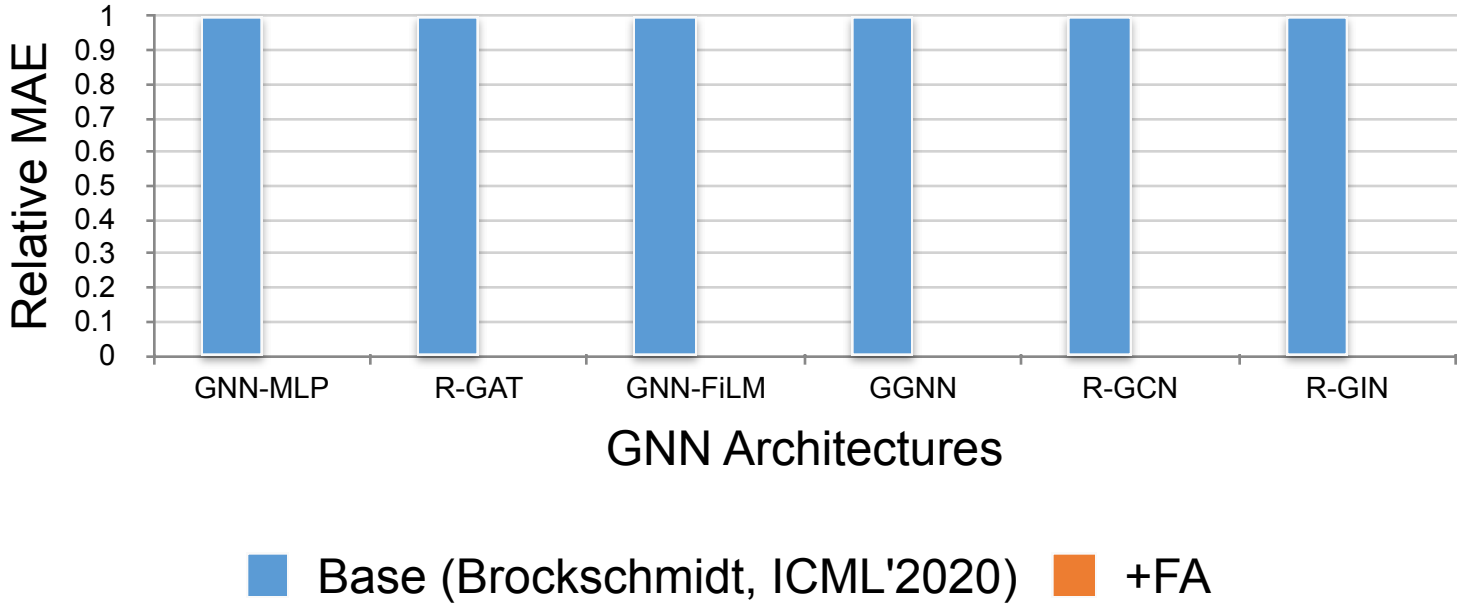
UnseenProj test set



Base (Brockschmidt, ICML'2020)
+FA

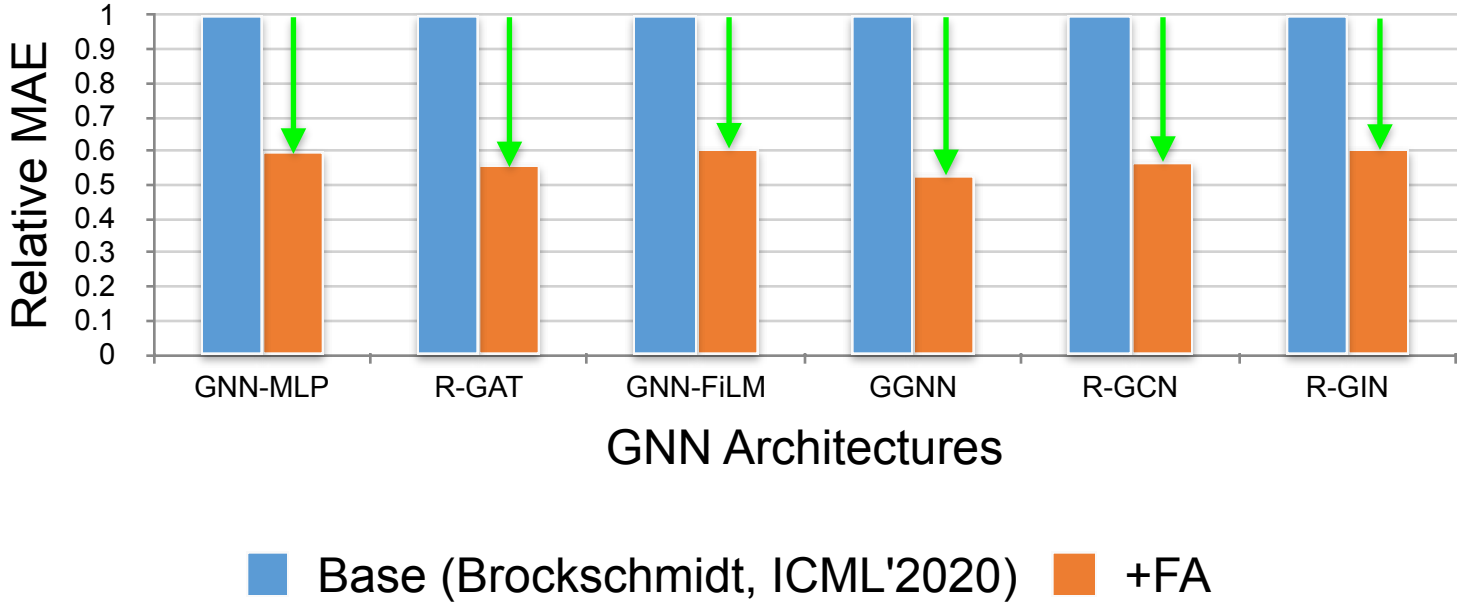
QM9 Dataset (molecules regression)

(mean absolute error, lower = better)

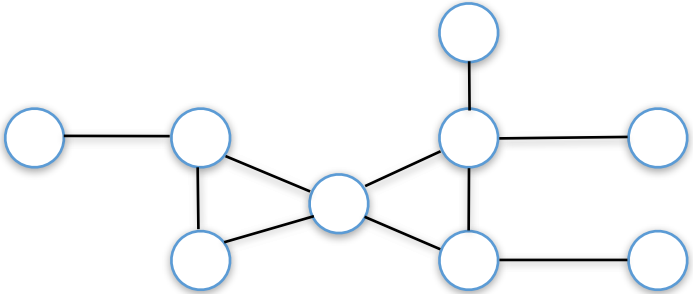


QM9 Dataset (molecules regression)

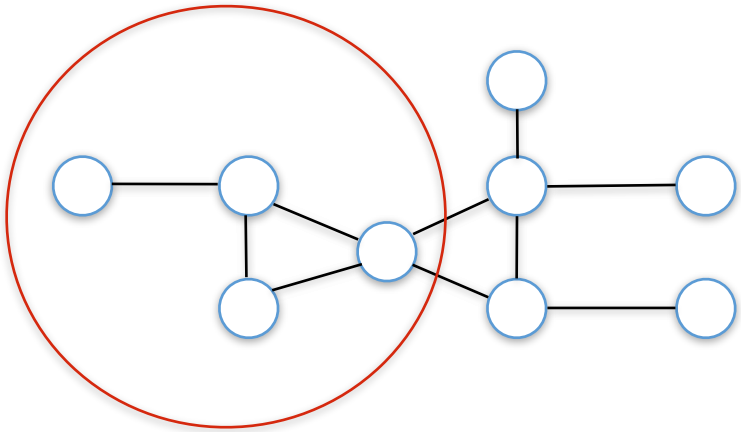
(mean absolute error, lower = better)



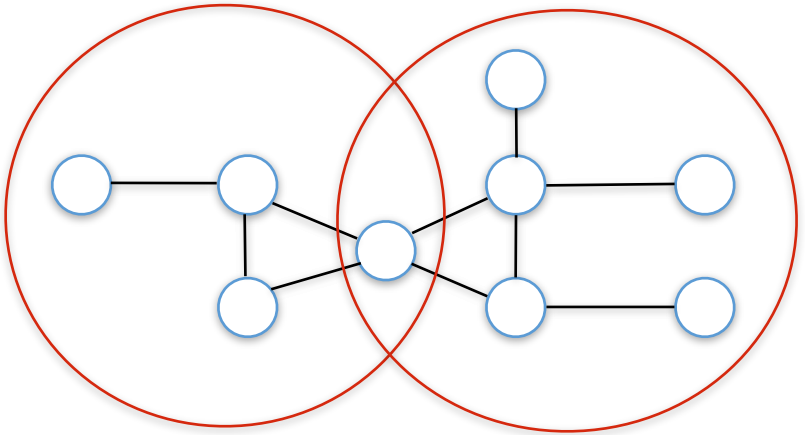
Maybe the number of layers simply wasn't enough?



Maybe the number of layers simply wasn't enough?

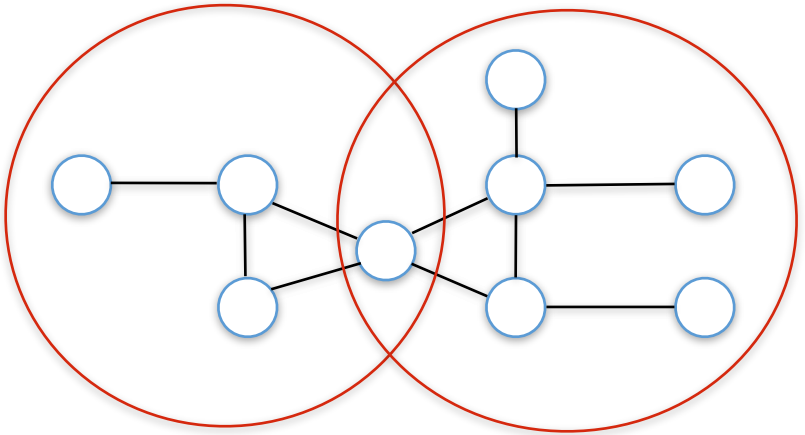


Maybe the number of layers simply wasn't enough?



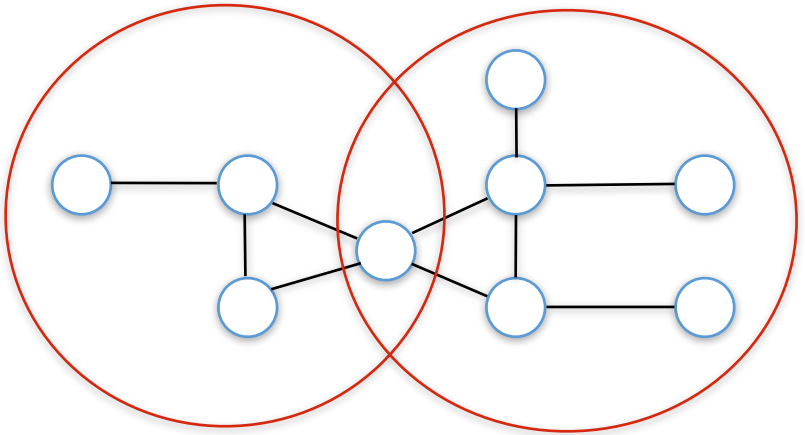
Maybe the number of layers simply wasn't enough?

- "Under-reaching": $K < diameter$



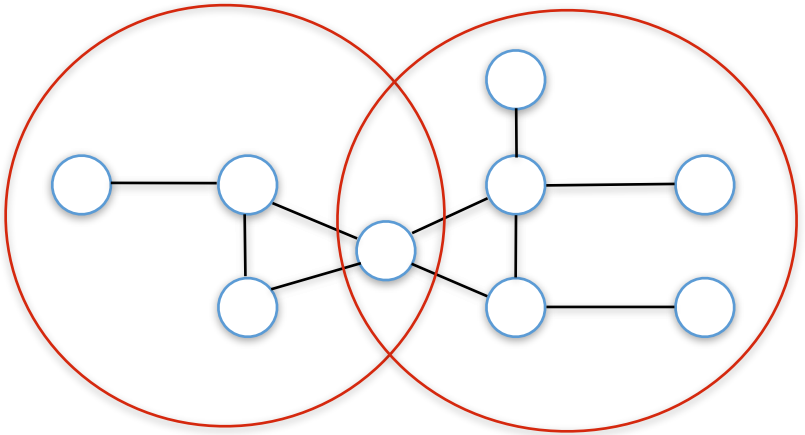
Maybe the number of layers simply wasn't enough?

- "Under-reaching": $K < diameter$
- In QM9:
  - $K = 8$



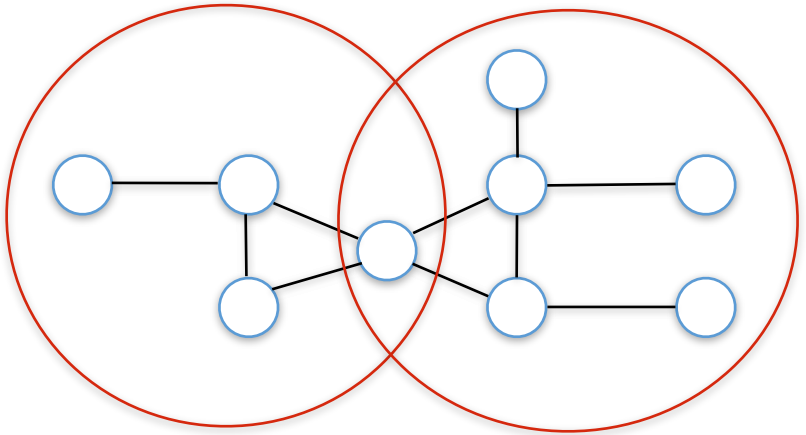
Maybe the number of layers simply wasn't enough?

- "Under-reaching": $K < diameter$
- In QM9:
  - $K = 8$
  - $diameter = 10$ (max)



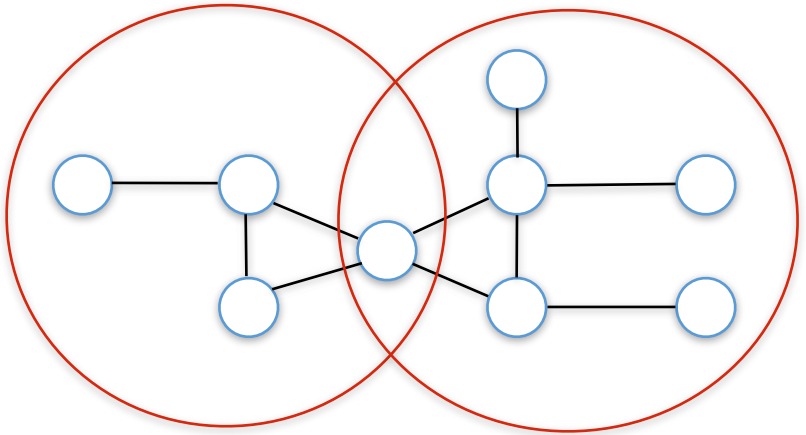
Maybe the number of layers simply wasn't enough?

- "Under-reaching": $K < diameter$
- In QM9:
  - $K = 8$
  - $diameter = 10$ (max)
  - $diameter = 6$ (average)



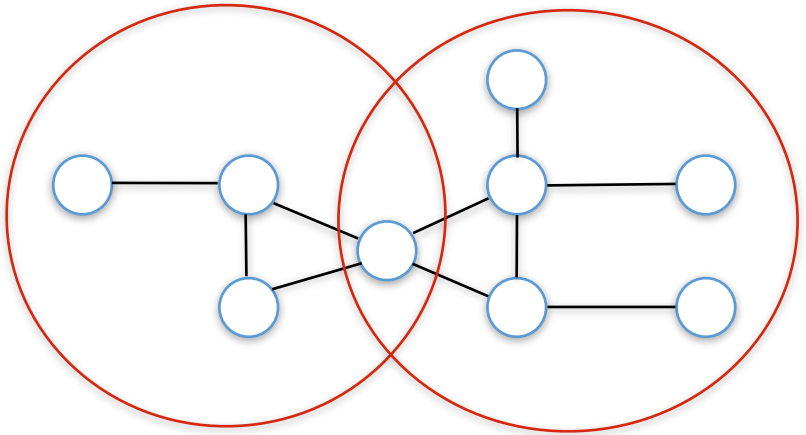
Maybe the number of layers simply wasn't enough?

- "Under-reaching": $K < diameter$
- In QM9:
  - $K = 8$
  - $diameter = 10$ (max)
  - $diameter = 6$ (average)
  - $diameter = 8$ (90'th percentile)



Maybe the number of layers simply wasn't enough?

- "Under-reaching": $K < diameter$
- In QM9:
  - $K = 8$
  - $diameter = 10$ (max)
  - $diameter = 6$ (average)
  - $diameter = 8$ (90'th percentile)
- Setting $K = 10$ didn't help



Maybe you can just increase d ?

Maybe you can just increase d ?

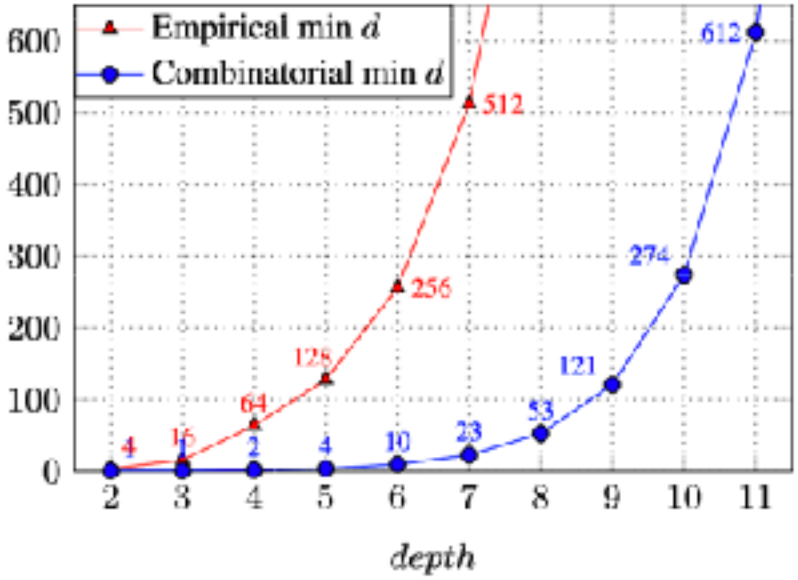
- 2x increase in the hidden size d $\longrightarrow$ $5\,\%$ error decrease

Maybe you can just increase d ?

- 2x increase in the hidden size d $\longrightarrow$ $5\,\%$ error decrease
  - (FA layer decreased the error by $40\,\%$, without added params)

Maybe you can just increase d ?

- 2x increase in the hidden size d $\longrightarrow$ $5\,\%$ error decrease
  - (FA layer decreased the error by $40\,\%$, without added params)



Breaking the Bottleneck in Previous Work

Some previous work implicitly avoided over-squashing:

Breaking the Bottleneck in Previous Work

Some previous work implicitly avoided over-squashing:

“Supersource edges”

[Scarselli, ’2008]

Breaking the Bottleneck in Previous Work

Some previous work implicitly avoided over-squashing:

“Supersource edges”

[Scarselli, ’2008]

“Virtual edges”

[Gilmer, ICML’2017]

Breaking the Bottleneck in Previous Work

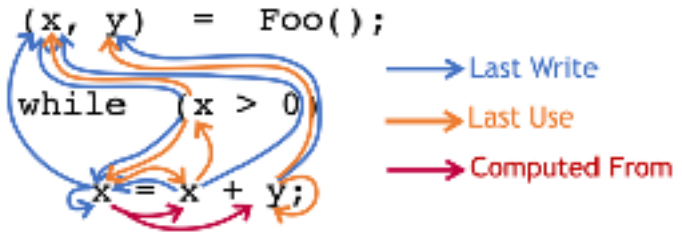
Some previous work implicitly avoided over-squashing:

“Supersource edges”

[Scarselli, ’2008]

“Virtual edges”

[Gilmer, ICML’2017]



[Allamanis, ICLR’2018] [Brockschmidt, ICLR’2019]

Slide from Marc Brockschmidt

Breaking the Bottleneck in Previous Work

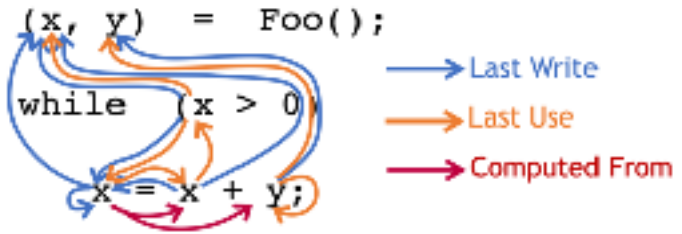
Some previous work implicitly avoided over-squashing:

“Supersource edges”

[Scarselli, ’2008]

“Virtual edges”

[Gilmer, ICML’2017]



[Allamanis, ICLR’2018] [Brockschmidt, ICLR’2019]

Slide from Marc Brockschmidt

Ad-hoc solutions, none of them explicitly highlighted the bottleneck, explained it, and measured its effects

Over-squashing vs. Over-smoothing

Over-squashing vs. Over-smoothing

r - the problem radius (the required range of interaction between nodes)

Over-squashing vs. Over-smoothing

r - the problem radius (the required range of interaction between nodes)

K - the number of GNN layers

Over-squashing vs. Over-smoothing

r - the problem radius (the required range of interaction between nodes)

K - the number of GNN layers

Over-smoothing

$K >> r$

Over-squashing vs. Over-smoothing

r - the problem radius (the required range of interaction between nodes)

K - the number of GNN layers

Over-smoothing

$K >> r$

$r = 1$

$K >> 1$

Over-squashing vs. Over-smoothing

r - the problem radius (the required range of interaction between nodes)

K - the number of GNN layers

Over-smoothing

$$K >> r$$

$$r = 1$$

$$K >> 1$$

Nodes will go indistinguishable

Over-squashing vs. Over-smoothing

r - the problem radius (the required range of interaction between nodes)

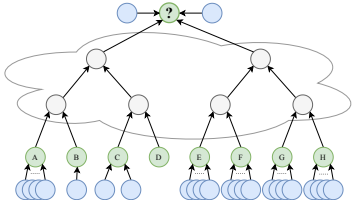
K - the number of GNN layers

Over-smoothing $K >> r$	Under-reaching $K < r$
$r = 1$ $K >> 1$ 	
Nodes will go indistinguishable	

Over-squashing vs. Over-smoothing

r - the problem radius (the required range of interaction between nodes)

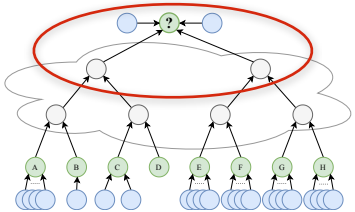
K - the number of GNN layers

Over-smoothing $K >> r$	Under-reaching $K < r$
$r = 1$ $K >> 1$ 	$r = 4$ $K < 4$ 
Nodes will go indistinguishable	

Over-squashing vs. Over-smoothing

r - the problem radius (the required range of interaction between nodes)

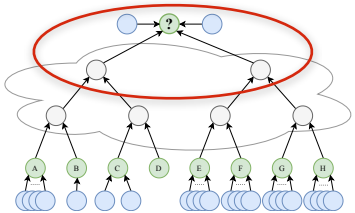
K - the number of GNN layers

Over-smoothing $K >> r$	Under-reaching $K < r$
$r = 1$ $K >> 1$ 	$r = 4$ $K < 4$ 
Nodes will go indistinguishable	

Over-squashing vs. Over-smoothing

r - the problem radius (the required range of interaction between nodes)

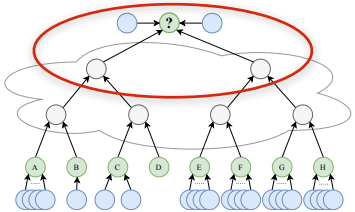
K - the number of GNN layers

Over-smoothing $K >> r$	Under-reaching $K < r$
$r = 1$ $K >> 1$ 	$r = 4$ $K < 4$ 
Nodes will go indistinguishable	The GNN cannot fit long-range patterns

Over-squashing vs. Over-smoothing

r - the problem radius (the required range of interaction between nodes)

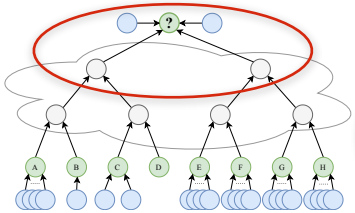
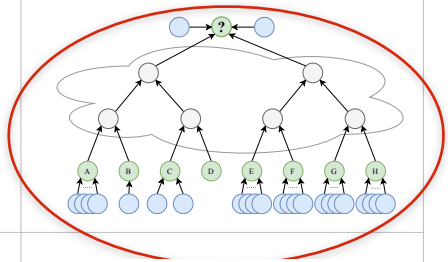
K - the number of GNN layers

Over-smoothing $K >> r$	Under-reaching $K < r$	Over-squashing $K = r >> 1$
$r = 1$ $K >> 1$ 	$r = 4$ $K < 4$ 	
Nodes will go indistinguishable	The GNN cannot fit long-range patterns	

Over-squashing vs. Over-smoothing

r - the problem radius (the required range of interaction between nodes)

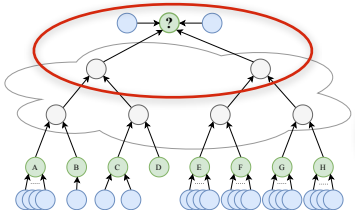
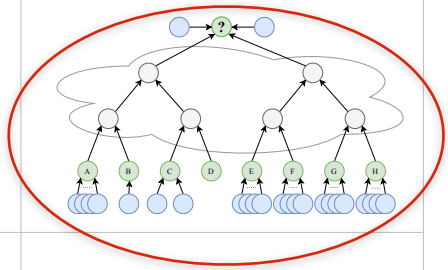
K - the number of GNN layers

Over-smoothing $K >> r$	Under-reaching $K < r$	Over-squashing $K = r >> 1$
$r = 1$ $K >> 1$ 	$r = 4$ $K < 4$ 	$r = 4$ $K = 4$ 
Nodes will go indistinguishable	The GNN cannot fit long-range patterns	

Over-squashing vs. Over-smoothing

r - the problem radius (the required range of interaction between nodes)

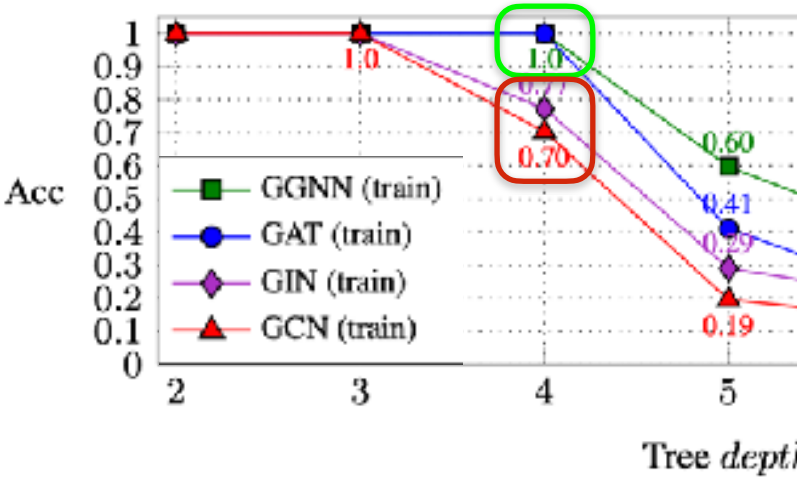
K - the number of GNN layers

Over-smoothing $K >> r$	Under-reaching $K < r$	Over-squashing $K = r >> 1$
$r = 1$ $K >> 1$ 	$r = 4$ $K < 4$ 	$r = 4$ $K = 4$ 
Nodes will go indistinguishable	The GNN cannot fit long-range patterns	The GNN *fails to* fit long-range patterns

Different GNNs are affected by the bottleneck differently

- **GCN** and **GIN** suffer from over-squashing **more** than **GAT** and **GGNN**.

- **GCN** $\mathbf{h}_v^{(k)} = ReLU\left(W^{(k)}\left(\frac{1}{c_v}\mathbf{h}_v^{(k-1)} + \sum_{u\in\mathcal{N}_v}\frac{1}{c_{v,u}}\mathbf{h}_u^{(k-1)}\right)\right)$
- **GIN** $\mathbf{h}_v^{(k)} = MLP^{(k)}\left(\left(1+\epsilon^{(k)}\right)\mathbf{h}_v^{(k-1)} + \sum_{u\in\mathcal{N}_v}\mathbf{h}_u^{(k-1)}\right)$
- **GGNN** $\mathbf{h}_v^{(k)} = GRU\left(\mathbf{h}_v^{(k-1)}, \sum_{u\in\mathcal{N}_v} W_{neighbor}\mathbf{h}_u^{(k-1)}\right)$
- **GAT** $\mathbf{h}_v^{(k)} = ReLU\left(MultiHeadAttention\left(\mathcal{N}_v \mid \mathbf{h}_v^{(k-1)}\right)\right)$



- **GAT**

$$\mathbf{h}_v^{(k)} = ReLU\left(MultiHeadAttention\left(\mathcal{N}_v \mid \mathbf{h}_v^{(k-1)}\right)\right)$$

- **GAT** $\mathbf{h}_v^{(k)} = ReLU\left(MultiHeadAttention\left(\mathcal{N}_v \mid \mathbf{h}_v^{(k-1)}\right)\right)$

How Attentive are Graph Neural Networks?



Shaked Brody **Uri Alon** Eran Yahav

- GAT

$$\mathbf{h}_v^{(k)} = ReLU\left(MultiHeadAttention\left(\mathcal{N}_v \mid \mathbf{h}_v^{(k-1)}\right)\right)$$

How Attentive are Graph Neural Networks?



Shaked Brody **Uri Alon** Eran Yahav

q0
q1
q2
q3
q4
q5
q6
q7
q8
q9

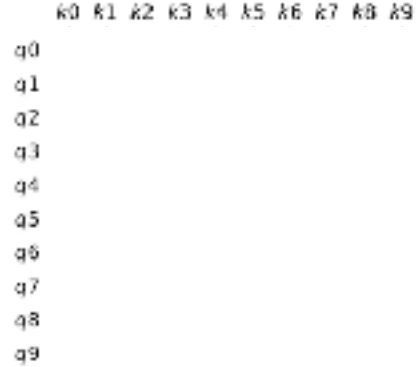
- **GAT**

$$\mathbf{h}_v^{(k)} = ReLU\left(MultiHeadAttention\left(\mathcal{N}_v \mid \mathbf{h}_v^{(k-1)}\right)\right)$$

How Attentive are Graph Neural Networks?

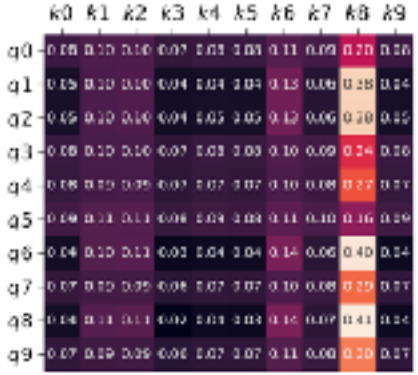


Shaked Brody **Uri Alon** Eran Yahav



- **GAT** $\mathbf{h}_v^{(k)} = ReLU\left(MultiHeadAttention\left(\mathcal{N}_v \mid \mathbf{h}_v^{(k-1)}\right)\right)$

How Attentive are Graph Neural Networks?



(a) Attention in standard GAT (Veličković et al. (2018))

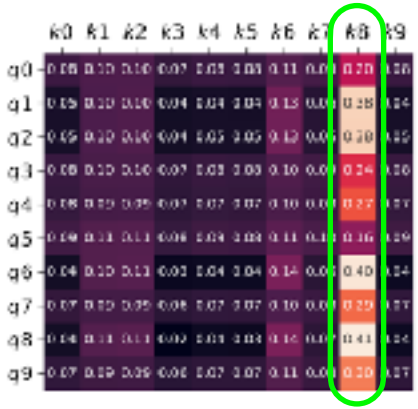


Shaked Brody **Uri Alon** Eran Yahav

- **GAT**

$$\mathbf{h}_v^{(k)} = ReLU\left(MultiHeadAttention\left(\mathcal{N}_v \mid \mathbf{h}_v^{(k-1)}\right)\right)$$

How Attentive are Graph Neural Networks?



(a) Attention in standard GAT (Veličković et al. (2018))

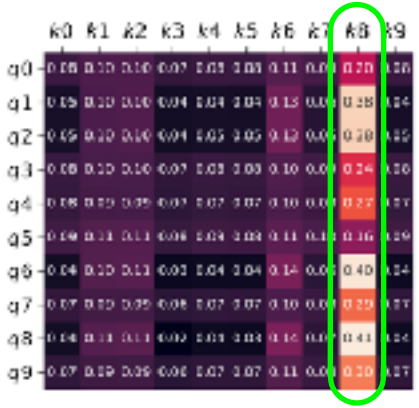


Shaked Brody **Uri Alon** Eran Yahav

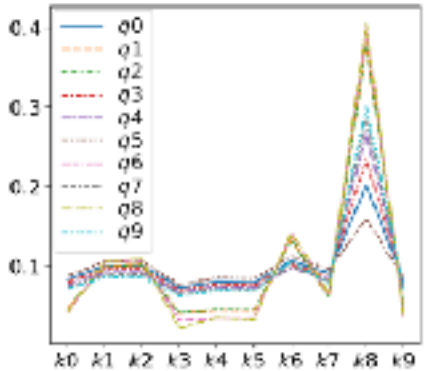
- GAT

$$\mathbf{h}_v^{(k)} = ReLU\left(MultiHeadAttention\left(\mathcal{N}_v \mid \mathbf{h}_v^{(k-1)}\right)\right)$$

How Attentive are Graph Neural Networks?



(a) Attention in standard GAT (Veličković et al. (2018))



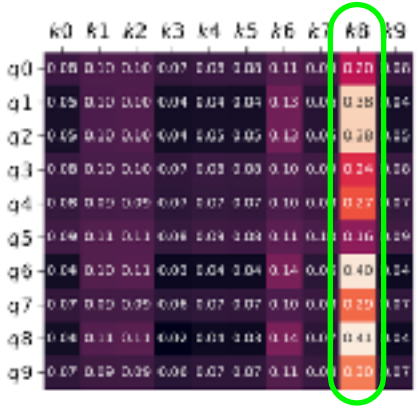


Shaked Brody **Uri Alon** Eran Yahav

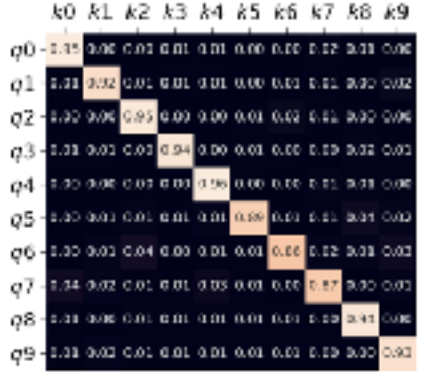
- **GAT**

$$\mathbf{h}_v^{(k)} = ReLU\left(MultiHeadAttention\left(\mathcal{N}_v \mid \mathbf{h}_v^{(k-1)}\right)\right)$$

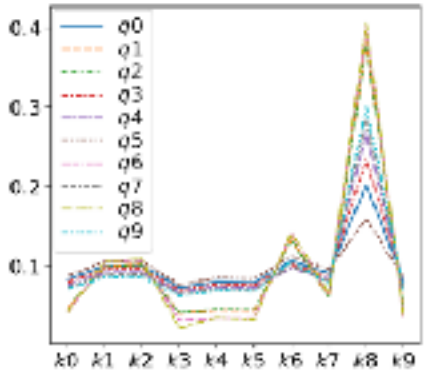
How Attentive are Graph Neural Networks?

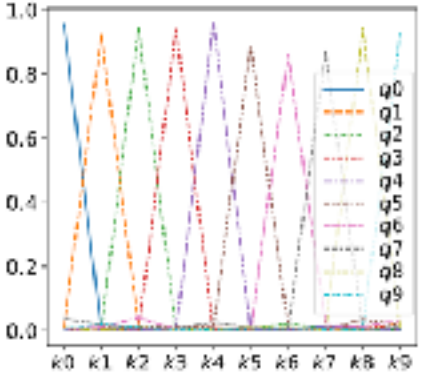


(a) Attention in standard GAT (Veličković et al. (2018))



(b) Attention in GATv2, our fixed version of GAT







Shaked Brody **Uri Alon** Eran Yahav

How Attentive are Graph Neural Networks?



Shaked Brody **Uri Alon** Eran Yahav

How Attentive are Graph Neural Networks?



Shaked Brody **Uri Alon** Eran Yahav

Static attention (GAT, Veličković et al., 2018):

$$e\left(h_i, h_j\right) = LeakyReLU\left(a^{\top} \cdot \left[Wh_i \| Wh_j\right]\right)$$

How Attentive are Graph Neural Networks?



Shaked Brody **Uri Alon** Eran Yahav

Static attention (GAT, Veličković et al., 2018):

$$e\left(h_i, h_j\right) = LeakyReLU\left(a^{\top} \cdot \left[Wh_i \| Wh_j\right]\right)$$

How Attentive are Graph Neural Networks?



Shaked Brody **Uri Alon** Eran Yahav

Static attention (GAT, Veličković et al., 2018):

$$e\left(h_i, h_j\right) = LeakyReLU\left(a^{\top} \cdot \left[Wh_i \| Wh_j\right]\right)$$

How Attentive are Graph Neural Networks?



Shaked Brody **Uri Alon** Eran Yahav

Static attention (GAT, Veličković et al., 2018):

$$e\left(h_i, h_j\right) = LeakyReLU\left(a^\top \cdot \left[Wh_i \| Wh_j\right]\right)$$

$$a^\top$$

How Attentive are Graph Neural Networks?



Shaked Brody **Uri Alon** Eran Yahav

Static attention (GAT, Veličković et al., 2018):

$$e\left(h_i, h_j\right) = LeakyReLU\left(a^{\top} \cdot \left[Wh_i \| Wh_j\right]\right)$$

$$a^{\top} LeakyReLU$$

How Attentive are Graph Neural Networks?



Shaked Brody **Uri Alon** Eran Yahav

Static attention (GAT, Veličković et al., 2018):

$$e\left(h_i, h_j\right) = LeakyReLU\left(a^{\top} \cdot \left[Wh_i \| Wh_j\right]\right)$$

$$a^{\top} LeakyReLU \quad W$$

How Attentive are Graph Neural Networks?



Shaked Brody **Uri Alon** Eran Yahav

Static attention (GAT, Veličković et al., 2018):

$$e\left(h_i, h_j\right) = LeakyReLU\left(a^{\top} \cdot \left[Wh_i \| Wh_j\right]\right)$$

$$a^{\top} LeakyReLU \quad W \cdot \left[h_i \| h_j\right]$$

How Attentive are Graph Neural Networks?



Shaked Brody **Uri Alon** Eran Yahav

Static attention (GAT, Veličković et al., 2018):

$$e\left(h_i, h_j\right) = LeakyReLU\left(a^{\top} \cdot \left[Wh_i \| Wh_j\right]\right)$$

Dynamic attention (GATv2, this work):

$$e\left(h_i, h_j\right) = a^{\top} LeakyReLU\left(W \cdot \left[h_i \| h_j\right]\right)$$

How Attentive are Graph Neural Networks?



Shaked Brody **Uri Alon** Eran Yahav

Static attention (GAT, Veličković et al., 2018):

$$e\left(h_i, h_j\right) = LeakyReLU\left(a^{\top} \cdot \left[W h_i \| W h_j\right]\right)$$

Dynamic attention (GATv2, this work):

$$e\left(h_i, h_j\right) = a^{\top} LeakyReLU\left(W \cdot \left[h_i \| h_j\right]\right)$$

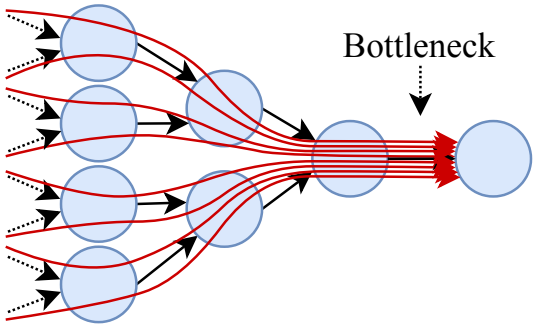
- Theoretically **much** more expressive
- Practically better across **11** benchmarks

Available in PyTorch Geometric:

from torch_geometric.nn.conv.gatv2_conv **import** GATv2Conv

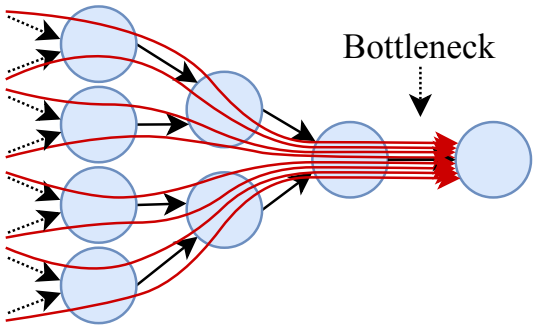
Summary

- To pass long-range messages - we need many GNN layers



Summary

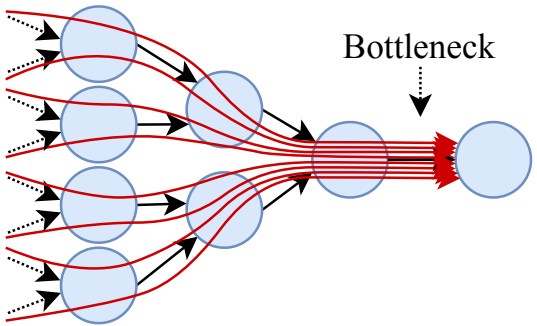
- To pass long-range messages - we need many GNN layers
- $|\text{Receptive Field}| = \mathcal{O}\left(exp\,(\text{layers})\right)$



Summary

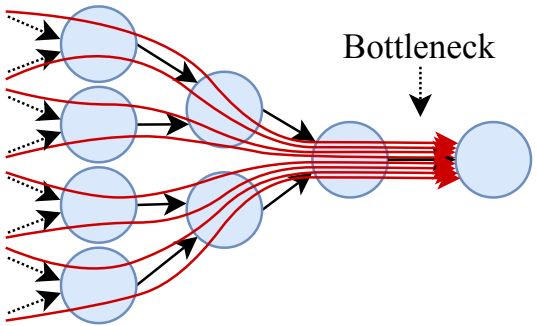
- To pass long-range messages - we need many GNN layers
- $|\text{Receptive Field}| = \mathcal{O}\left(exp\,(\text{layers})\right)$

  ➡ Leads to a **bottleneck** and **over-squashing**



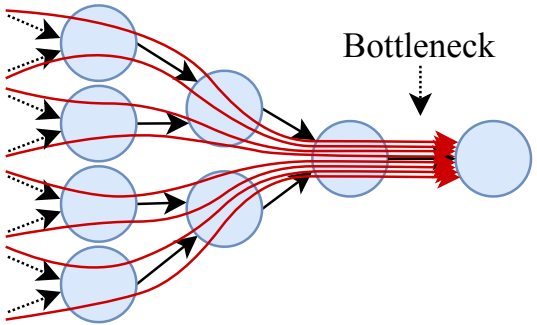
Summary

- To pass long-range messages - we need many GNN layers
- $|\text{Receptive Field}| = \mathcal{O}\left(exp\,(\text{layers})\right)$
  - ➡ Leads to a **bottleneck** and **over-squashing**
  - ➡ **Prevents** GNNs from fitting long-range patterns in the data



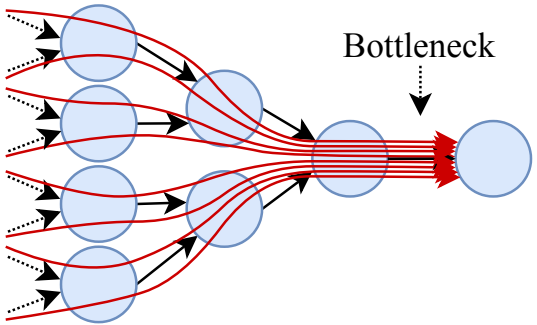
Summary

- To pass long-range messages - we need many GNN layers
- $|\text{Receptive Field}| = \mathcal{O}\left(exp\,(\text{layers})\right)$
  - ➡ Leads to a **bottleneck** and **over-squashing**
  - ➡ **Prevents** GNNs from fitting long-range patterns in the data
- **GCN** and **GIN** suffer from over-squashing **more** than others



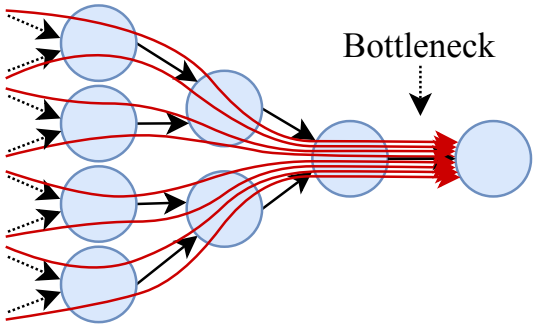
Summary

- To pass long-range messages - we need many GNN layers
- $|\text{Receptive Field}| = \mathcal{O}\left(exp\,(\text{layers})\right)$
  - ➡ Leads to a **bottleneck** and **over-squashing**
  - ➡ **Prevents** GNNs from fitting long-range patterns in the data
- **GCN** and **GIN** suffer from over-squashing **more** than others
- SoTA models can be **improved** by considering the bottleneck



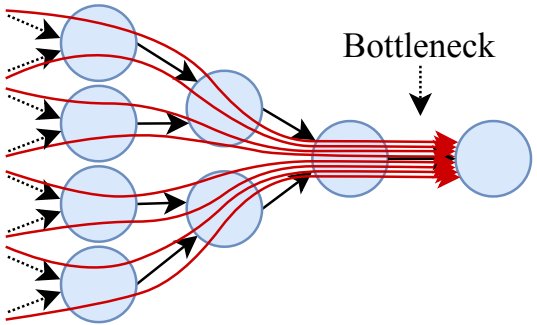
Summary

- To pass long-range messages - we need many GNN layers
- $|\text{Receptive Field}| = \mathcal{O}\left(exp\,(\text{layers})\right)$
  - ➡ Leads to a **bottleneck** and **over-squashing**
  - ➡ **Prevents** GNNs from fitting long-range patterns in the data
- **GCN** and **GIN** suffer from over-squashing **more** than others
- SoTA models can be **improved** by considering the bottleneck
- Use GATv2 instead of GAT



Summary

- To pass long-range messages - we need many GNN layers
- $|\text{Receptive Field}| = \mathcal{O}\left(exp\,(\text{layers})\right)$
  - ➡ Leads to a **bottleneck** and **over-squashing**
  - ➡ **Prevents** GNNs from fitting long-range patterns in the data
- **GCN** and **GIN** suffer from over-squashing **more** than others
- SoTA models can be **improved** by considering the bottleneck
- Use GATv2 instead of GAT



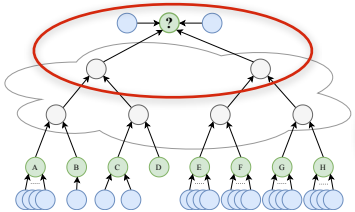
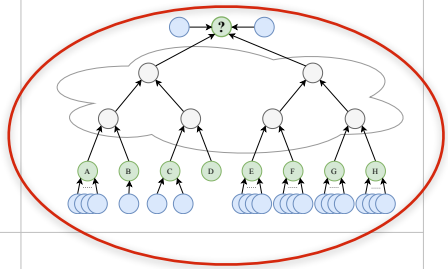
ualon@cs.cmu.edu

http://urialon.ml

Over-squashing vs. Over-smoothing

r - the problem radius (the required range of interaction between nodes)

K - the number of GNN layers

Over-smoothing $K >> r$	Under-reaching $K < r$	Over-squashing $K = r >> 1$
$r = 1$ $K >> 1$ 	$r = 4$ $K < 4$ 	$r = 4$ $K = 4$ 
Nodes will go indistinguishable	The GNN cannot fit long-range patterns	The GNN *fails to* fit long-range patterns