Graph Learning: Principles, Challenges, and Open Directions

Adrián Arnaiz-Rodríguez and Ameya Velingker

ellis ALICANTE unit

Google Research

ICML 2024
Vienna, Austria
22/07/2024

https://icml2024graphs.ameyavelingker.com/

- **Intro to GNNs and relevant concepts**
    - Early Graph embedding methods
    - GNNs
    - Spectral Graph Theory
- **Transformers**
- **Expressiveness**
- **Generalizability**
- **GNN Challenges**
    - Under-reaching
    - Over-smoothing
    - Over-squashing
- **Open questions**
- **Panel Discussion**

Graph Learning: Principles, Challenges, and Open Directions

ICML 2024 - 22/07/2024

Michael Bronstein
DeepMind Professor of
Artificial Intelligence

Michael Galkin
AI Research Scientist
at Intel Labs

Christopher Morris
Assistant Professor
at RWTH Aachen
University

Bryan Perozzi
Research Scientist at
Google Research

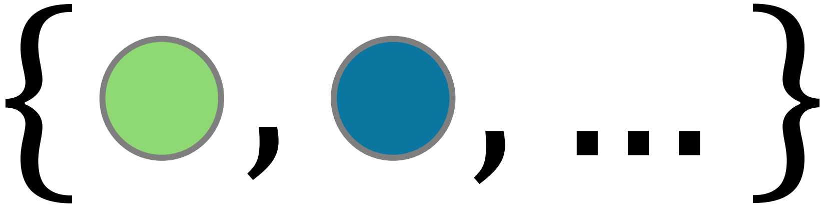
Panel Discussion

Open questions and challenges on GNNs
Graphs + LLMs
Graph Foundation Models

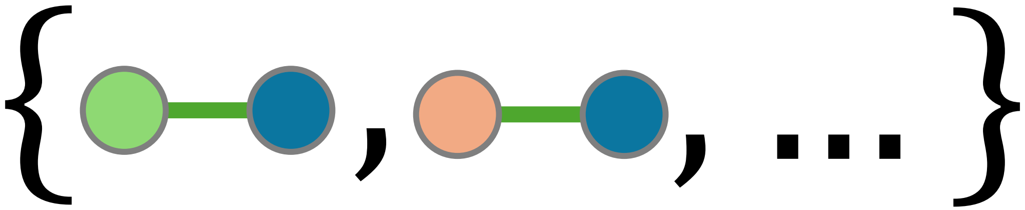
INTRODUCTION

Graphs

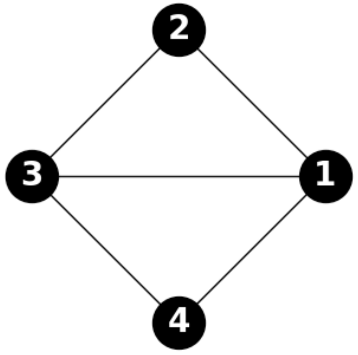
- **Set of Nodes =** V
  - Optionally with **features** X



- **Set of Edges =** E

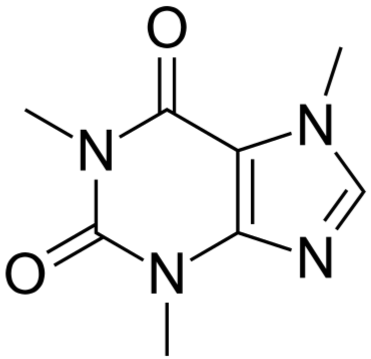


- **Adjacency Matrix =** A

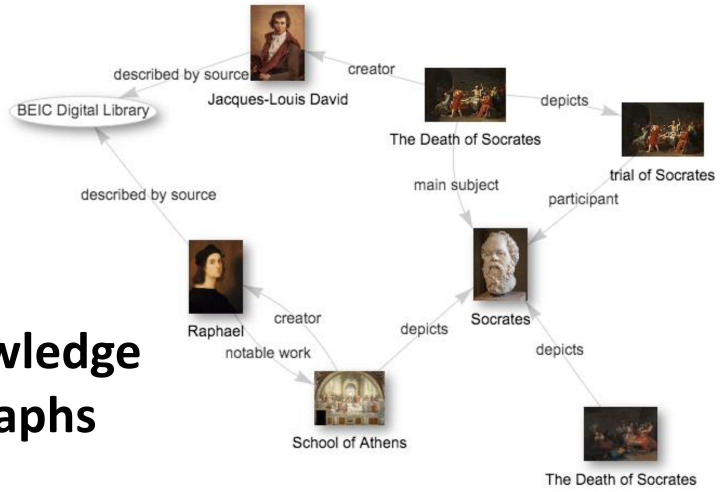


	1	2	3	4
1	0	1	1	1
2	1	0	1	0
3	1	1	0	1
4	1	0	1	0

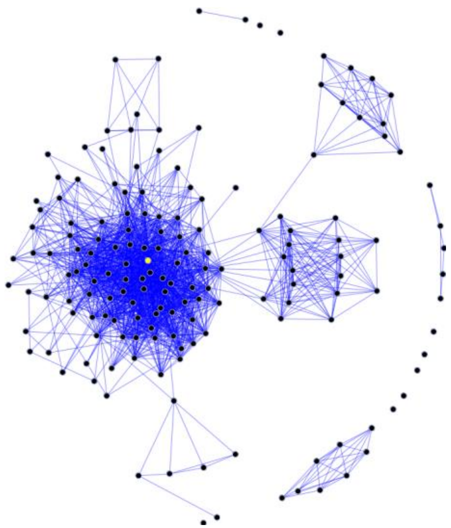
Graph Learning



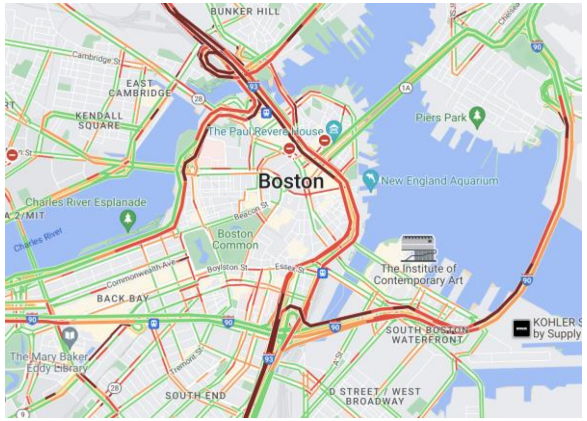
Molecular Graphs



Knowledge Graphs



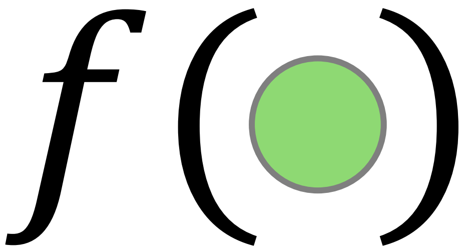
Social Network Graphs



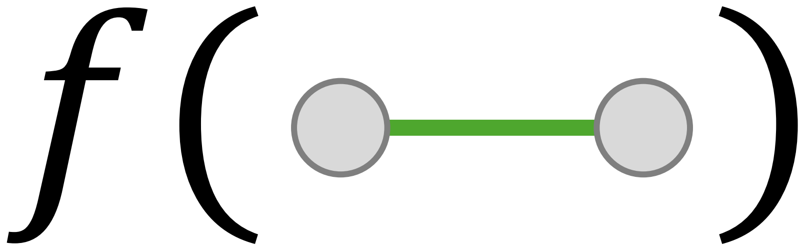
Road Network Graphs

Types of Tasks

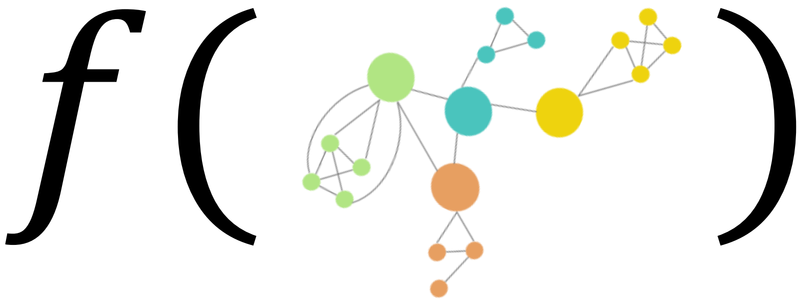
- **Node-level tasks**
  - Node classification
  - Node clustering
  - Node regression



- **Edge-level tasks**
  - Link prediction
  - Edge classification
  - Knowledge graph completion



- **Graph-level tasks**
  - Graph classification
  - Graph regression



Challenges of Machine Learning on Graphs

- Much of deep learning is on sequence or grid data
  - Transformers on sequences of tokens
  - Convolutional neural networks (CNNs) on pixel grids

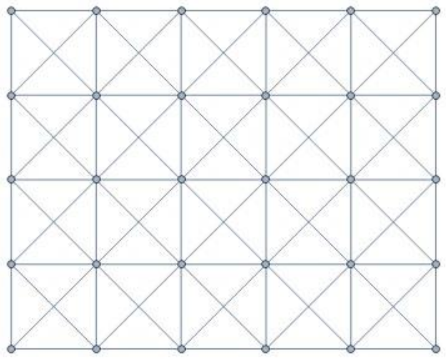
- Graphs have more general **topological structures**

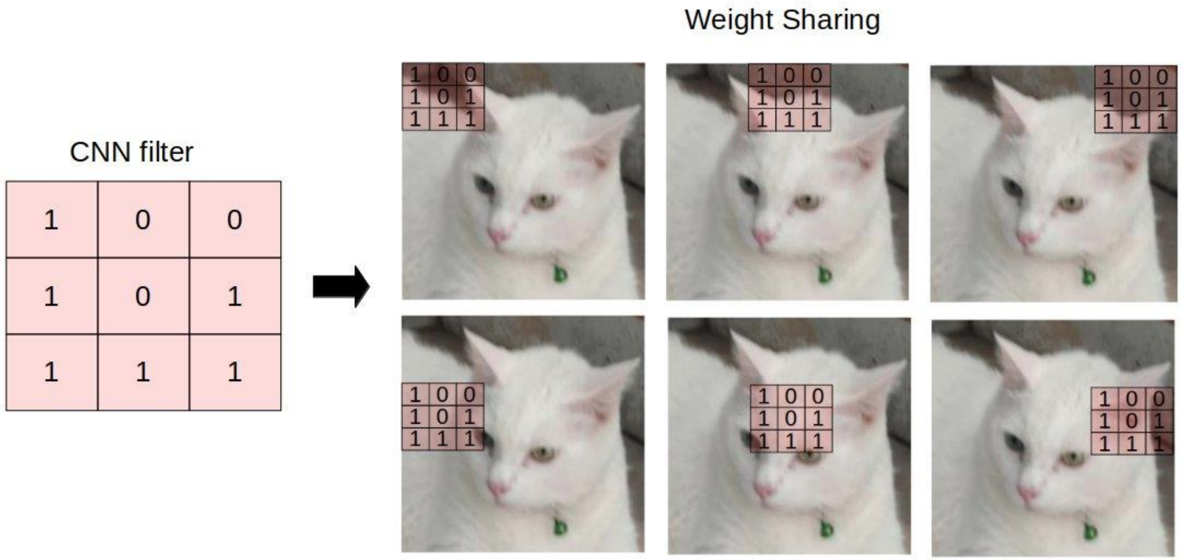
- Local neighborhoods vary in structure

- How to identify or order nodes within the graph?

Convolutional Networks

- Weight sharing
- Filters capture neighborhood on a grid graph





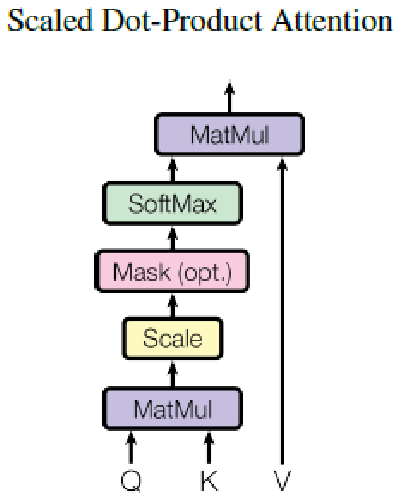
1	0	0
1	0	1
1	1	1

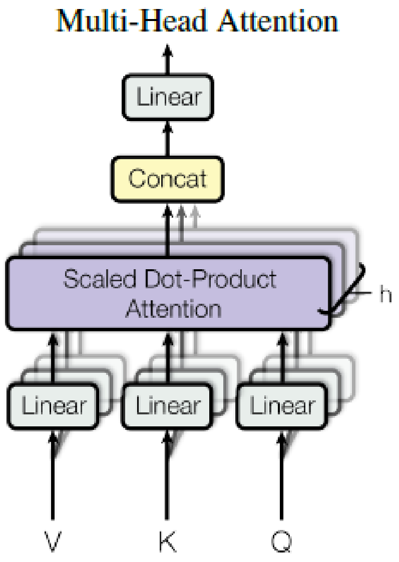
- ResNet, VGGNet, etc.

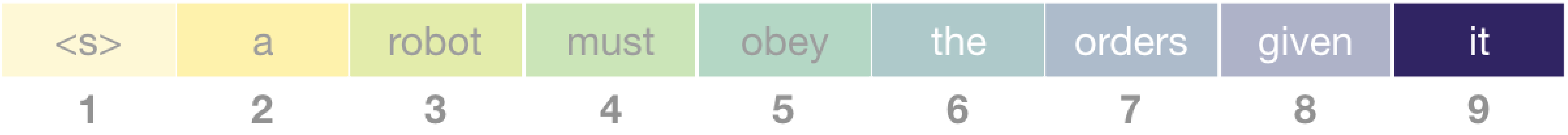
Image from https://towardsdatascience.com/understanding-graph-convolutional-networks-for-node-classification-a2bfdb7aba7b

Transformers

- Sequences of tokens
- Next token prediction based on a past context window







<s>	a	robot	must	obey	the	orders	given	it
1	2	3	4	5	6	7	8	9

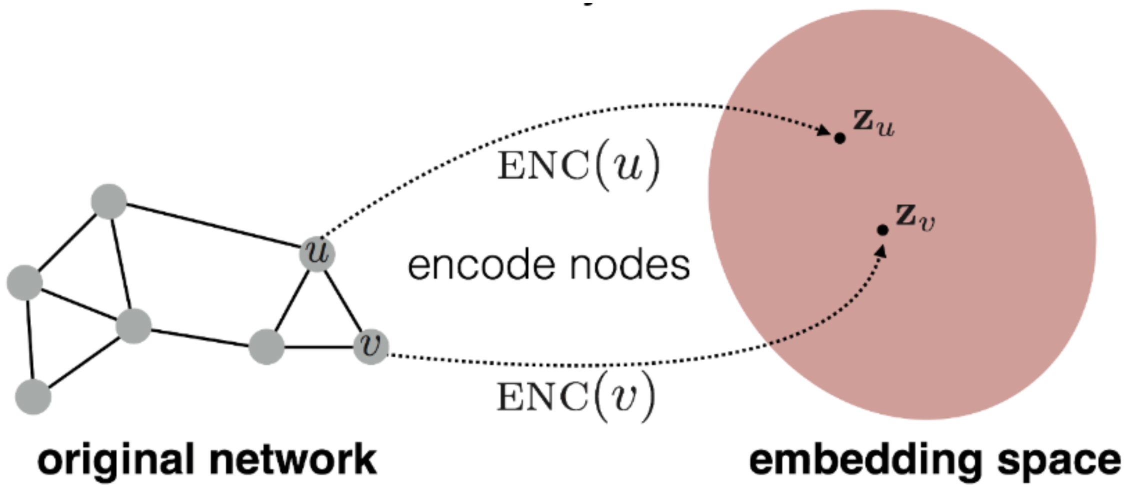
- Line graph

Image from https://towardsdatascience.com/understanding-graph-convolutional-networks-for-node-classification-a2bfdb7aba7b

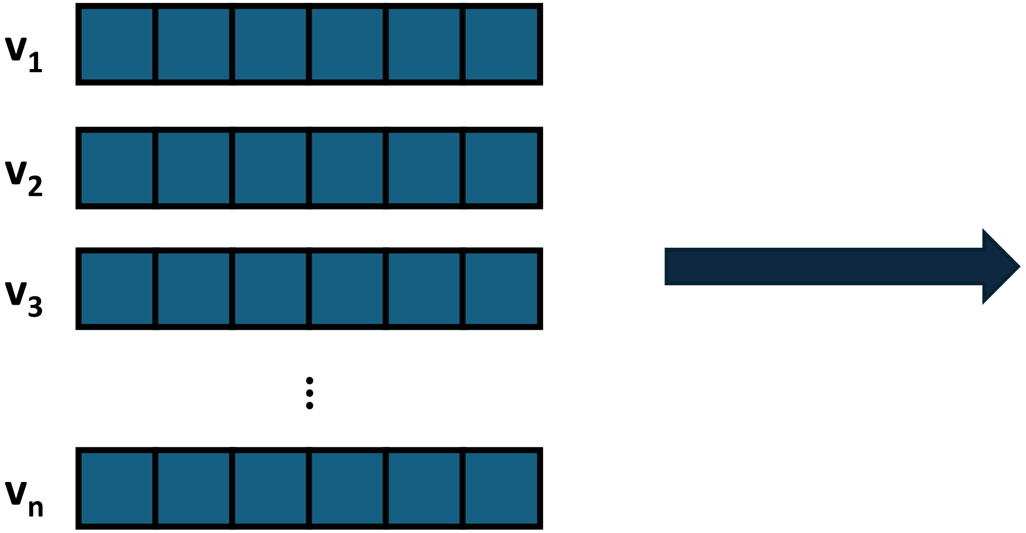
EARLY METHODS

Early Methods: Node Embeddings and Graph Kernels

- Map nodes into a (low-dimensional) embedding space
  - Similar nodes should have similar embeddings
- Methods
  - DeepWalk ([Perozzi et al., 2014])
  - Node2vec ([Grover and Leskovec, 2016])
- Techniques based on random walks, matrix factorization
- Graph kernels: map graphs to embeddings



Node Embeddings for Downstream Tasks



- Graph clustering
- Link prediction
- Graph classification
- Node classification
- Node regression
- Anomaly detection

Similarity in Node Embeddings

- **Similarity** of two nodes given by embeddings: $\langle \boldsymbol{z}^{u}, \boldsymbol{z}^{v} \rangle$
- Embeddings should maximize $\langle \boldsymbol{z}^{u}, \boldsymbol{z}^{v} \rangle$ for those pairs (u, v) that are similar
- How to decide whether u, v are similar?
  - Supervised approach: learn node embeddings based on tasks, labels
  - Unsupervised approach: learn node embeddings according to some structural aspects of the network

Random Walks for Node Embeddings

- **Idea**: Similarity of two nodes determined by whether they occur together in a random walk
- Random walks capture local information as well as some multi-hop information
- Filters out pairs of nodes that don't occur together on random walks (efficiency)

Random Walks for Node Embeddings

- Collect random walk statistics according to some random walk strategy
  - Run short fixed-length random walks starting from different nodes
  - Collect statistics on which nodes appear on random walks starting from each node
- Optimize the embeddings according to random walk stats
  - Define loss (e.g., based on maximum likelihood)
  - Stochastic gradient descent (SGD)

Random Walk Strategies

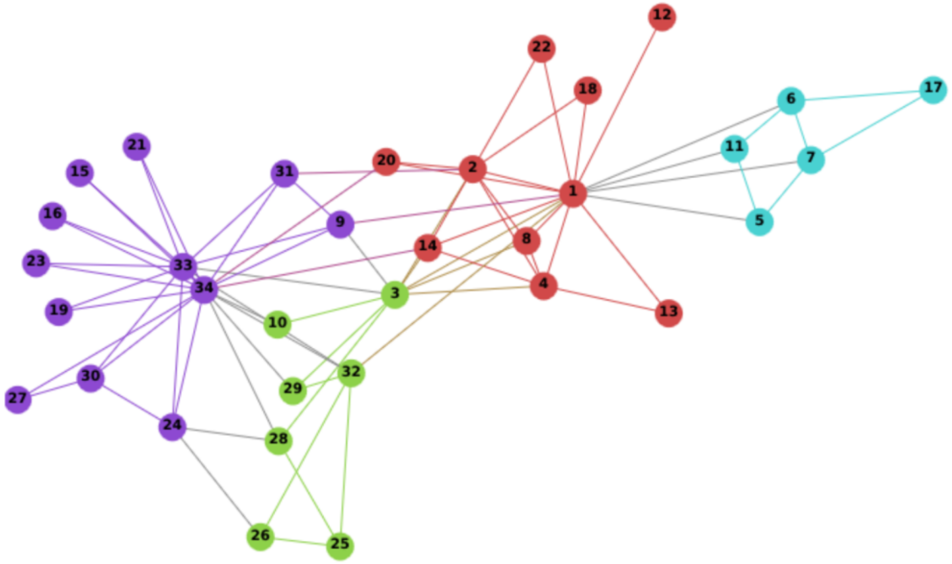
- **DeepWalk** ([Perozzi et al., 2013]): Use fixed-length, unbiased random walks starting from every node

- **node2vec** ([Grover and Leskovec, 2016]): Use biased random walks that can trade off between local and global views of the graph
  - Interpolate between BFS and DFS
  - In-out parameter and return parameter

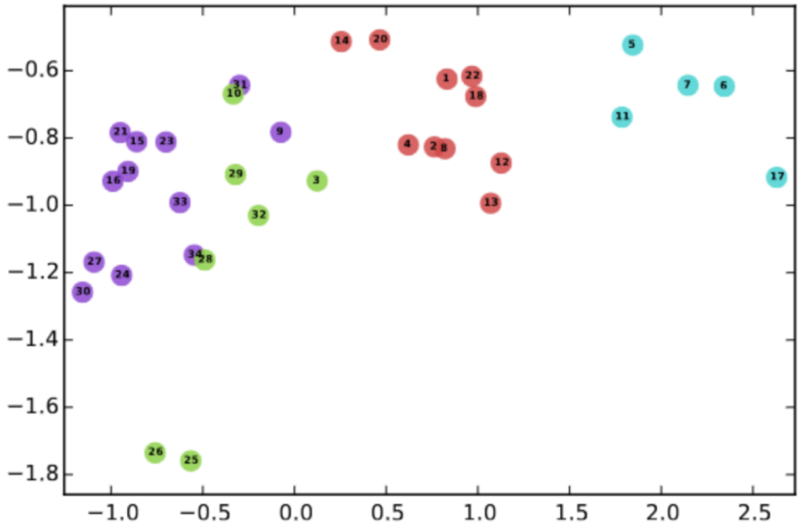
DeepWalk

[Perozzi et al., 2014]

- Fixed length, unbiased random walks from every node



(a) Input: Karate Graph



(b) Output: Representation

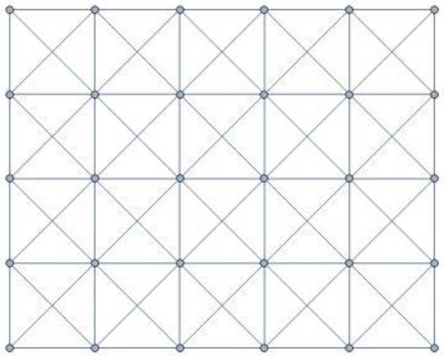
Limitations of Node Embeddings

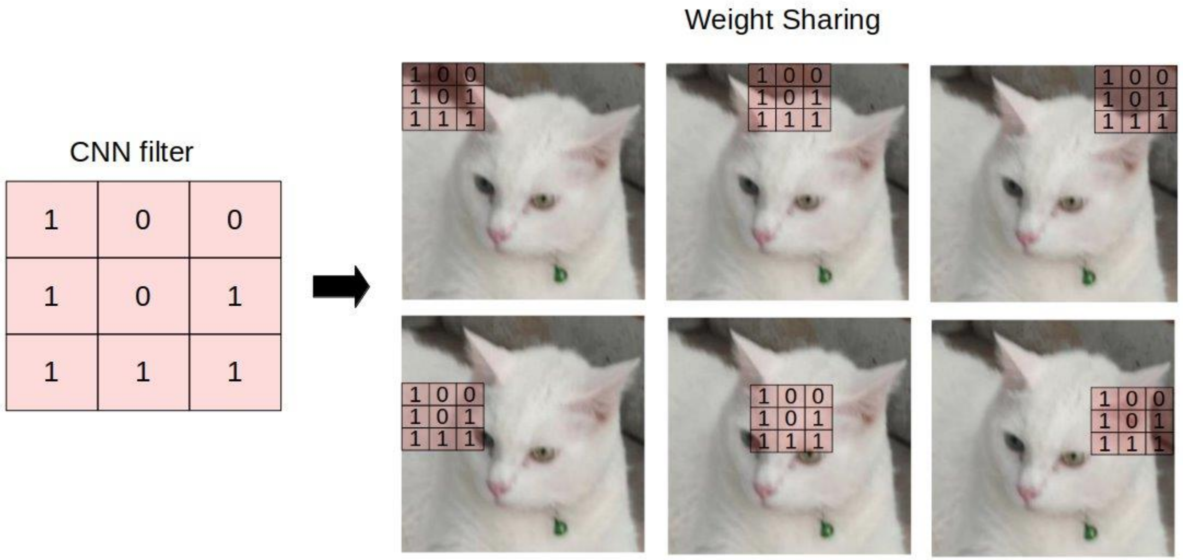
- *Transductive*: Cannot get embeddings for nodes not seen during training, e.g., in new or dynamic graphs
- Unable to capture common structural properties across long distances
- Unclear how to incorporate rich node-, edge-, and graph-level features

SOLUTION: Graph neural networks (GNNs) for deep representation learning!

Intuition: Convolutional Networks

- Weight sharing
- Filters capture neighborhood on a grid graph





1	0	0
1	0	1
1	1	1

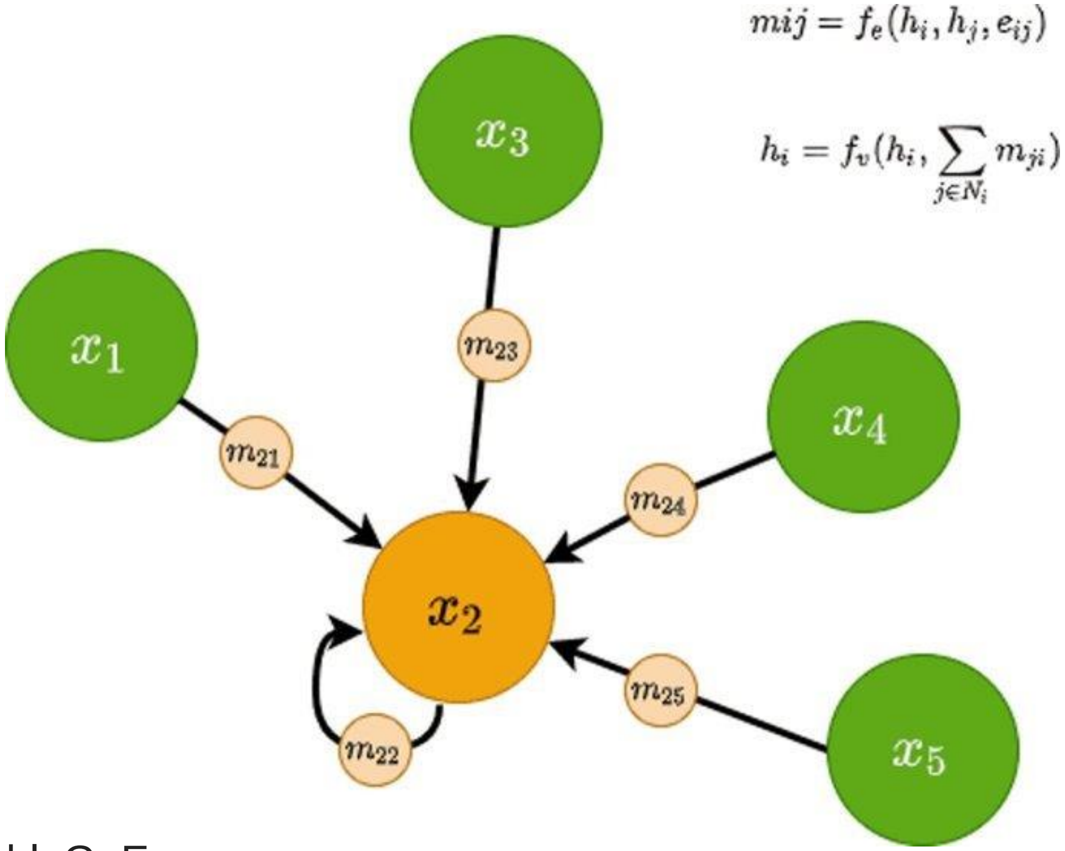
- GNNs generalize this intuition to general graphs
- Challenge: neighborhoods look different!

Image from https://towardsdatascience.com/understanding-graph-convolutional-networks-for-node-classification-a2bfdb7aba7b

GRAPH NEURAL NETWORKS (GNNs)

GNNs - Message Passing Networks (MPNNs)

- Combine node-, edge-, and graph-level features
- Perform iterative message passing step
- Deal with varying local neighborhoods
- Parameter sharing (#params not depending on graph size)


$mij = f_e(h_i, h_j, e_{ij})$
$h_i = f_v(h_i, \sum_{j \in N_i} m_{ji})$
x_3
x_1
m_{23}
x_4
m_{21}
m_{24}
x_2
m_{25}
m_{22}
x_5

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. Neural message passing for quantum chemistry. ICLM 2017.

MPNNs: Aggregate and Update

- Initial node features $\mathbf{h}_u^{(0)}$
- Edge features $\mathbf{e}_{u,v}$
- Iteratively perform L message passing steps to produce node embeddings $h_u^{(1)}$, $h_u^{(2)}$, … $h_u^{(L)}$:

$$\mathbf{m}_{u,v}^{(t+1)} = M_t(\mathbf{h}_u^{(t)}, \mathbf{h}_v^{(t)}, \mathbf{e}_{u,v})$$

Message from neighbor v to u

$$\mathbf{a}_u^{(t+1)} = f_{\text{aggregate}}(\{\!\{\mathbf{m}_{u,v}^{(t+1)} \mid v \in \mathcal{N}(u)\}\!\})$$

Permutation invariant function (e.g., SUM, MAX, AVG)

$$\mathbf{h}_u^{(t+1)} = f_{\text{update}}(\mathbf{h}_u^{(t)}, \mathbf{a}_u^{(t+1)})$$

Final update combining aggregation with self features

Readout Layer

- After L message-passing layers, we get embeddings $h_u^{(L)}$ at each u
- How to convert to a final prediction?
- Use a *readout layer*

Node-level task:

$$\mathbf{g}_v = f_{\text{readout}}(\mathbf{h}_v^{(L)})$$

Graph-level task:

$$\mathbf{h}_G = f_{\text{readout}}(\{\!\{\mathbf{h}_v^{(L)} \mid v \in V\}\!\})$$

Edge-level task:

$$\mathbf{g}_{u,v} = f_{\text{readout}}(\mathbf{h}_u^{(L)}, \mathbf{h}_v^{(L)})$$

GNN in Action: Node Classification

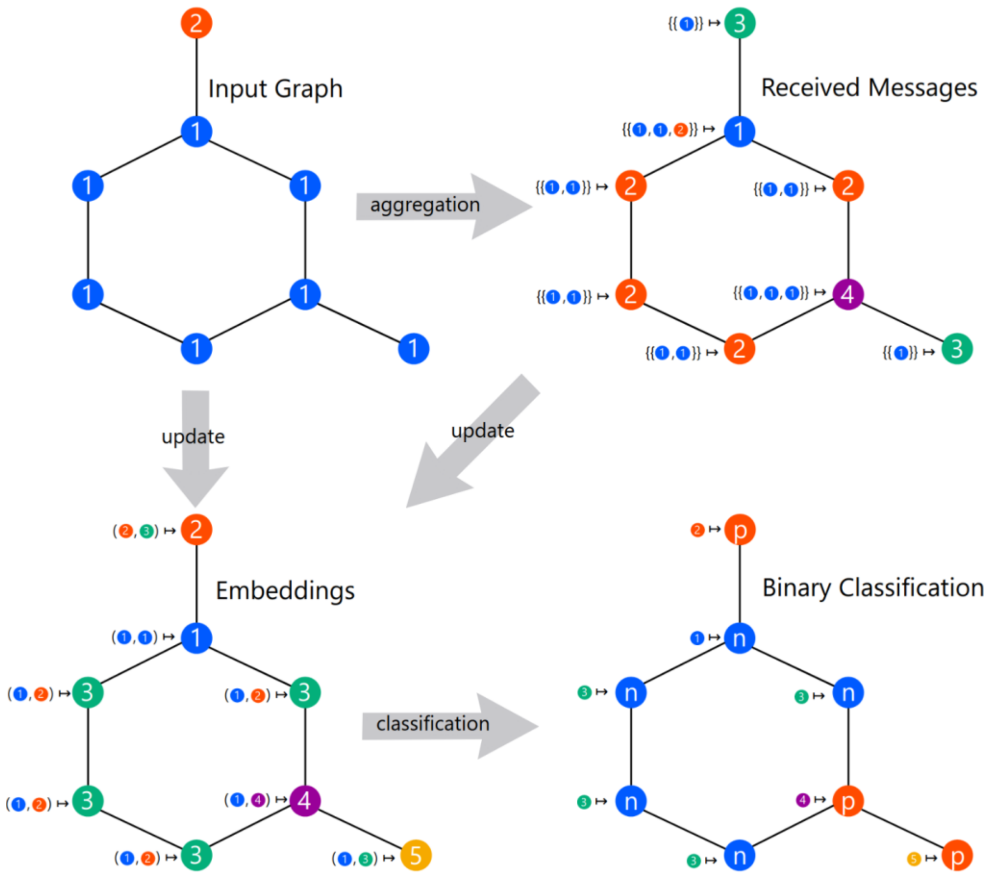


Figure 1: One-layered message passing graph neural networks.

Sato, R. (2020). A survey on the expressive power of graph neural networks. *arXiv:2003.04078*.

GNN in Action: More Layers

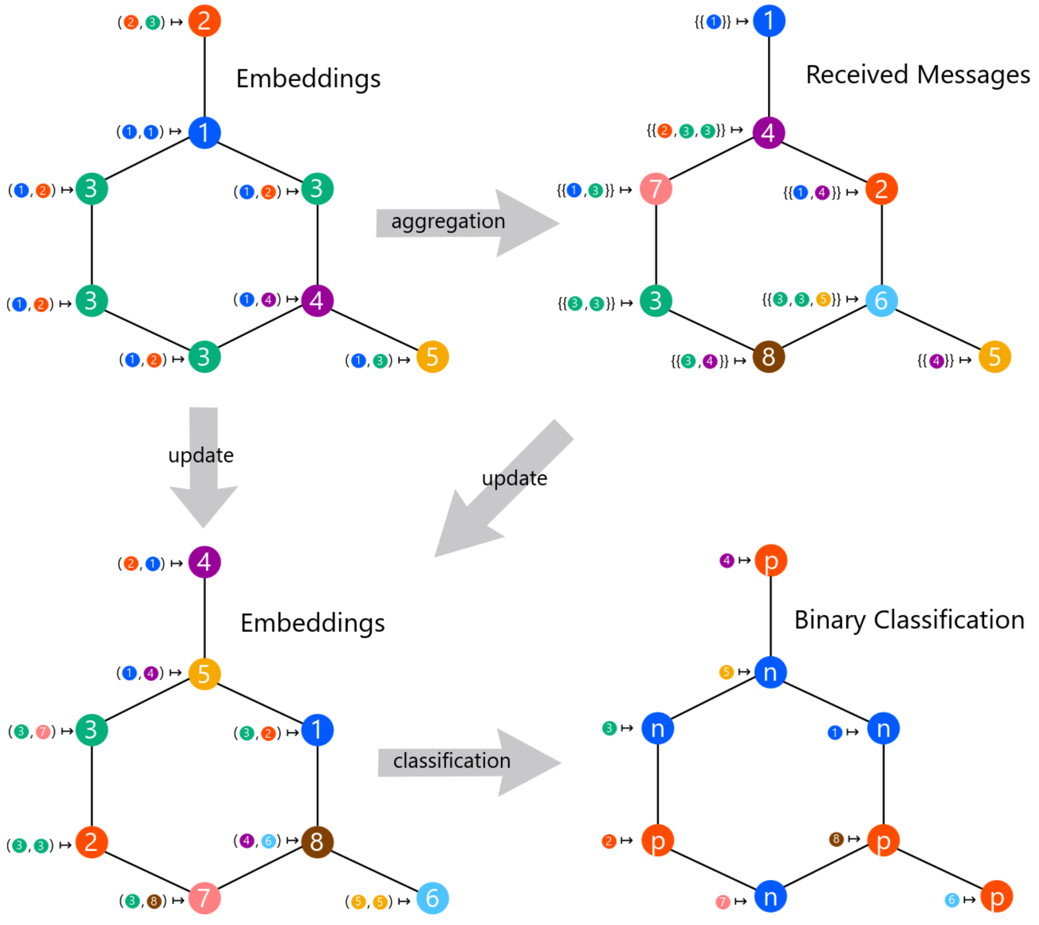


Figure 2: Two-layered message passing graph neural networks.

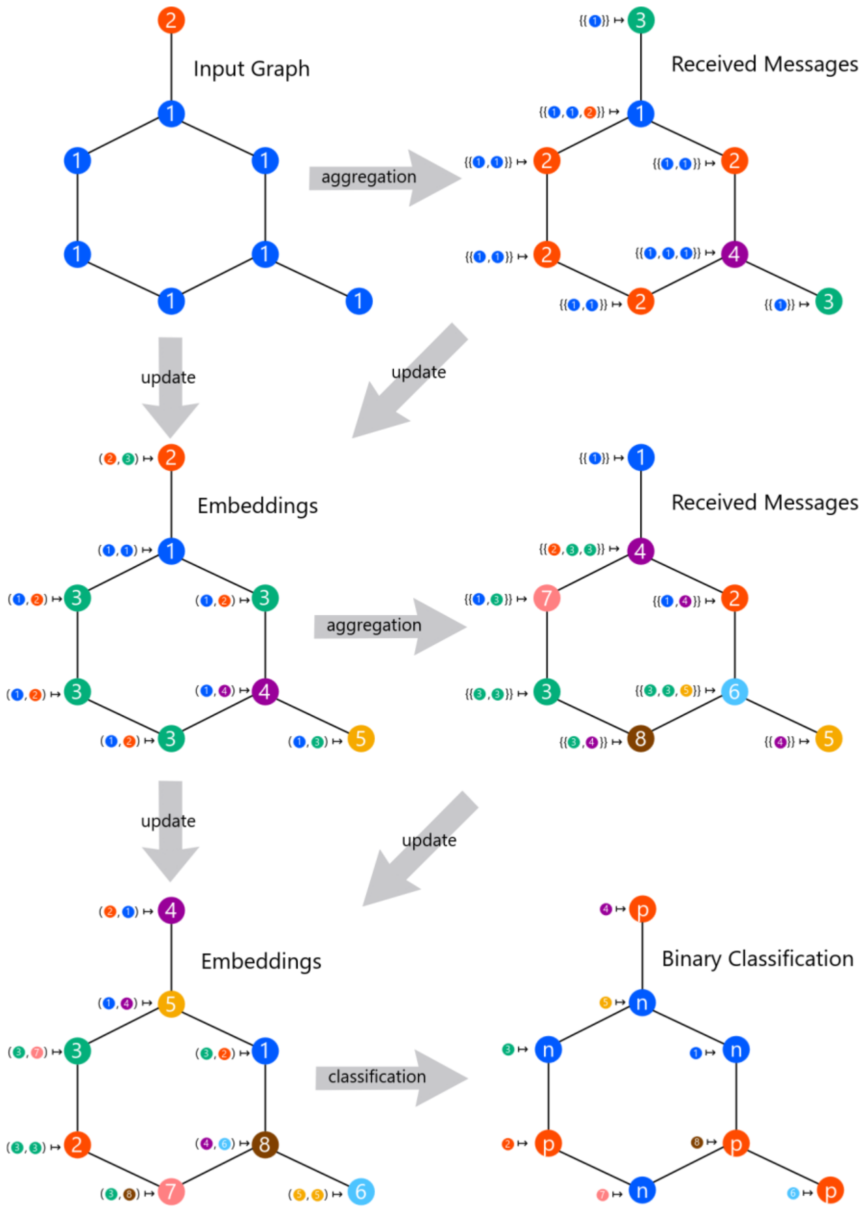


Figure 2: Two-layered message passing graph neural networks.

Sato, R. (2020). A survey on the expressive power of graph neural networks. *arXiv:2003.04078*.

Graph Convolutional Networks (GCN)

[Kipf and Welling, 2017]

$$\widehat{A} = A + I$$

$$\widehat{D} = D + I$$

- A = *n* x *n* adjacency matrix
- D = *n* x *n* diagonal (degree) matrix

- Update Rule:

Column-stacked node representations

$$\mathbf{H}^{(t+1)} = \sigma\left(\mathbf{W}^{(t)}\mathbf{H}^{(t)}\widehat{D}^{-1/2}\widehat{A}\widehat{D}^{-1/2}\right)$$

Nonlinear activation (e.g., ReLU)

d_{t+1} x d_t weight matrix (learnable)

Graph Convolutional Networks (GCN)

[Kipf and Welling, 2017]

$$\mathbf{m}_{u,v}^{(t+1)} = M_t(\mathbf{h}_u^{(t)}, \mathbf{h}_v^{(t)}, \mathbf{e}_{u,v}) = \frac{\mathbf{h}_v^{(t)}}{\sqrt{\deg(u)\deg(v)}}$$

$$\mathbf{a}_u^{(t+1)} = f_{\text{aggregate}}(\{\!\{\mathbf{m}_{u,v}^{(t+1)} \mid v \in \mathcal{N}(u)\}\!\}) = \sum_{v \in \mathcal{N}(u)} \mathbf{m}_{u,v}^{(t+1)}$$

$$\mathbf{h}_u^{(t+1)} = f_{\text{update}}(\mathbf{h}_u^{(t)}, \mathbf{a}_u^{(t+1)}) = \sigma\left(\mathbf{W}^{(t)}\left(\mathbf{a}_u^{(t+1)} + \frac{\mathbf{h}_u^{(t)}}{\deg(u)}\right)\right)$$

- Update Rule:

Column-stacked node representations

$$\mathbf{H}^{(t+1)} = \sigma\left(\mathbf{W}^{(t)}\mathbf{H}^{(t)}\widehat{D}^{-1/2}\widehat{A}\widehat{D}^{-1/2}\right)$$

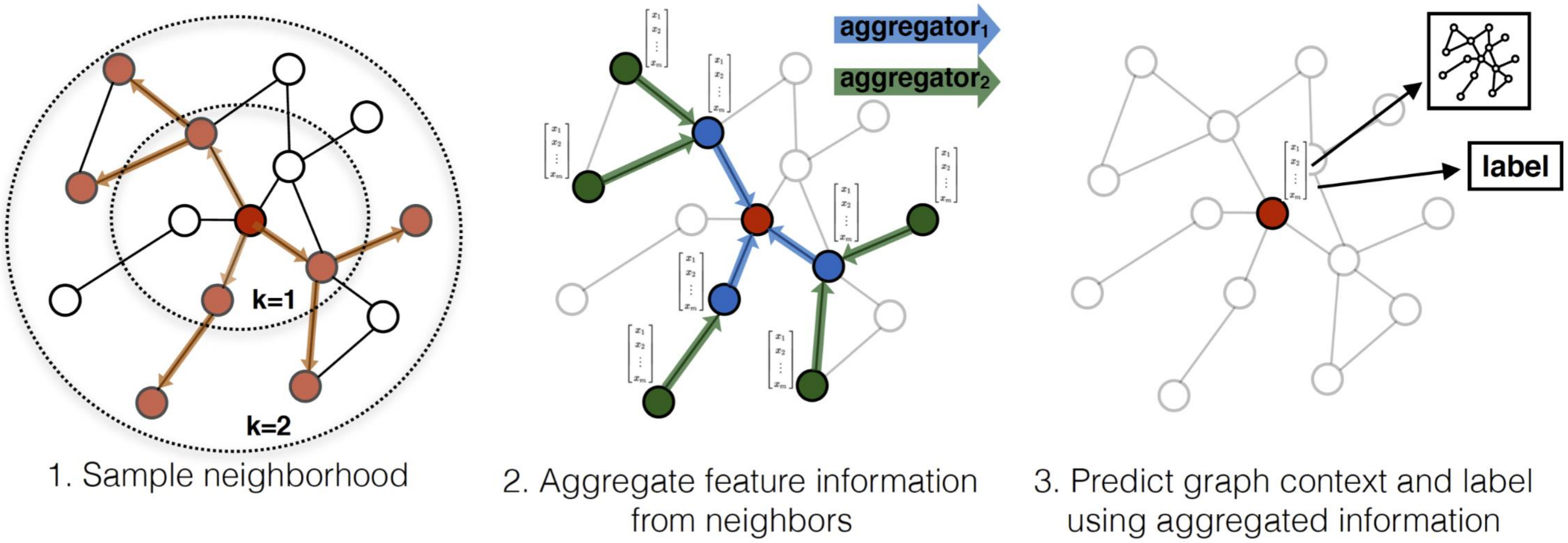
Nonlinear activation (e.g., ReLU)

d_{t+1} x d_t weight matrix (learnable)

GraphSAGE

[Hamilton et al., 2017]

- Idea: Sample and aggregate
- Use random samples of neighbors from multiple hops



1. Sample neighborhood

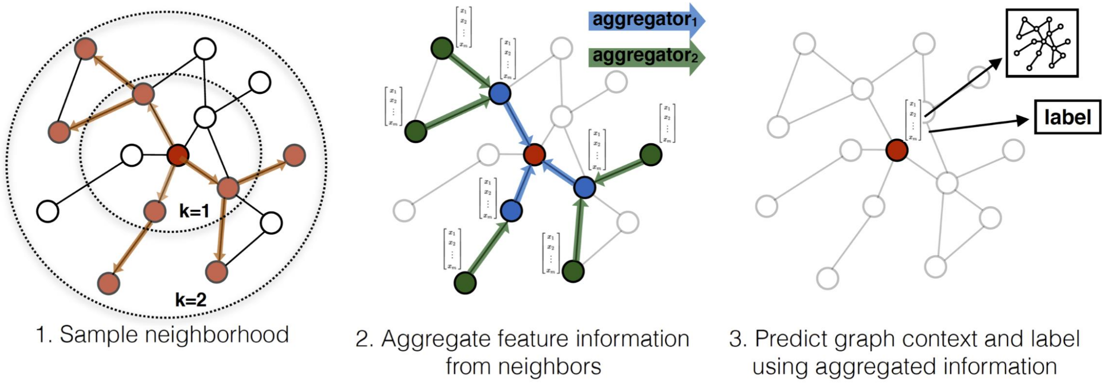
2. Aggregate feature information from neighbors

3. Predict graph context and label using aggregated information

Figure 1: Visual illustration of the GraphSAGE sample and aggregate approach.

GraphSAGE

[Hamilton et al., 2017]



k=1
k=2
aggregator1
aggregator2
label
1. Sample neighborhood
2. Aggregate feature information from neighbors
3. Predict graph context and label using aggregated information

Figure 1: Visual illustration of the GraphSAGE sample and aggregate approach.

$$\mathbf{m}_{u,v}^{(t+1)} = M_t(\mathbf{h}_u^{(t)}, \mathbf{h}_v^{(t)}, \mathbf{e}_{u,v}) = \mathbf{h}_v^{(t)}$$

Neighborhood = fixed-size sample from 1-hop, 2-hop neighbors

$$\mathbf{a}_u^{(t+1)} = f_{\text{aggregate}}(\{\!\{\mathbf{m}_{u,v}^{(t+1)} \mid v \in \mathcal{N}(u)\}\!\}) = \frac{1}{\deg(u)} \sum_{v \in \mathcal{N}(u)} \mathbf{m}_{u,v}^{(t+1)}$$

$$\mathbf{h}_u^{(t+1)} = f_{\text{update}}(\mathbf{h}_u^{(t)}, \mathbf{a}_u^{(t+1)}) = \sigma(\mathbf{W}^{(t)}[\mathbf{h}_u^{(t)}, \mathbf{a}_u^{(t+1)}])$$

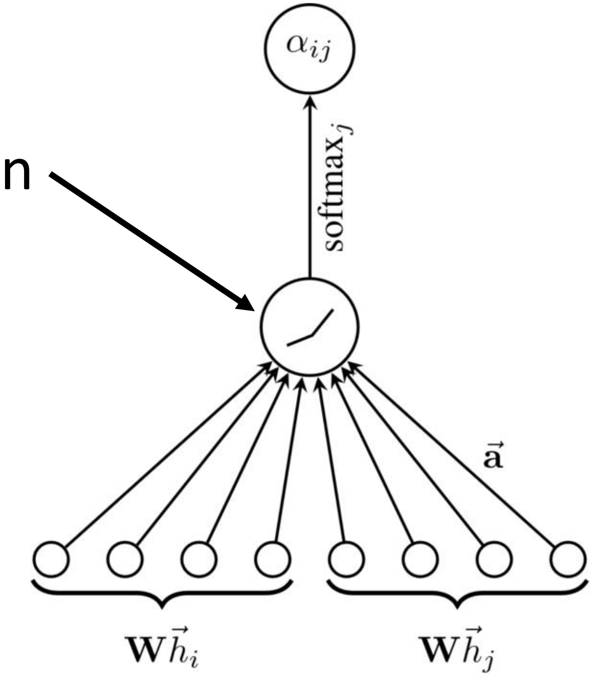
Nonlinear activation (e.g., ReLU)

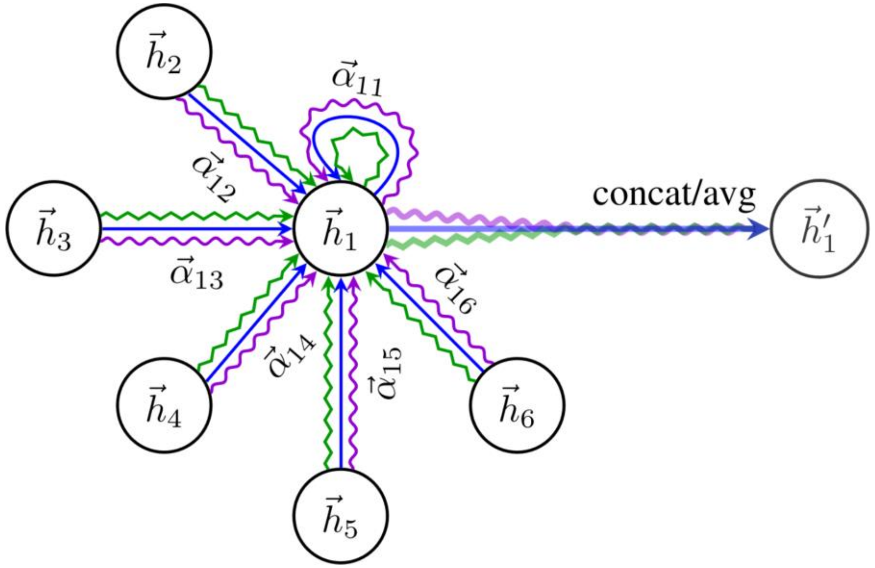
Graph Attention Networks (GAT)

[Veličković et al., 2018]

- Assign importances to neighbors in the aggregation step
- Use an attention mechanism to compute scores

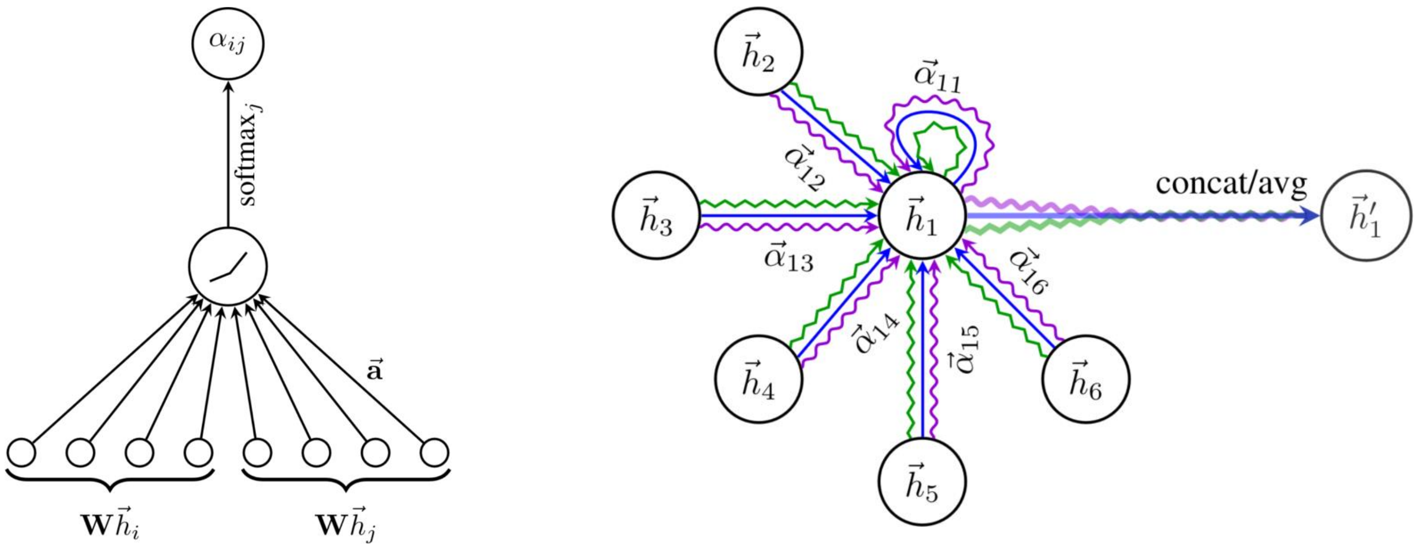
LeakyReLU activation function





Graph Attention Networks (GAT)

[Veličković et al., 2018]



$$\mathbf{m}_{u,v}^{(t+1)} = \mathbf{h}_v^{(t)}$$

$$\mathbf{a}_u^{(t+1)} = \sum_{v \in \mathcal{N}(u)} \alpha_{u,v}^{(t+1)} \mathbf{m}_{u,v}^{(t+1)}$$

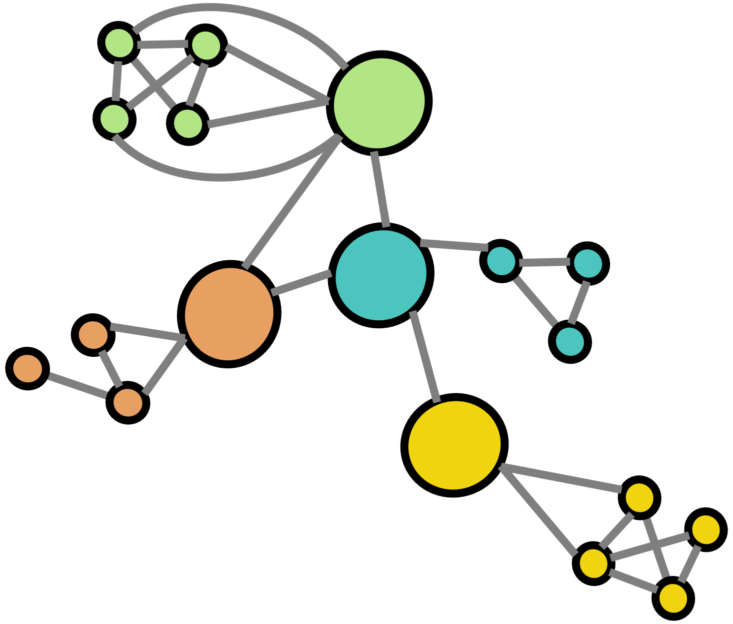
$$\mathbf{h}_u^{(t+1)} = \sigma(\mathbf{W}^{(t)} \mathbf{a}_u^{(t+1)})$$

Attention Scores:

$$\alpha_{u,v}^{(t+1)} = \frac{\exp(\textsc{LeakyReLU}((\mathbf{b}^{(t)})^\top [\mathbf{W}^{(t)} \mathbf{h}_u^{(t)} \parallel \mathbf{W}^{(t)} \mathbf{h}_v^{(t)}]))}{\sum_{w \in \mathcal{N}(u)} \exp(\textsc{LeakyReLU}((\mathbf{b}^{(t)})^\top [\mathbf{W}^{(t)} \mathbf{h}_u^{(t)} \parallel \mathbf{W}^{(t)} \mathbf{h}_w^{(t)}]))}$$

TOOLS FOR GRAPH LEARNING

Spectral Graph Theory



$$= \begin{array}{ccc} \boldsymbol{\Phi} & \boldsymbol{\Lambda} & \boldsymbol{\Phi}^T \\ \begin{pmatrix} | & \vdots & | \\ \mathbf{v_1} & \vdots & \mathbf{v_n} \\ | & \vdots & | \end{pmatrix} & \begin{pmatrix} \lambda_0 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \lambda_n \end{pmatrix} & \begin{pmatrix} - & \mathbf{v_1} & - \\ \cdots & \cdots & \cdots \\ - & \mathbf{v_n} & - \end{pmatrix} \end{array}$$

Laplacian Eigenvectors

Eigenvectors are a set orthonormal functions that minimize the Rayliegh Quotient on the graph

[Chung, 1997]

$$L = D - A = \Phi \Lambda \Phi^T = \sum_i^n \lambda_i \phi_i \phi_i^T$$

$$\mathcal{L} = D^{-1/2} L D^{-1/2} = \hat{\Phi} \hat{\Lambda} \hat{\Phi}^T$$

$$\Phi = [\phi_1, \phi_2, \ldots, \phi_n]$$

$$\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$$

$$\begin{cases} L\phi_1 & = \lambda_1 \phi_1 = 0 \\ L\phi_2 & = \lambda_2 \phi_2 \ (\text{s.t } \phi_2 \perp \phi_1) \\ \ldots & \\ L\phi_n & = \lambda_n \phi_n \ (\text{s.t } \phi_n \perp (\phi_1, \ldots, \phi_{n-1})) \end{cases}$$

$$\lambda_1 = 0 < \lambda_2 \leq \cdots \lambda_n$$

Laplacian Eigenvectors

Eigenvectors are a set orthonormal functions that minimize the Rayliegh Quotient on the graph

[Chung, 1997]

Variability of node signal wrt graph structure

$$L = D - A = \Phi\Lambda\Phi^T = \sum_{i}^{n} \lambda_i \phi_i \phi_i^T$$

$$\mathcal{L} = D^{-1/2} L D^{-1/2} = \hat{\Phi}\hat{\Lambda}\hat{\Phi}^T$$

$$\Phi = [\phi_1, \phi_2, \ldots, \phi_n]$$

$$\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$$

$$\begin{cases} L\phi_1 & = \lambda_1\phi_1 = 0 \\ L\phi_2 & = \lambda_2\phi_2 \text{ (s.t } \phi_2 \perp \phi_1) \\ \ldots & \\ L\phi_n & = \lambda_n\phi_n \text{ (s.t } \phi_n \perp (\phi_1, \ldots, \phi_{n-1})) \end{cases}$$

$$\lambda_1 = 0 < \lambda_2 \leq \cdots \lambda_n$$

$$f^\top L f = \sum_{(u,v)\in E} (f_u - f_v)^2$$

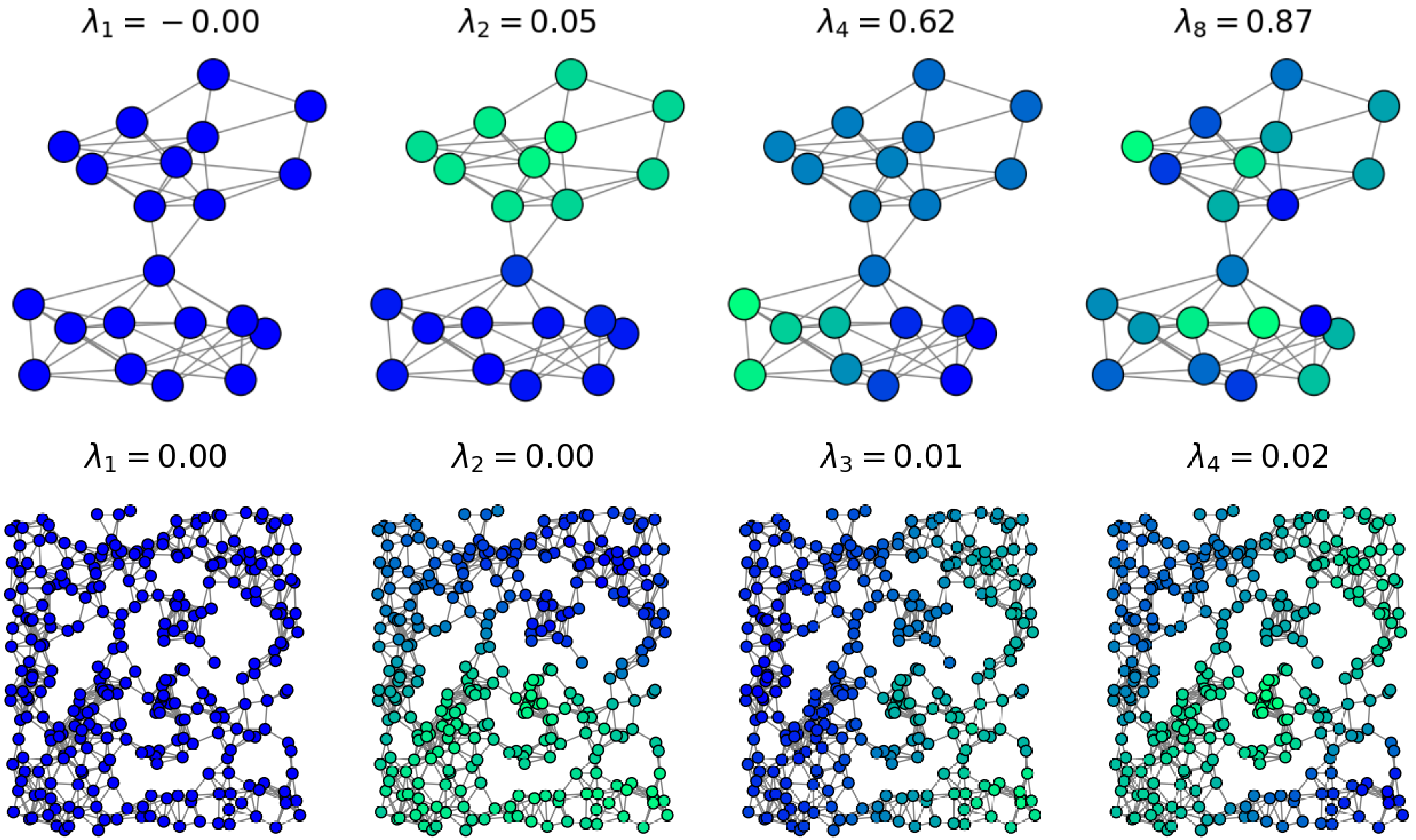
$$R_G(f) = \frac{f^T L f}{\|f\|_2^2}$$

$$\phi_n = \mathrm{argmin}_{f \perp (f_1, \ldots, f_{n-1})} R_G(f)$$

$$\lambda_n = R_G(\phi_n) = \frac{\phi_n^T L \phi_n}{\phi_n^T \phi_n}$$

Eigenvalues are the Rayleigh Quotient
of the **eigenvectors** of the graph
(**orthonormal functions that minimizes variability
wrt the structure of the graph**)

Laplacian Eigenvectors



Cheeger constant [Cheeger 1970; Chung 1997]

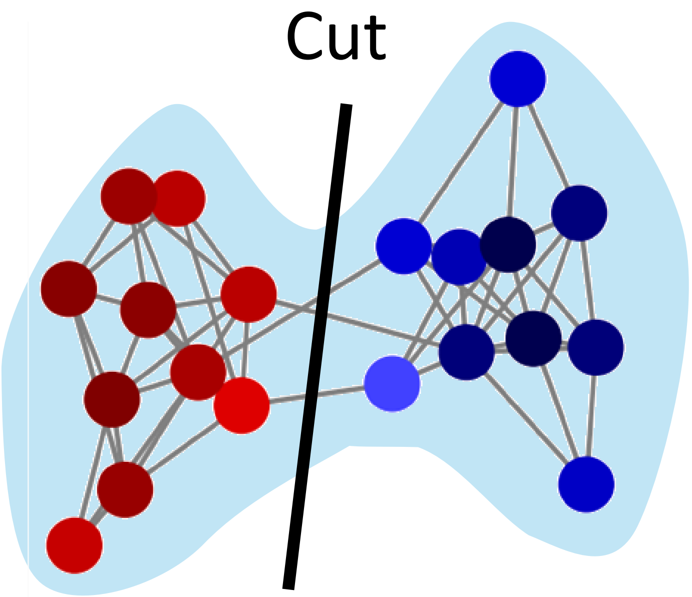
Min Cut

Size of the minimum **cut** to disconnect the graph

$$\partial S = \{e = (u, v) : u \in S, v \in \bar{S}\}$$ # of inter-edges

$$h_S = \frac{|\partial S|}{\min\{\mathrm{vol}(S),\ \mathrm{vol}(\bar{S})\}}$$ Normalized by minimum volume of node subset

$$\boxed{h_G = \min_{S \subset V} h_S}$$ **Cheeger Constant**



Cheeger constant [Cheeger 1970; Chung 1997]

Min Cut

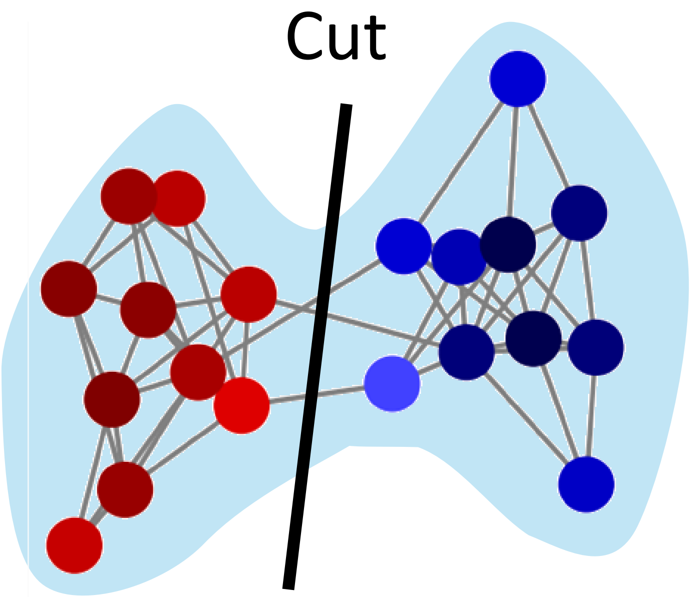
Size of the minimum **cut** to disconnect the graph

$$\partial S = \{e = (u,v) : u \in S, v \in \bar{S}\}$$ # of inter-edges

$$h_S = \frac{|\partial S|}{\min\{\mathrm{vol}(S),\ \mathrm{vol}(\bar{S})\}}$$ Normalized by minimum volume of node subset

$$h_G = \min_{S \subset V} h_S$$ **Cheeger Constant**

$$\frac{\lambda_2}{2} \le h_G \le \sqrt{2\lambda_2}$$ **Cheeger Inequality**



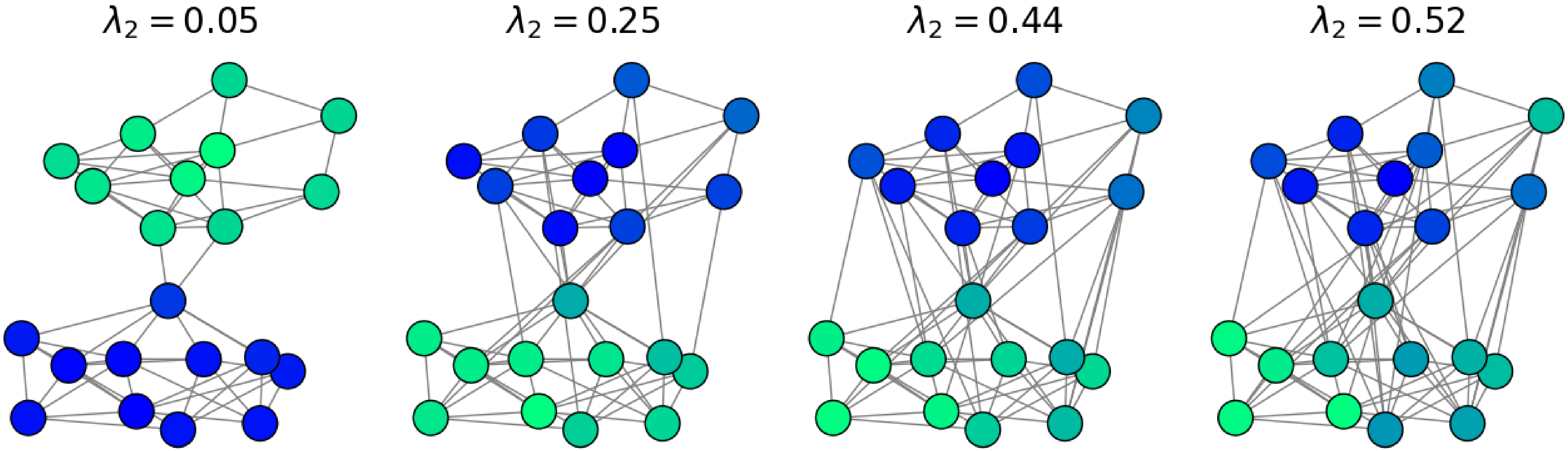
Size of Cut as Laplacian Quadratic form $$x_i = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{if } i \notin S \end{cases}$$

$$|\partial S| = x^\top L x = \sum_{(u,v)\in E} (x_u - x_v)^2 = \sum_{(u,v)\in E} \mathbb{I}\left[(u,v) \in \partial S\right]$$

$$\lambda_2 = \min_{f \perp \mathbf{1}} \frac{f^T L f}{f^T f}$$

Laplacian Eigenvectors

$$\boldsymbol{\lambda_2}$$



$\lambda_2 = 0.05$

$\lambda_2 = 0.25$

$\lambda_2 = 0.44$

$\lambda_2 = 0.52$

Minimum amount of energy needed to disconnect the graph

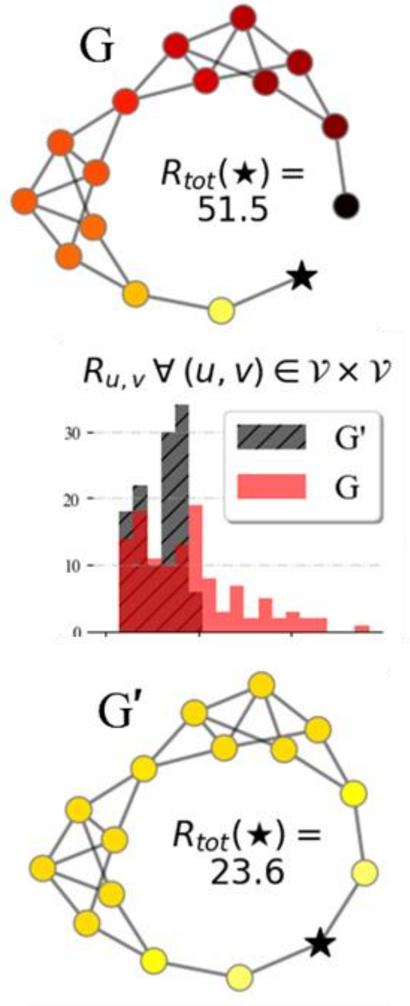
Bottleneck

Effective Resistance and Commute Time

- Commute time: **Expected number of steps** for a Random-Walker **go** from **u** to **v** and **come back**
  Captures **global behavior** and **long-range** dependencies

$$\mathrm{CT}(u,v) = H(u,v) + H(v,u)$$

$$R_{u,v} = \frac{\mathrm{CT}}{\mathrm{vol}(G)}$$



The **more** and **shorter paths** between a pair of nodes
The smaller $\mathbf{R_{uv}}$ is

Even if SP does not change

Effective Resistance and Commute Time

- Commute time: **Expected number of steps** for a Random-Walker **go** from **u** to **v** and **come back**
  Captures **global behavior** and **long-range** dependencies

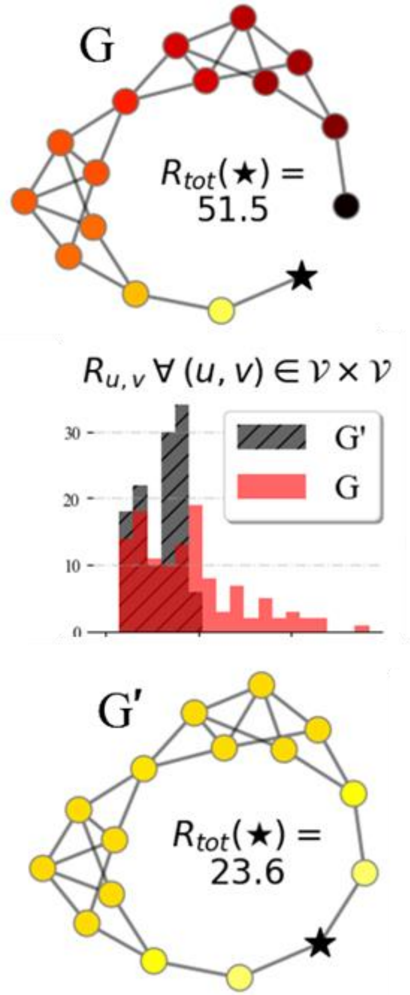
$$\boxed{\begin{gathered}\mathrm{CT}(u,v) = H(u,v) + H(v,u) \\ R_{u,v} = \frac{\mathrm{CT}}{\mathrm{vol}(G)}\end{gathered}}$$

$$R_{u,v} = \sum_{i=1}^{n} \frac{1}{\lambda_i} \left(\frac{\phi_i(u)}{\sqrt{d_u}} - \frac{\phi_i(v)}{\sqrt{d_v}} \right)^2$$

$$L^+ = \sum_{i>0} \frac{1}{\lambda_i} \phi_i \phi_i^T = \left(L + \frac{\mathbf{1}\mathbf{1}^T}{n} \right)^{-1} - \frac{\mathbf{1}\mathbf{1}^T}{n}$$ Pseudo-Inverse

$$R_{u,v} = (\mathbf{e_i} - \mathbf{e_j})^T L^+ (\mathbf{e_i} - \mathbf{e_i}) \;\rightarrow\; R_{u,v} = L^+_{ii} + L^+_{jj} - 2L^+_{ij}$$

$$\mathbf{R} \in \mathbb{R}^{n \times n} = \mathbf{1}\mathrm{diag}(L^+)^T + \mathrm{diag}(L^+)\mathbf{1}^T - 2L^+$$

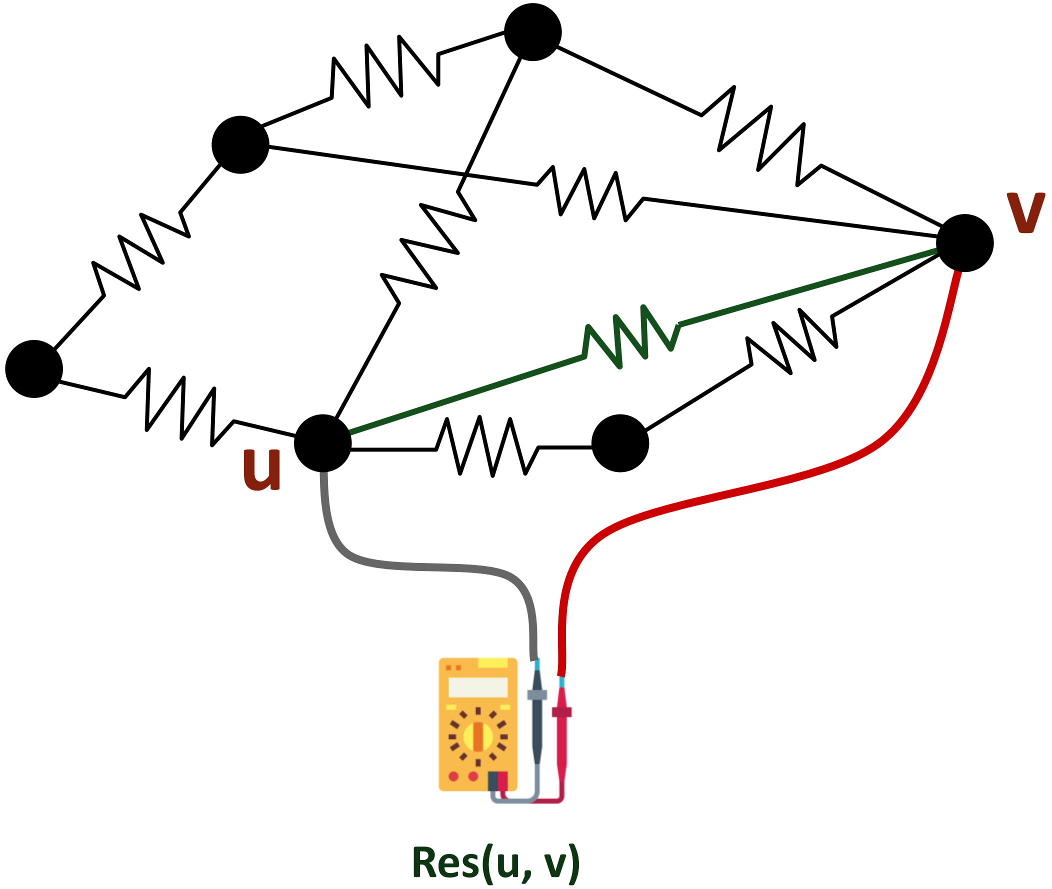


The **more** and **shorter paths** between a pair of nodes
The smaller $\mathbf{R_{uv}}$ is

Even if SP does not change

Effective Resistance

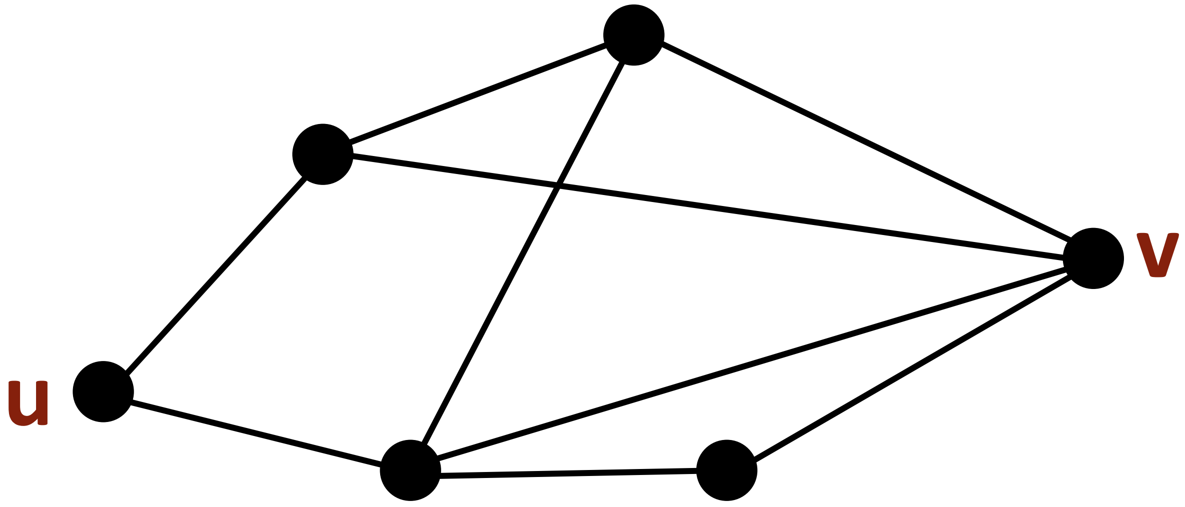
- View graph as electrical circuit
  - Edges are resistors
  - Send current between two points and measure effective resistance
- ERs capture topological structure in graph
- Widely used in theoretical computer science
  - Graph sparsification
  - Linear system solvers
  - Graph clustering



Effective Resistance as Commute Times

- Start random walk from u
- $C_{u,v}$ is expected time to reach v and come back to u

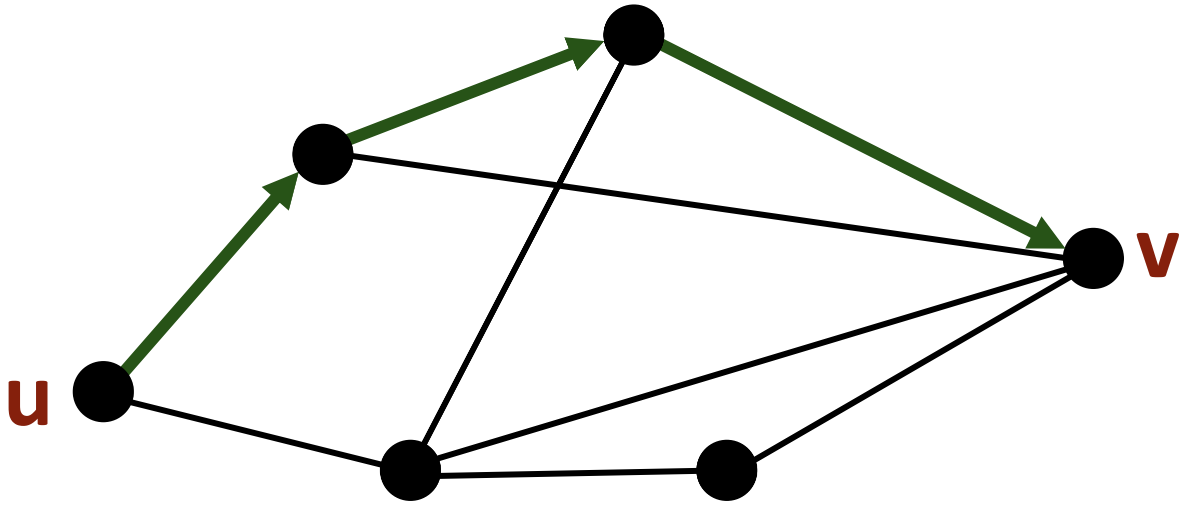
$$\mathrm{Res}(u,v) = \frac{1}{2|E|} C_{u,v}$$



Effective Resistance as Commute Times

- Start random walk from u
- $C_{u,v}$ is expected time to reach v and come back to u

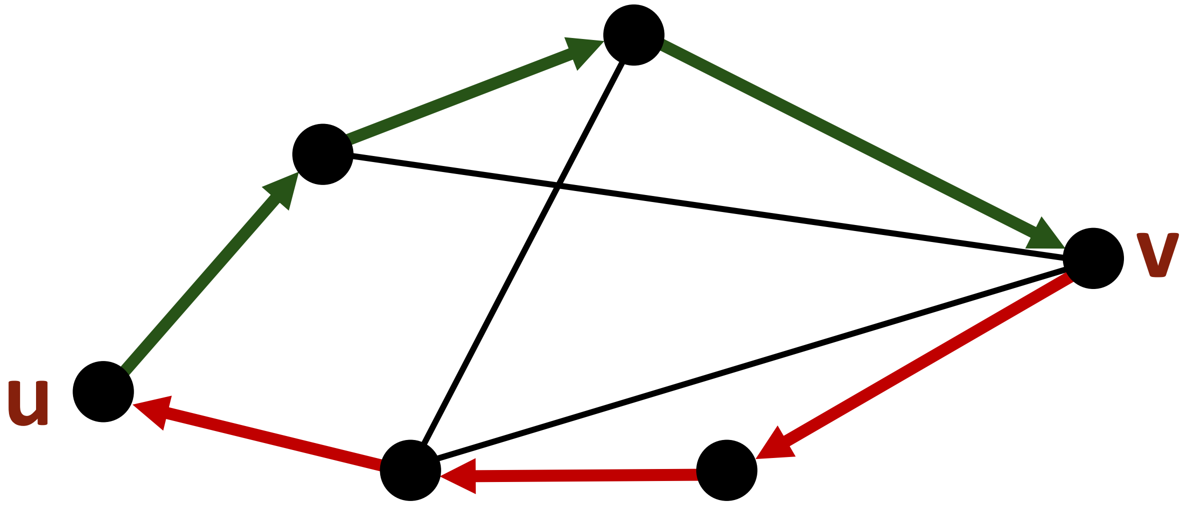
$$\text{Res}(u,v) = \frac{1}{2|E|} C_{u,v}$$



Effective Resistance as Commute Times

- Start random walk from u
- $C_{u,v}$ is expected time to reach v and come back to u

$$\mathrm{Res}(u,v) = \frac{1}{2|E|} C_{u,v}$$

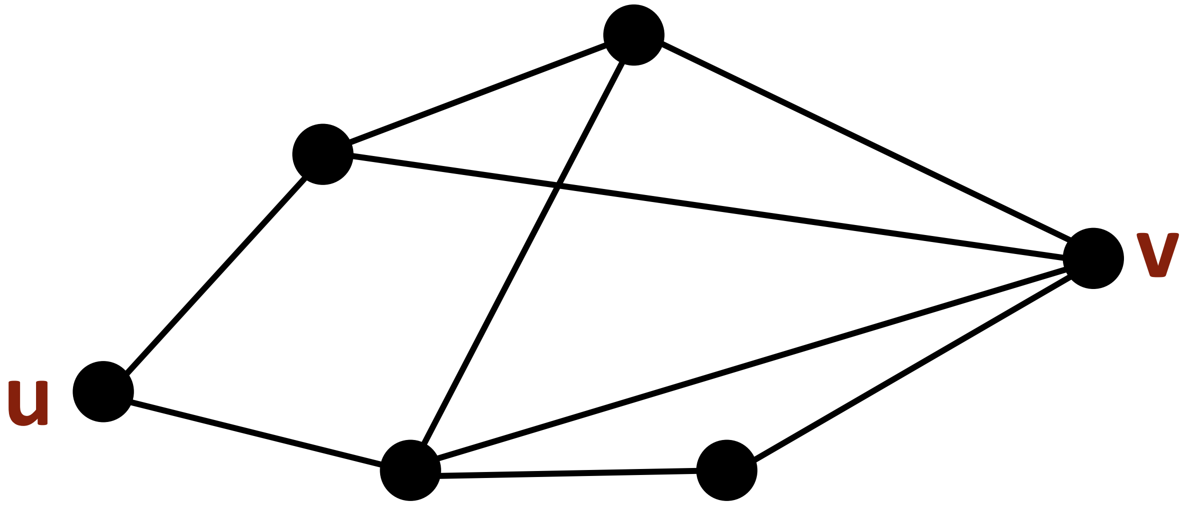


u ⇝ v ⇝ u in 6 steps

Effective Resistance as Commute Times

- Start random walk from u
- $C_{u,v}$ is expected time to reach v and come back to u

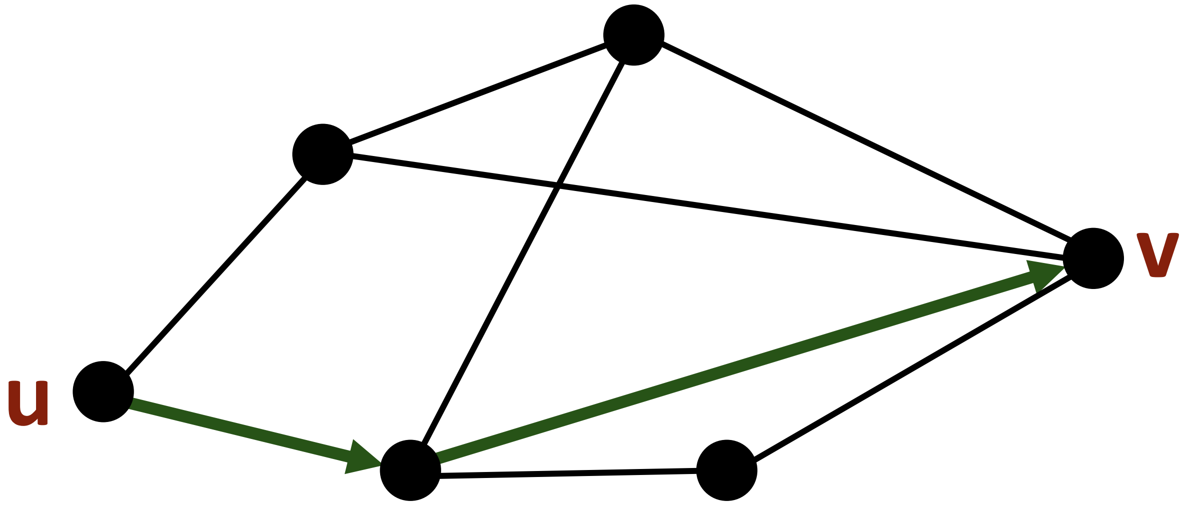
$$\mathrm{Res}(u,v) = \frac{1}{2|E|} C_{u,v}$$



Effective Resistance as Commute Times

- Start random walk from u
- $C_{u,v}$ is expected time to reach v and come back to u

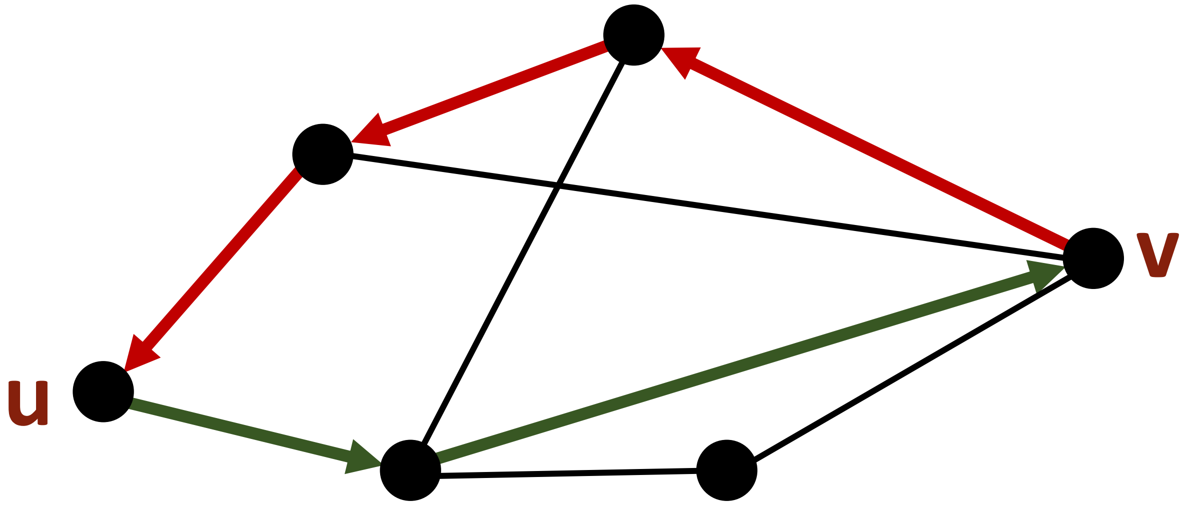
$$\mathrm{Res}(u,v) = \frac{1}{2|E|} C_{u,v}$$



Effective Resistance as Commute Times

- Start random walk from u
- $C_{u,v}$ is expected time to reach v and come back to u

$$\mathrm{Res}(u,v) = \frac{1}{2|E|} C_{u,v}$$

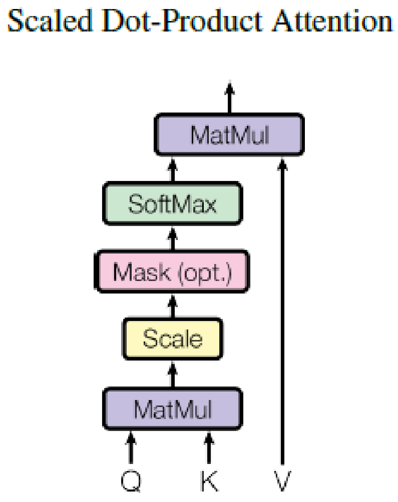


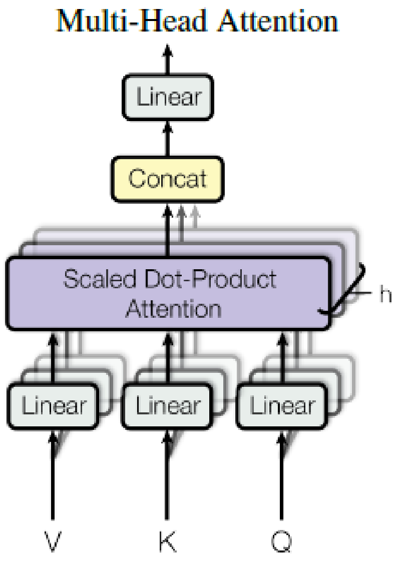
u $\rightsquigarrow$ v $\rightsquigarrow$ u in 5 steps

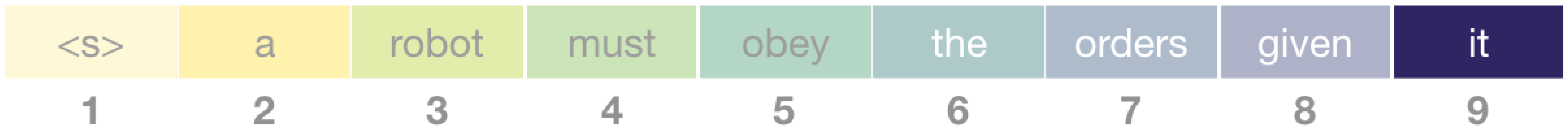
GRAPH TRANSFORMERS

Transformers

- Sequences of tokens
- Next token prediction based on a past context window







- Line graph
- Typically dense

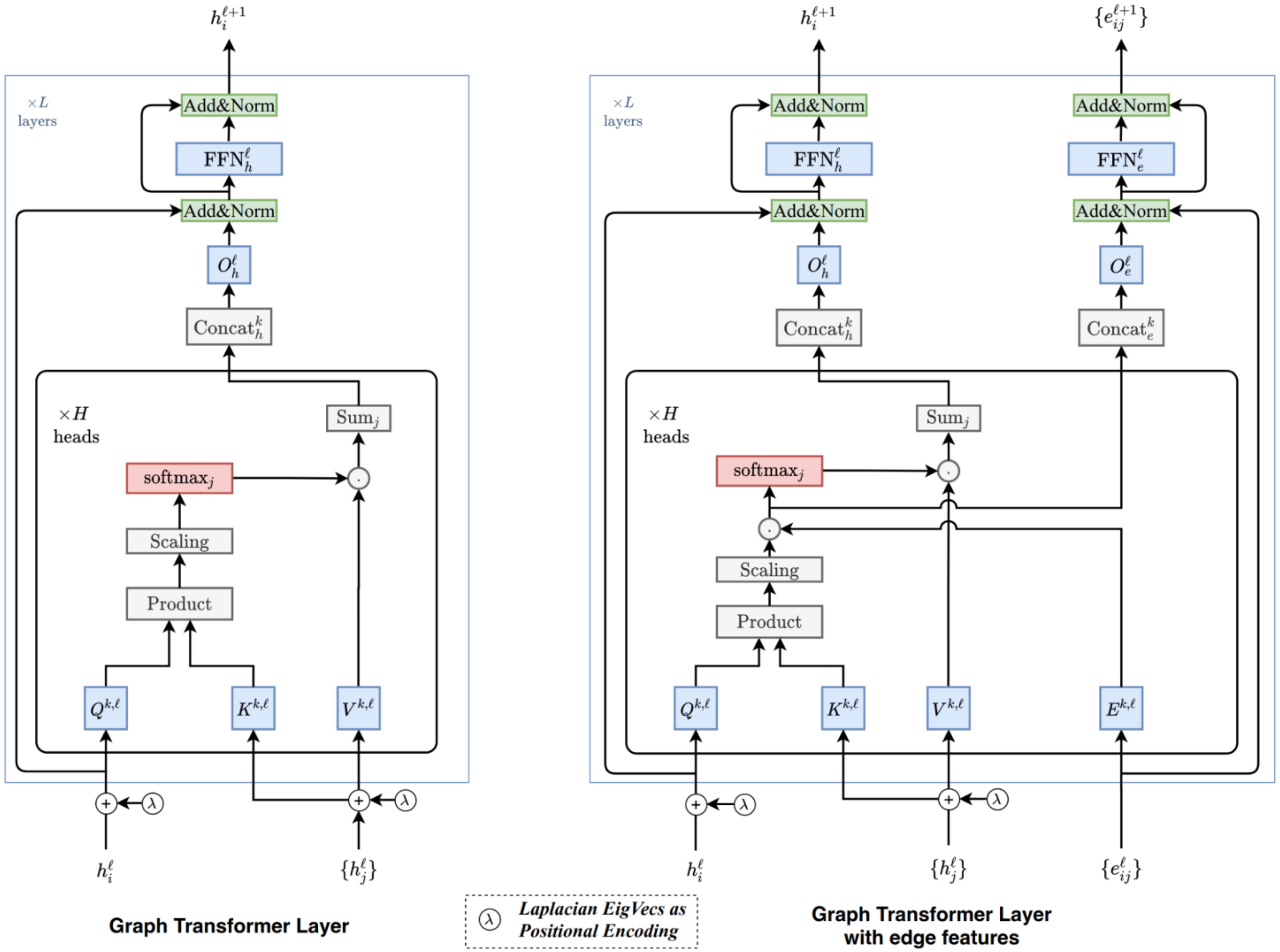
Image from https://towardsdatascience.com/understanding-graph-convolutional-networks-for-node-classification-a2bfdb7aba7b

Graph Transformers

- Extension of transformers to graph-structured data
- Instead of next-token prediction in sequences, learn node representations
- Often dense (full connections) - computational graph different from input graph
- Challenges
  - How to account for loss of inductive bias from input graph structure?
  - How to identify nodes (position, structure) within the graph?
  - Scaling?

Graph Transformers

- Sample architecture ([Dwivedi and Bresson, 2021])
- Number of newer architectures
  - SAN
  - Graphormer
  - GraphGPS



Graph Transformer Layer

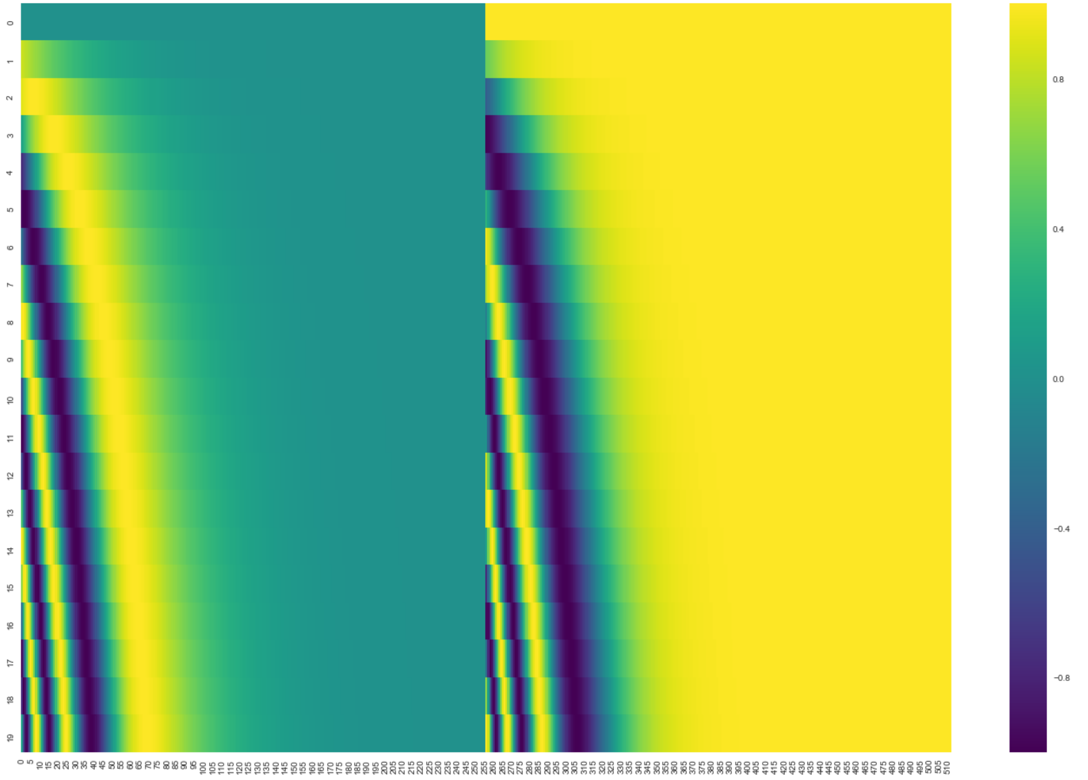
Graph Transformer Layer with edge features

Positional Encodings

- Sequence transformers use sinusoids (sin, cos functions)

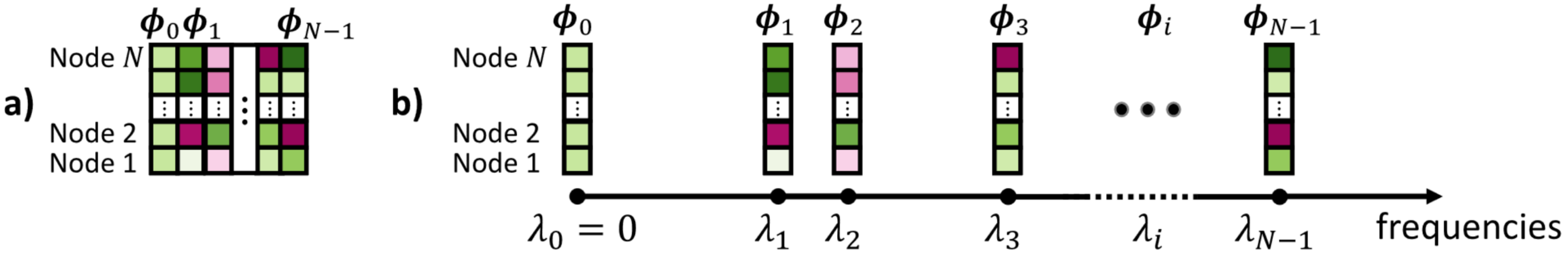
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$



Positional Encodings

- Sin, cos functions arise as eigenfunctions of Laplacians in Euclidean space
- *On graphs*: eigenvectors of graph Laplacian **L = D - A**



Laplacian Eigenvalues/Eigenvectors

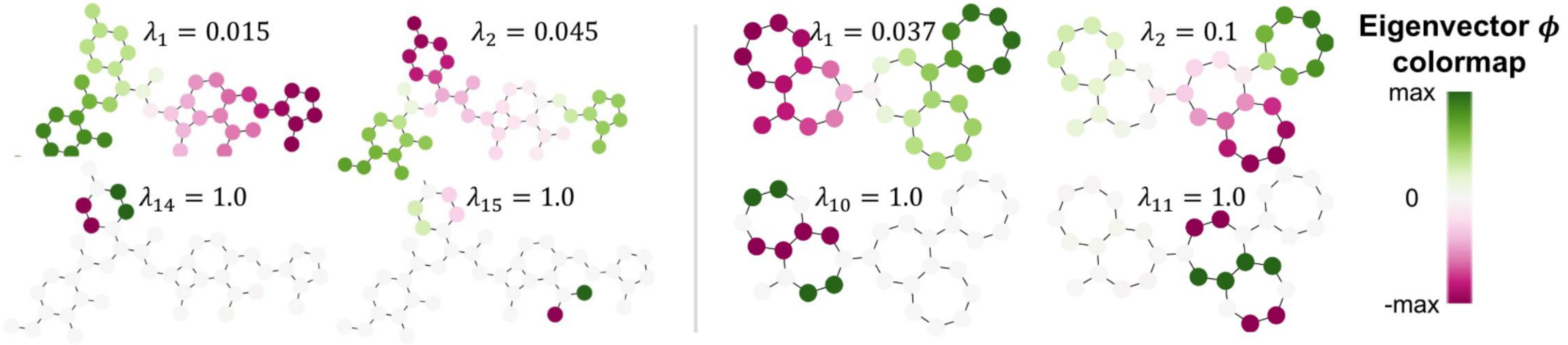
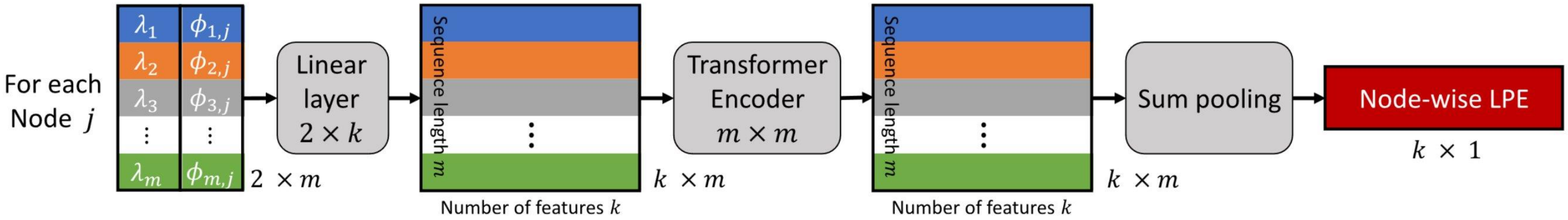


Figure 3: Examples of eigenvalues λ_i and eigenvectors ϕ_i for molecular graphs. The low-frequency eigenvectors ϕ_1, ϕ_2 are spread accross the graph, while higher frequencies, such as ϕ_{14}, ϕ_{15} for the left molecule or ϕ_{10}, ϕ_{11} for the right molecule, often resonate in local structures.

Learned Positional Encodings (LPE)

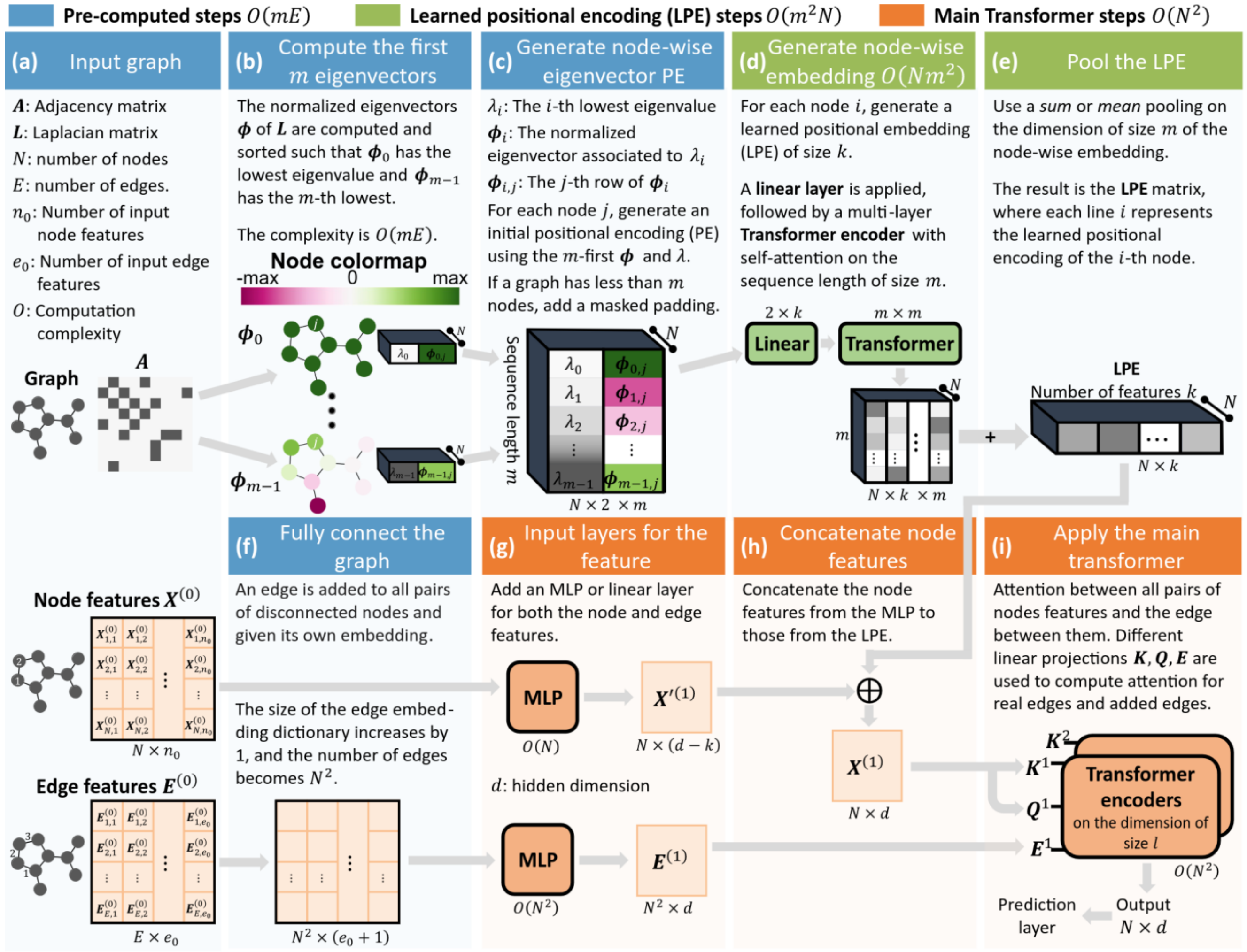
[Kreuzer et al., 2021]

- Learned positional encodings (LPE) on top of Laplacian-based features



Combine LPEs with Attention

- Spectral Attention Networks (SAN) - [Kreuzer et al., 2021]



🟦 Pre-computed steps $O(mE)$ 🟩 Learned positional encoding (LPE) steps $O(m^2N)$ 🟧 Main Transformer steps $O(N^2)$

(a) Input graph

$\boldsymbol{A}$: Adjacency matrix
$\boldsymbol{L}$: Laplacian matrix
N: number of nodes
E: number of edges.
n_0: Number of input node features
e_0: Number of input edge features
O: Computation complexity

(b) Compute the first m eigenvectors

The normalized eigenvectors $\boldsymbol{\phi}$ of $\boldsymbol{L}$ are computed and sorted such that $\boldsymbol{\phi}_0$ has the lowest eigenvalue and $\boldsymbol{\phi}_{m-1}$ has the m-th lowest.

The complexity is $O(mE)$.

(c) Generate node-wise eigenvector PE

λ_i: The i-th lowest eigenvalue
$\boldsymbol{\phi}_i$: The normalized eigenvector associated to λ_i
$\boldsymbol{\phi}_{i,j}$: The j-th row of $\boldsymbol{\phi}_i$

For each node j, generate an initial positional encoding (PE) using the m-first $\boldsymbol{\phi}$ and λ.

If a graph has less than m nodes, add a masked padding.

(d) Generate node-wise embedding $O(Nm^2)$

For each node i, generate a learned positional embedding (LPE) of size k.

A **linear layer** is applied, followed by a multi-layer **Transformer encoder** with self-attention on the sequence length of size m.

(e) Pool the LPE

Use a *sum* or *mean* pooling on the dimension of size m of the node-wise embedding.

The result is the **LPE** matrix, where each line i represents the learned positional encoding of the i-th node.

(f) Fully connect the graph

An edge is added to all pairs of disconnected nodes and given its own embedding.

The size of the edge embedding dictionary increases by 1, and the number of edges becomes N^2.

(g) Input layers for the feature

Add an MLP or linear layer for both the node and edge features.

d: hidden dimension

(h) Concatenate node features

Concatenate the node features from the MLP to those from the LPE.

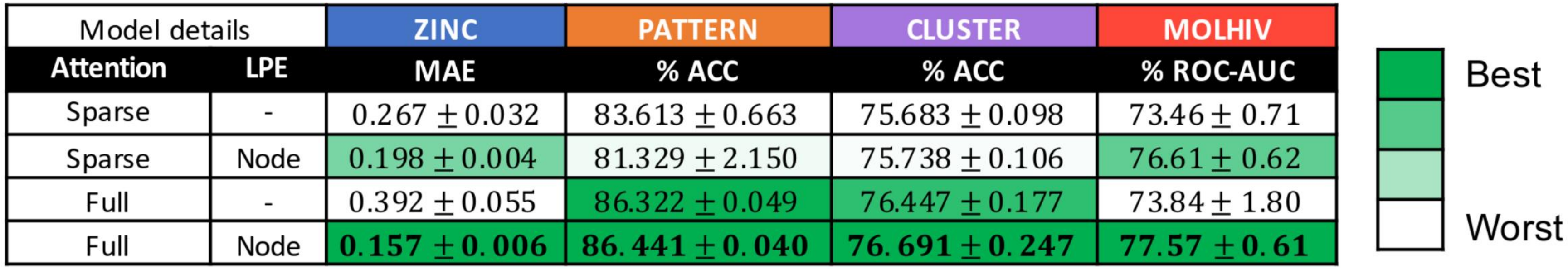
(i) Apply the main transformer

Attention between all pairs of nodes features and the edge between them. Different linear projections $\boldsymbol{K}, \boldsymbol{Q}, \boldsymbol{E}$ are used to compute attention for real edges and added edges.

Graph
$\boldsymbol{A}$
Node colormap
-max
0
max
$\boldsymbol{\phi}_0$
$\boldsymbol{\phi}_{m-1}$
j
N
λ_0
$\boldsymbol{\phi}_{0,j}$
λ_{m-1}
$\boldsymbol{\phi}_{m-1,j}$
Sequence length m
λ_1
$\boldsymbol{\phi}_{1,j}$
λ_2
$\boldsymbol{\phi}_{2,j}$
$N \times 2 \times m$
$2 \times k$
Linear
$m \times m$
Transformer
m
$N \times k \times m$
+
LPE
Number of features k
$N \times k$
Node features $\boldsymbol{X}^{(0)}$
$\boldsymbol{X}^{(0)}_{1,1}$ $\boldsymbol{X}^{(0)}_{1,2}$ $\boldsymbol{X}^{(0)}_{1,n_0}$
$\boldsymbol{X}^{(0)}_{2,1}$ $\boldsymbol{X}^{(0)}_{2,2}$ $\boldsymbol{X}^{(0)}_{2,n_0}$
$\boldsymbol{X}^{(0)}_{N,1}$ $\boldsymbol{X}^{(0)}_{N,2}$ $\boldsymbol{X}^{(0)}_{N,n_0}$
$N \times n_0$
Edge features $\boldsymbol{E}^{(0)}$
$\boldsymbol{E}^{(0)}_{1,1}$ $\boldsymbol{E}^{(0)}_{1,2}$ $\boldsymbol{E}^{(0)}_{1,e_0}$
$\boldsymbol{E}^{(0)}_{2,1}$ $\boldsymbol{E}^{(0)}_{2,2}$ $\boldsymbol{E}^{(0)}_{2,e_0}$
$\boldsymbol{E}^{(0)}_{E,1}$ $\boldsymbol{E}^{(0)}_{E,2}$ $\boldsymbol{E}^{(0)}_{E,e_0}$
$E \times e_0$
1
2
3
$N^2 \times (e_0 + 1)$
MLP
$O(N)$
$\boldsymbol{X}'^{(1)}$
$N \times (d - k)$
MLP
$O(N^2)$
$\boldsymbol{E}^{(1)}$
$N^2 \times d$
$\oplus$
$\boldsymbol{X}^{(1)}$
$N \times d$
$\boldsymbol{K}^2$
$\boldsymbol{K}^1$
$\boldsymbol{Q}^1$
$\boldsymbol{E}^1$
Transformer encoders
on the dimension of size l
$O(N^2)$
Prediction layer
Output
$N \times d$

Figure 1: The proposed SAN model with the node LPE, a generalization of Transformers to graphs.

Positional Encodings (with vs. without)



Model details		ZINC	PATTERN	CLUSTER	MOLHIV
Attention	LPE	MAE	% ACC	% ACC	% ROC-AUC
Sparse	-	0.267 ± 0.032	83.613 ± 0.663	75.683 ± 0.098	73.46 ± 0.71
Sparse	Node	0.198 ± 0.004	81.329 ± 2.150	75.738 ± 0.106	76.61 ± 0.62
Full	-	0.392 ± 0.055	86.322 ± 0.049	76.447 ± 0.177	73.84 ± 1.80
Full	Node	**0.157 ± 0.006**	**86.441 ± 0.040**	**76.691 ± 0.247**	**77.57 ± 0.61**

Figure 6: Ablation study on datasets from [15, 21] for the node LPE and full graph attention, with no hyperparameter tuning other than γ taken from Figure 5. For a given dataset, all models use the same hyperparameters, but the hidden dimensions are adjusted to have $\sim 500k$ learnable parameters. Means and uncertainties are derived from four runs, with different seeds (except MolHIV).

Graphormer

[Ying et al., 2021]

- Two key new ideas
- Centrality encoding

$$h_i^{(0)} = x_i + z^-_{\deg^-(v_i)} + z^+_{\deg^+(v_i)},$$

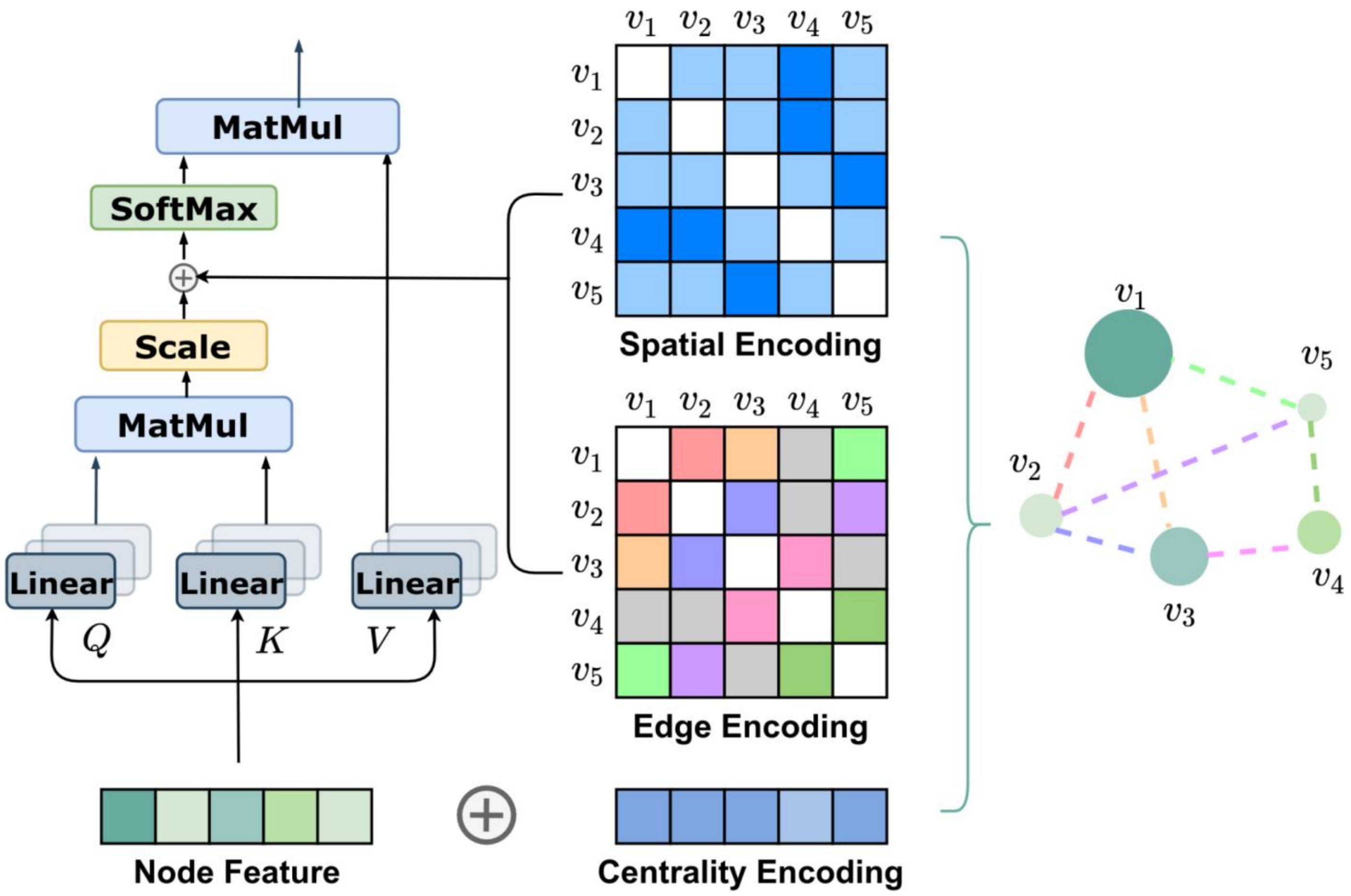
- Spatial encoding (distance-based attention bias)

Shortest path distance

$$A_{ij} = \underbrace{\frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d}}}_{\text{Standard attention computation}} + \underbrace{b_{\phi(v_i, v_j)}}_{\text{Attention bias}},$$

Graphormer

[Ying et al., 2021]



Graphormer: Molecular Graphs

[Ying et al., 2021]

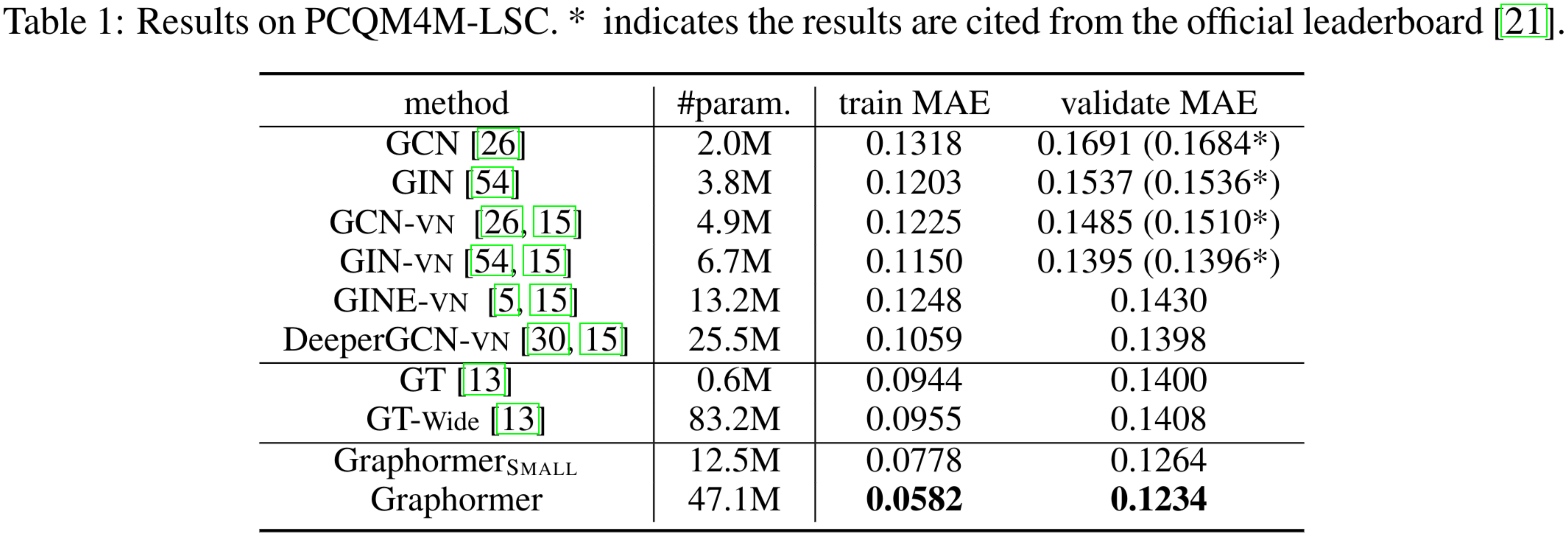


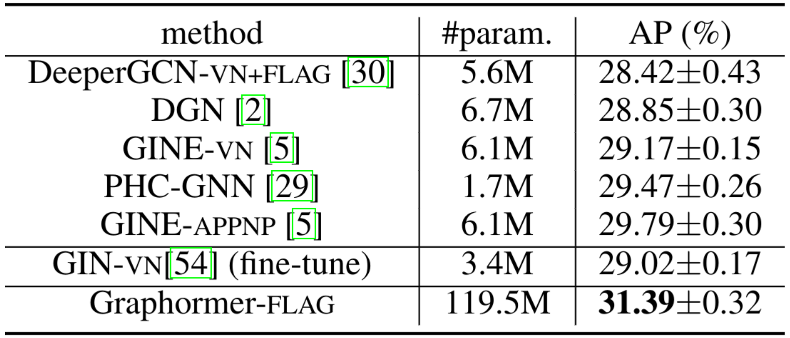
Table 1: Results on PCQM4M-LSC. * indicates the results are cited from the official leaderboard [21].

method	#param.	train MAE	validate MAE
GCN [26]	2.0M	0.1318	0.1691 (0.1684*)
GIN [54]	3.8M	0.1203	0.1537 (0.1536*)
GCN-VN [26, 15]	4.9M	0.1225	0.1485 (0.1510*)
GIN-VN [54, 15]	6.7M	0.1150	0.1395 (0.1396*)
GINE-VN [5, 15]	13.2M	0.1248	0.1430
DeeperGCN-VN [30, 15]	25.5M	0.1059	0.1398
GT [13]	0.6M	0.0944	0.1400
GT-Wide [13]	83.2M	0.0955	0.1408
Graphormer$_{\text{SMALL}}$	12.5M	0.0778	0.1264
Graphormer	47.1M	**0.0582**	**0.1234**

Graphormer: Molecular Graphs

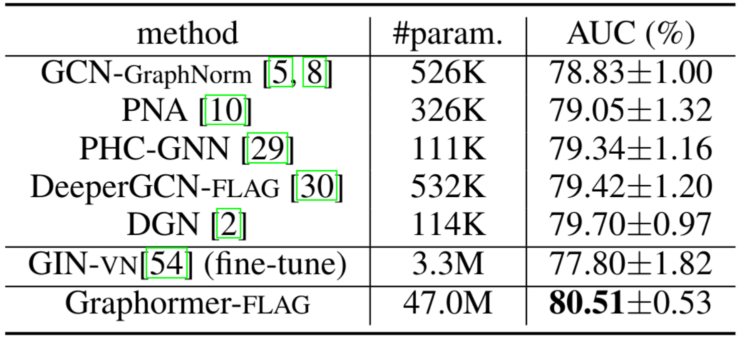
[Ying et al., 2021]

Table 2: Results on MolPCBA.



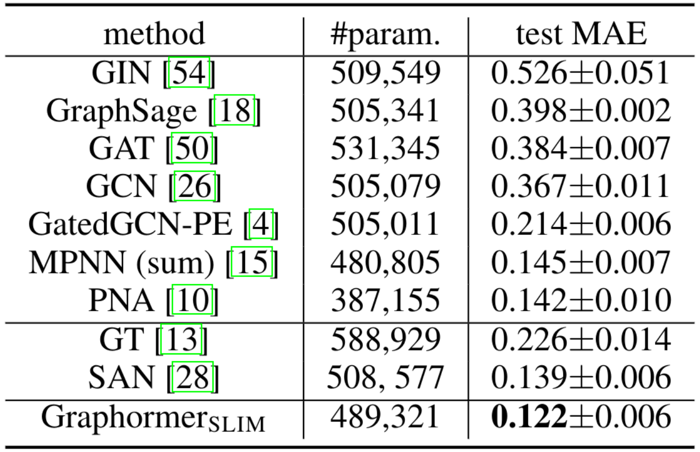
method	#param.	AP (%)
DeeperGCN-VN+FLAG [30]	5.6M	28.42±0.43
DGN [2]	6.7M	28.85±0.30
GINE-VN [5]	6.1M	29.17±0.15
PHC-GNN [29]	1.7M	29.47±0.26
GINE-APPNP [5]	6.1M	29.79±0.30
GIN-VN[54] (fine-tune)	3.4M	29.02±0.17
Graphormer-FLAG	119.5M	**31.39**±0.32

Table 3: Results on MolHIV.



method	#param.	AUC (%)
GCN-GraphNorm [5, 8]	526K	78.83±1.00
PNA [10]	326K	79.05±1.32
PHC-GNN [29]	111K	79.34±1.16
DeeperGCN-FLAG [30]	532K	79.42±1.20
DGN [2]	114K	79.70±0.97
GIN-VN[54] (fine-tune)	3.3M	77.80±1.82
Graphormer-FLAG	47.0M	**80.51**±0.53

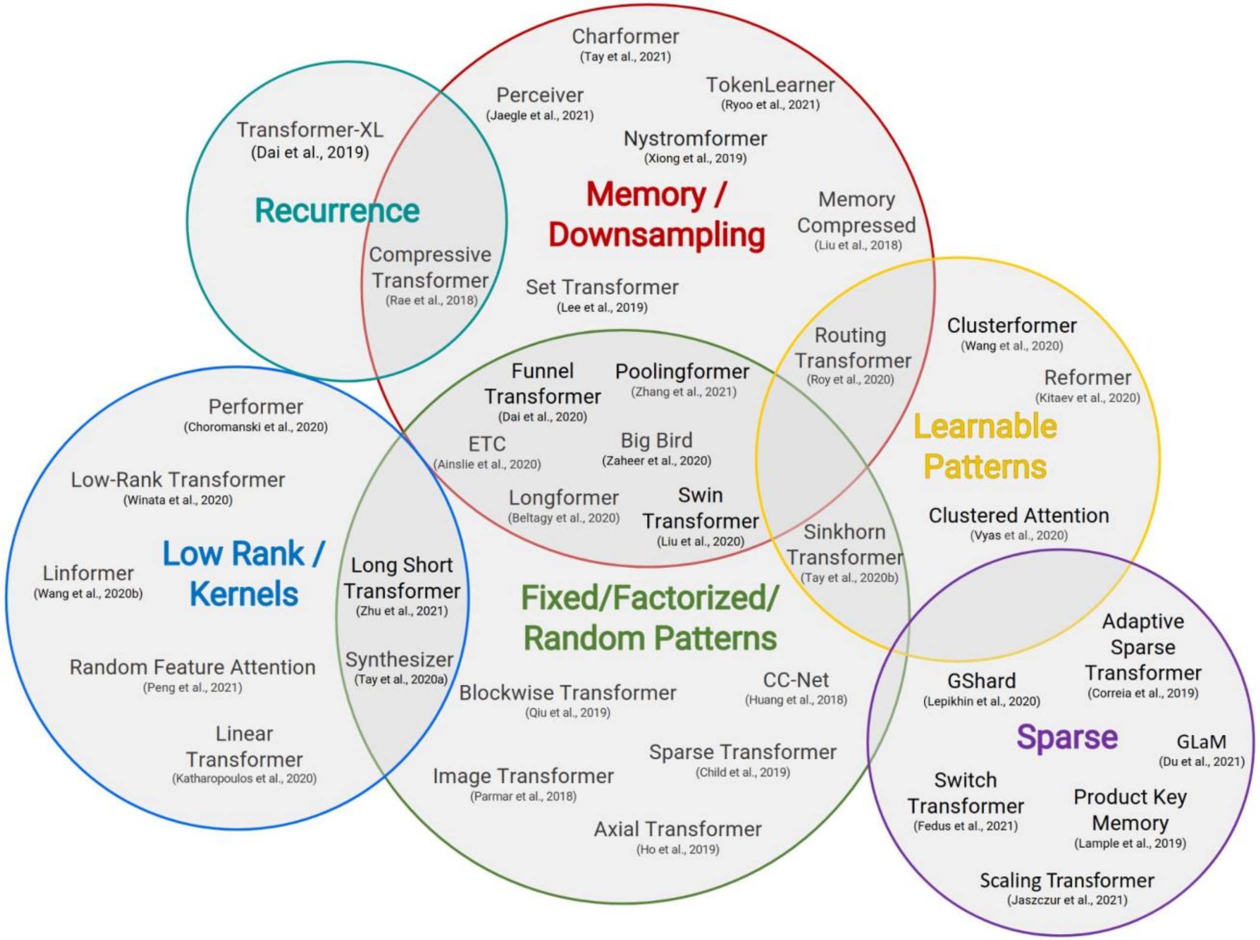
Table 4: Results on ZINC.



method	#param.	test MAE
GIN [54]	509,549	0.526±0.051
GraphSage [18]	505,341	0.398±0.002
GAT [50]	531,345	0.384±0.007
GCN [26]	505,079	0.367±0.011
GatedGCN-PE [4]	505,011	0.214±0.006
MPNN (sum) [15]	480,805	0.145±0.007
PNA [10]	387,155	0.142±0.010
GT [13]	588,929	0.226±0.014
SAN [28]	508, 577	0.139±0.006
Graphormer$_{\text{SLIM}}$	489,321	**0.122**±0.006

Scaling

- Typical transformer has O(N^2) dependence – prohibitive for long sequences or large graphs
- Dense attention ⟶ **sparse attention**
  - Still maintain good global connectivity
  - Efficient computation: O(N+M) interaction pairs
- Many sparse attention mechanisms proposed for *sequence* transformers



Charformer (Tay et al., 2021)
Perceiver (Jaegle et al., 2021)
TokenLearner (Ryoo et al., 2021)
Transformer-XL (Dai et al., 2019)
Nystromformer (Xiong et al., 2019)
Recurrence
Memory / Downsampling
Memory Compressed (Liu et al., 2018)
Compressive Transformer (Rae et al., 2018)
Set Transformer (Lee et al., 2019)
Routing Transformer (Roy et al., 2020)
Clusterformer (Wang et al., 2020)
Funnel Transformer (Dai et al., 2020)
Poolingformer (Zhang et al., 2021)
Reformer (Kitaev et al., 2020)
Performer (Choromanski et al., 2020)
ETC (Ainslie et al., 2020)
Big Bird (Zaheer et al., 2020)
Learnable Patterns
Low-Rank Transformer (Winata et al., 2020)
Longformer (Beltagy et al., 2020)
Swin Transformer (Liu et al., 2020)
Clustered Attention (Vyas et al., 2020)
Sinkhorn Transformer (Tay et al., 2020b)
Linformer (Wang et al., 2020b)
Low Rank / Kernels
Long Short Transformer (Zhu et al., 2021)
Fixed/Factorized/ Random Patterns
Adaptive Sparse Transformer (Correia et al., 2019)
Random Feature Attention (Peng et al., 2021)
Synthesizer (Tay et al., 2020a)
Blockwise Transformer (Qiu et al., 2019)
CC-Net (Huang et al., 2018)
GShard (Lepikhin et al., 2020)
Sparse
GLaM (Du et al., 2021)
Linear Transformer (Katharopoulos et al., 2020)
Sparse Transformer (Child et al., 2019)
Image Transformer (Parmar et al., 2018)
Switch Transformer (Fedus et al., 2021)
Product Key Memory (Lample et al., 2019)
Axial Transformer (Ho et al., 2019)
Scaling Transformer (Jaszczur et al., 2021)

Y. Tay, M. Dehghani, D. Bahri, and D. Metzler. Efficient Transformers: A Survey. ACM Computing Survey, volume 55. 2022

Sparse Transformers for *Graphs*

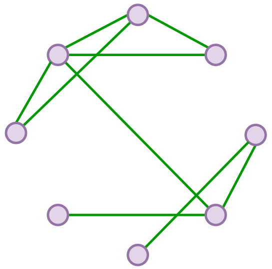
- Exphormer ([Shirzad et al., 2023]) – similar to Big Bird ([Zaheer et al., 2020])
- Nodeformer ([Wu et al., 2023)) – inspired by Performer ([Choromanski et al., 2021]), uses kernelized Gumbel-Softmax operator
- Sampling-based: Gophormer ([Zhao et al., 2021]), NAGphormer ([Chen et al., 2022])
- Diffusion-based: Difformer ([Wu et al., 2023])
- Spectral filtering: Specformer ([Bo et al., 2023])
- And many more…

Exphormer

[Shirzad, Velingker, Venkatachalam, Sutherland, Sinop – ICML 2023]

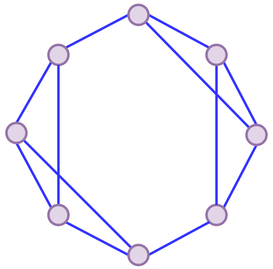
Original Graph

- Preserve locality from the original graph



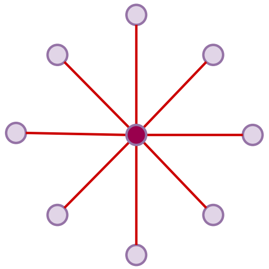
Expander Graph

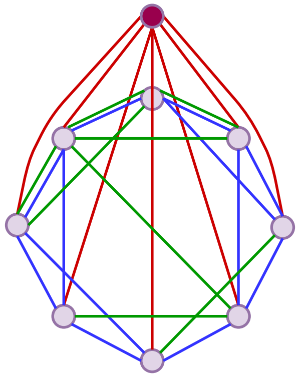
- Random walk mixing
- Constant degree, O(N) edges



Global Sink

- "Storage sink"
- Short connections between all node pairs





Exphormer: Combine all three to form the interaction graph!

Exphormer: Experimental Results

[Shirzad, Velingker, Venkatachalam, Sutherland, Sinop – ICML 2023]

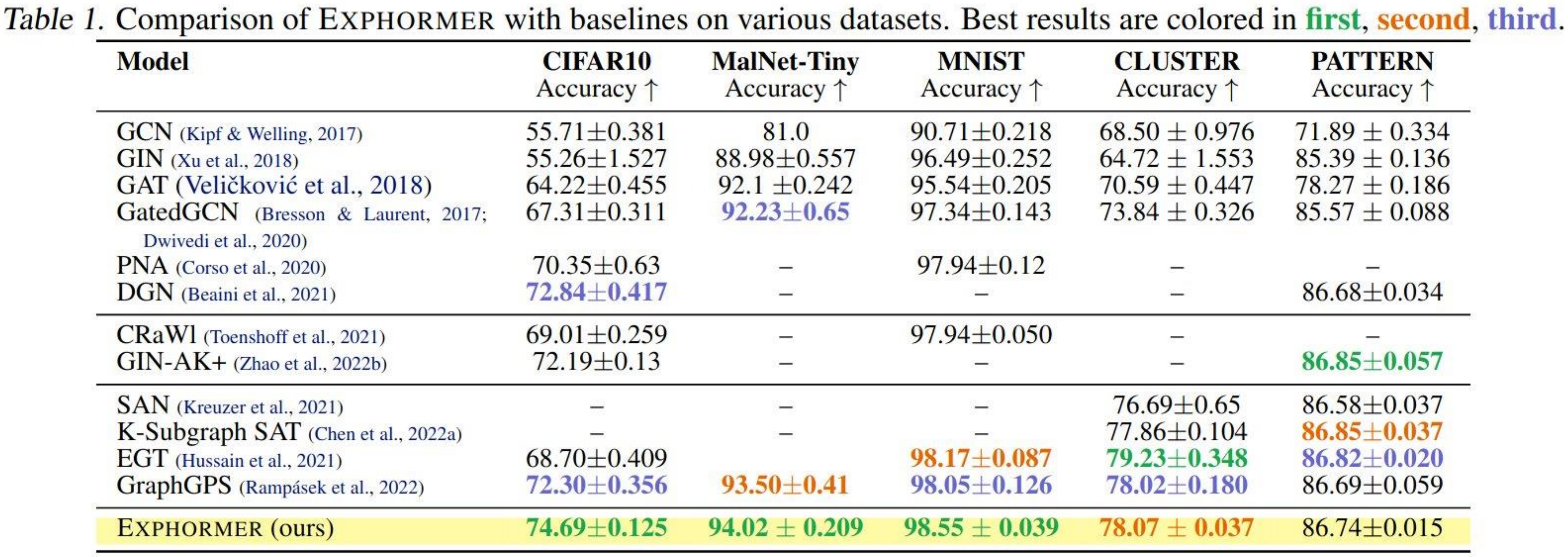


Table 1. Comparison of EXPHORMER with baselines on various datasets. Best results are colored in **first**, **second**, **third**.

Model	CIFAR10 Accuracy ↑	MalNet-Tiny Accuracy ↑	MNIST Accuracy ↑	CLUSTER Accuracy ↑	PATTERN Accuracy ↑
GCN (Kipf & Welling, 2017)	55.71±0.381	81.0	90.71±0.218	68.50 ± 0.976	71.89 ± 0.334
GIN (Xu et al., 2018)	55.26±1.527	88.98±0.557	96.49±0.252	64.72 ± 1.553	85.39 ± 0.136
GAT (Veličković et al., 2018)	64.22±0.455	92.1 ±0.242	95.54±0.205	70.59 ± 0.447	78.27 ± 0.186
GatedGCN (Bresson & Laurent, 2017; Dwivedi et al., 2020)	67.31±0.311	**92.23±0.65**	97.34±0.143	73.84 ± 0.326	85.57 ± 0.088
PNA (Corso et al., 2020)	70.35±0.63	–	97.94±0.12	–	–
DGN (Beaini et al., 2021)	**72.84±0.417**	–	–	–	86.68±0.034
CRaWl (Toenshoff et al., 2021)	69.01±0.259	–	97.94±0.050	–	–
GIN-AK+ (Zhao et al., 2022b)	72.19±0.13	–	–	–	**86.85±0.057**
SAN (Kreuzer et al., 2021)	–	–	–	76.69±0.65	86.58±0.037
K-Subgraph SAT (Chen et al., 2022a)	–	–	–	77.86±0.104	**86.85±0.037**
EGT (Hussain et al., 2021)	68.70±0.409		**98.17±0.087**	**79.23±0.348**	**86.82±0.020**
GraphGPS (Rampásek et al., 2022)	**72.30±0.356**	**93.50±0.41**	**98.05±0.126**	**78.02±0.180**	86.69±0.059
EXPHORMER (ours)	**74.69±0.125**	**94.02 ± 0.209**	**98.55 ± 0.039**	**78.07 ± 0.037**	86.74±0.015

Exphormer: Long-Range Benchmark

[Shirzad, Velingker, Venkatachalam, Sutherland, Sinop – ICML 2023]

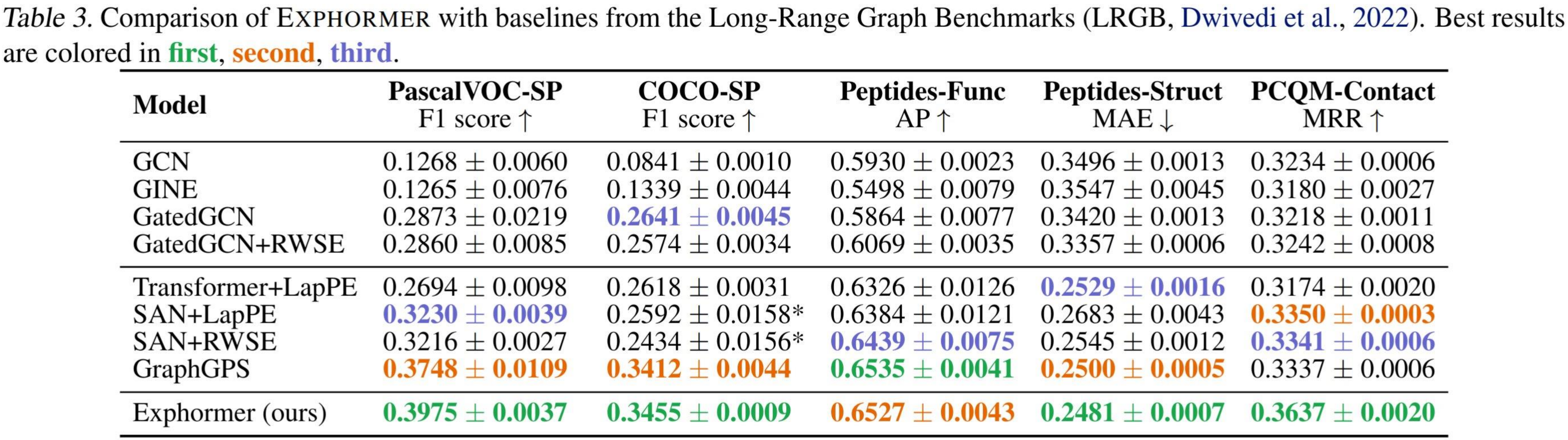


Table 3. Comparison of EXPHORMER with baselines from the Long-Range Graph Benchmarks (LRGB, Dwivedi et al., 2022). Best results are colored in **first**, **second**, **third**.

Model	**PascalVOC-SP** F1 score ↑	**COCO-SP** F1 score ↑	**Peptides-Func** AP ↑	**Peptides-Struct** MAE ↓	**PCQM-Contact** MRR ↑
GCN	0.1268 ± 0.0060	0.0841 ± 0.0010	0.5930 ± 0.0023	0.3496 ± 0.0013	0.3234 ± 0.0006
GINE	0.1265 ± 0.0076	0.1339 ± 0.0044	0.5498 ± 0.0079	0.3547 ± 0.0045	0.3180 ± 0.0027
GatedGCN	0.2873 ± 0.0219	**0.2641 ± 0.0045**	0.5864 ± 0.0077	0.3420 ± 0.0013	0.3218 ± 0.0011
GatedGCN+RWSE	0.2860 ± 0.0085	0.2574 ± 0.0034	0.6069 ± 0.0035	0.3357 ± 0.0006	0.3242 ± 0.0008
Transformer+LapPE	0.2694 ± 0.0098	0.2618 ± 0.0031	0.6326 ± 0.0126	**0.2529 ± 0.0016**	0.3174 ± 0.0020
SAN+LapPE	**0.3230 ± 0.0039**	0.2592 ± 0.0158*	0.6384 ± 0.0121	0.2683 ± 0.0043	**0.3350 ± 0.0003**
SAN+RWSE	0.3216 ± 0.0027	0.2434 ± 0.0156*	**0.6439 ± 0.0075**	0.2545 ± 0.0012	**0.3341 ± 0.0006**
GraphGPS	**0.3748 ± 0.0109**	**0.3412 ± 0.0044**	**0.6535 ± 0.0041**	**0.2500 ± 0.0005**	0.3337 ± 0.0006
Exphormer (ours)	**0.3975 ± 0.0037**	**0.3455 ± 0.0009**	**0.6527 ± 0.0043**	**0.2481 ± 0.0007**	**0.3637 ± 0.0020**

Message Passing vs. Graph Transformers

Message Passing

Updates across edges of input graph

✅ Captures inductive bias from input graph topology

✅ Efficient computation: O(N + M)

❌ Difficulty with long-range dependencies

❌ Oversmoothing, oversquashing

❌ Expressivity limitations

Graph Transformers

Use global attention

✅ Computation graph can be different from input graph

✅ Long-range modeling

Positional and structural encodings

❌ Identifying nodes within graph

❌ Loss of inductive bias from graph

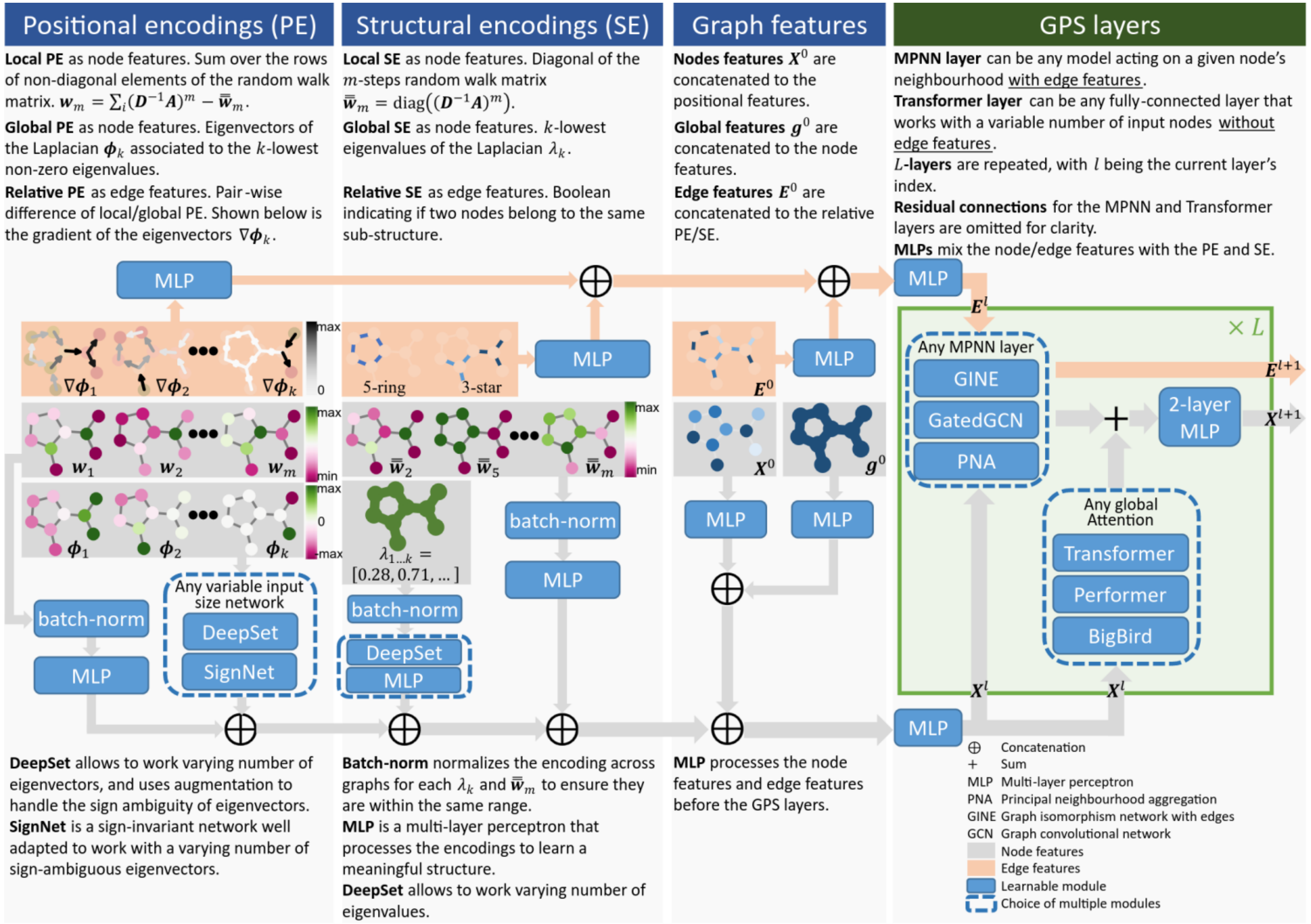
❌ Inefficient computation: $O(N^2)$

Graph-oriented sparse attention schemes

GraphGPS

[Rampášek et al., 2022]

- Combine transformers with message-passing
- Transformers give added expressivity while message-passing retains input graph structure
- Framework – mix and match MPNN layers, attention layers, and positional/structural encodings



Positional encodings (PE)

Local PE as node features. Sum over the rows of non-diagonal elements of the random walk matrix. $\boldsymbol{w}_m = \sum_i (\boldsymbol{D}^{-1}\boldsymbol{A})^m - \bar{\bar{\boldsymbol{w}}}_m$.

Global PE as node features. Eigenvectors of the Laplacian $\boldsymbol{\phi}_k$ associated to the k-lowest non-zero eigenvalues.

Relative PE as edge features. Pair -wise difference of local/global PE. Shown below is the gradient of the eigenvectors $\nabla\boldsymbol{\phi}_k$.

DeepSet allows to work varying number of eigenvectors, and uses augmentation to handle the sign ambiguity of eigenvectors.

SignNet is a sign-invariant network well adapted to work with a varying number of sign-ambiguous eigenvectors.

Structural encodings (SE)

Local SE as node features. Diagonal of the m-steps random walk matrix $\bar{\bar{\boldsymbol{w}}}_m = \mathrm{diag}\big((\boldsymbol{D}^{-1}\boldsymbol{A})^m\big)$.

Global SE as node features. k-lowest eigenvalues of the Laplacian λ_k.

Relative SE as edge features. Boolean indicating if two nodes belong to the same sub-structure.

Batch-norm normalizes the encoding across graphs for each λ_k and $\bar{\bar{\boldsymbol{w}}}_m$ to ensure they are within the same range.

MLP is a multi-layer perceptron that processes the encodings to learn a meaningful structure.

DeepSet allows to work varying number of eigenvalues.

Graph features

Nodes features $\boldsymbol{X}^0$ are concatenated to the positional features.

Global features $\boldsymbol{g}^0$ are concatenated to the node features.

Edge features $\boldsymbol{E}^0$ are concatenated to the relative PE/SE.

MLP processes the node features and edge features before the GPS layers.

GPS layers

MPNN layer can be any model acting on a given node's neighbourhood with edge features.

Transformer layer can be any fully-connected layer that works with a variable number of input nodes without edge features.

L**-layers** are repeated, with l being the current layer's index.

Residual connections for the MPNN and Transformer layers are omitted for clarity.

MLPs mix the node/edge features with the PE and SE.

⊕	Concatenation
+	Sum
MLP	Multi-layer perceptron
PNA	Principal neighbourhood aggregation
GINE	Graph isomorphism network with edges
GCN	Graph convolutional network
▨	Node features
▨	Edge features
▭	Learnable module
⬚	Choice of multiple modules

Figure 1: Modular GPS graph Transformer, with examples of PE and SE. Task specific layers for node/graph/edge-level predictions, such as pooling or output MLP, are omitted for simplicity.

EXPRESSIVITY

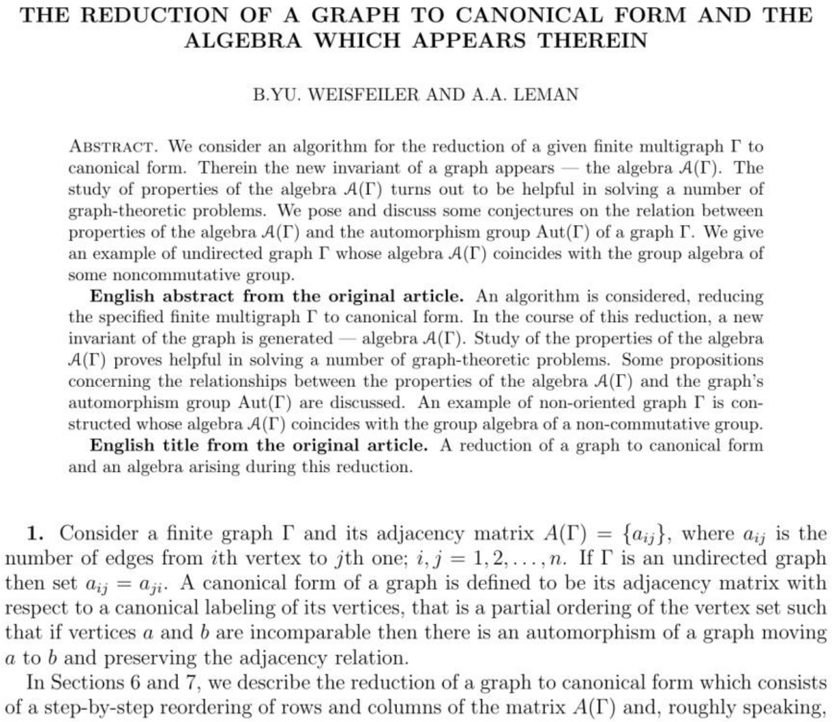
Expressivity of GNNs

- GNN architectures can represent some functions but not others

- What functions can a message-passing GNN represent?

Expressivity: WL Isomorphism Test

- WL test was proposed in 1968 as a heuristic for the existence of an isomorphism between two graphs
- Relation to result of [Babai, 2015]
- GNNs known to be bounded in expressivity by the Weisfeiler-Leman (WL) test ([Morris et al., 2019], [Xu et al., 2019])
- 1-WL test: Hash aggregated color multisets of neighbors at each step



THE REDUCTION OF A GRAPH TO CANONICAL FORM AND THE ALGEBRA WHICH APPEARS THEREIN

B.YU. WEISFEILER AND A.A. LEMAN

ABSTRACT. We consider an algorithm for the reduction of a given finite multigraph Γ to canonical form. Therein the new invariant of a graph appears — the algebra $\mathcal{A}(\Gamma)$. The study of properties of the algebra $\mathcal{A}(\Gamma)$ turns out to be helpful in solving a number of graph-theoretic problems. We pose and discuss some conjectures on the relation between properties of the algebra $\mathcal{A}(\Gamma)$ and the automorphism group Aut(Γ) of a graph Γ. We give an example of undirected graph Γ whose algebra $\mathcal{A}(\Gamma)$ coincides with the group algebra of some noncommutative group.

English abstract from the original article. An algorithm is considered, reducing the specified finite multigraph Γ to canonical form. In the course of this reduction, a new invariant of the graph is generated — algebra $\mathcal{A}(\Gamma)$. Study of the properties of the algebra $\mathcal{A}(\Gamma)$ proves helpful in solving a number of graph-theoretic problems. Some propositions concerning the relationships between the properties of the algebra $\mathcal{A}(\Gamma)$ and the graph's automorphism group Aut(Γ) are discussed. An example of non-oriented graph Γ is constructed whose algebra $\mathcal{A}(\Gamma)$ coincides with the group algebra of a non-commutative group.

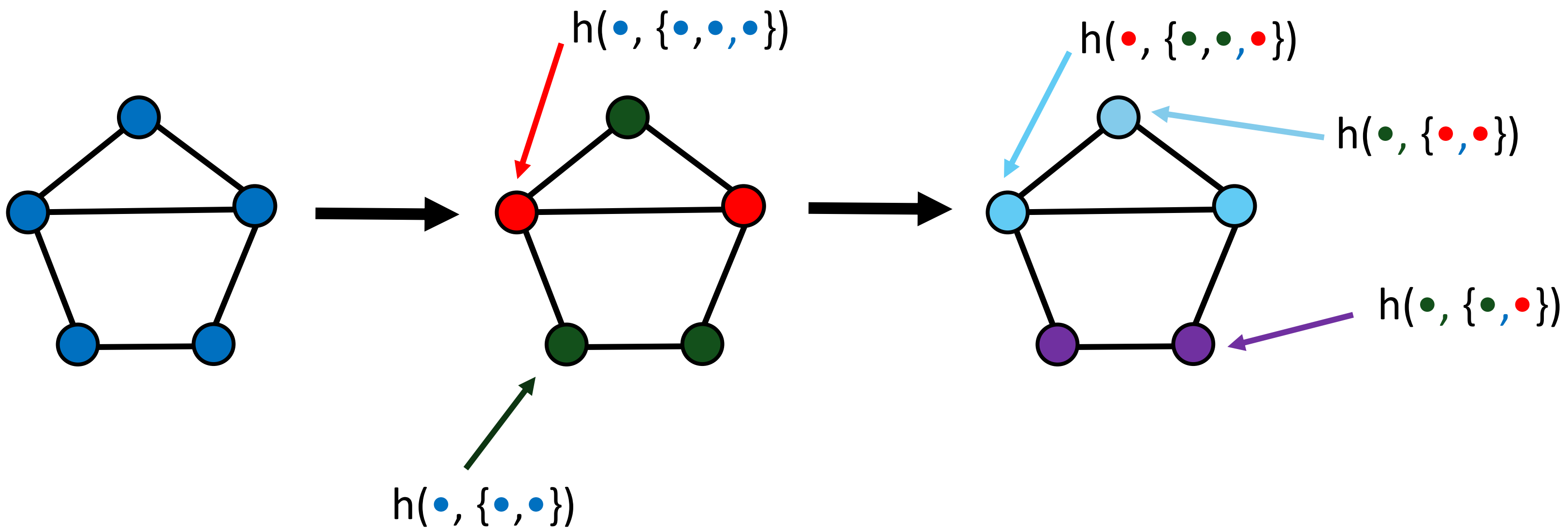
English title from the original article. A reduction of a graph to canonical form and an algebra arising during this reduction.

1. Consider a finite graph Γ and its adjacency matrix $A(\Gamma) = \{a_{ij}\}$, where a_{ij} is the number of edges from ith vertex to jth one; $i, j = 1, 2, \ldots, n$. If Γ is an undirected graph then set $a_{ij} = a_{ji}$. A canonical form of a graph is defined to be its adjacency matrix with respect to a canonical labeling of its vertices, that is a partial ordering of the vertex set such that if vertices a and b are incomparable then there is an automorphism of a graph moving a to b and preserving the adjacency relation.

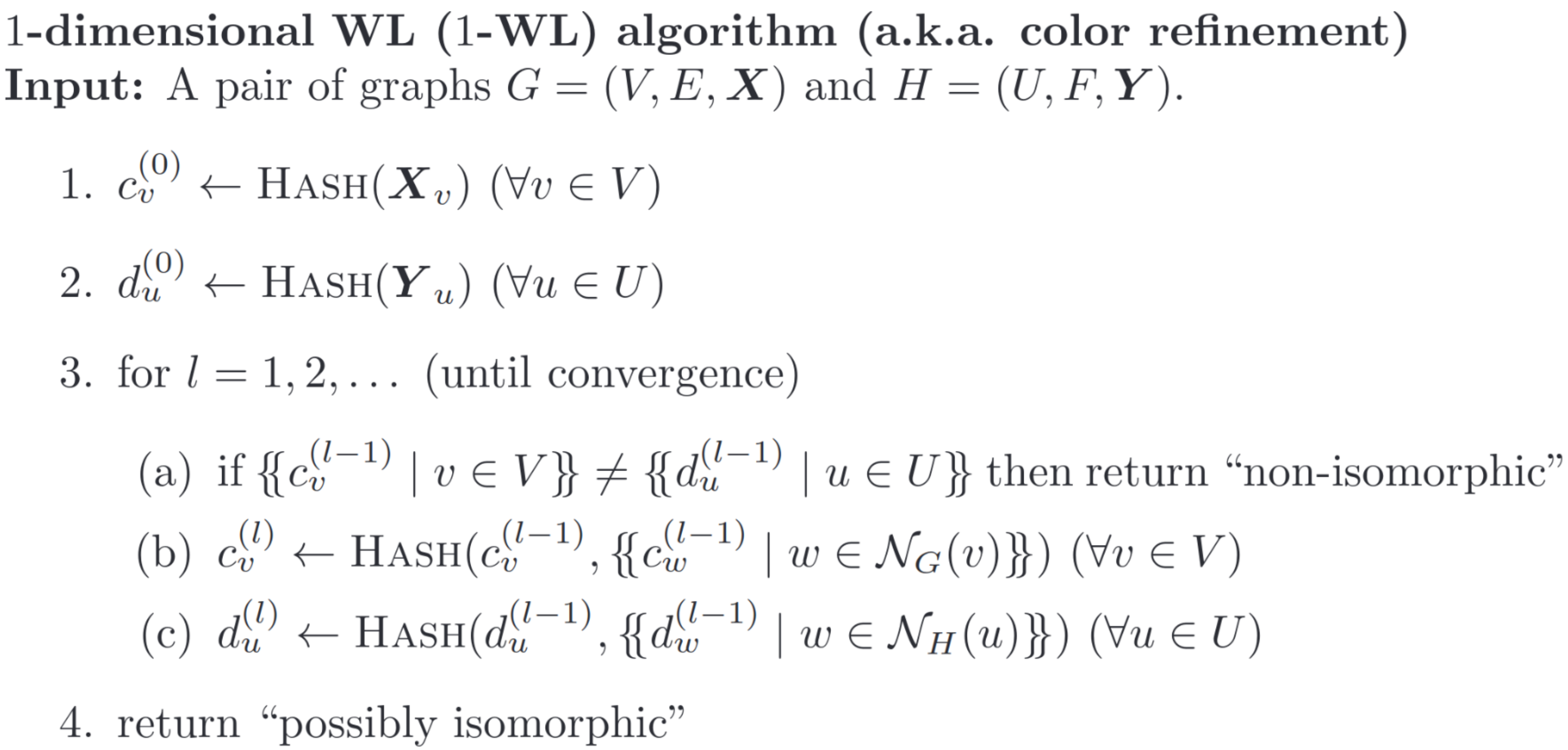
In Sections 6 and 7, we describe the reduction of a graph to canonical form which consists of a step-by-step reordering of rows and columns of the matrix $A(\Gamma)$ and, roughly speaking,

1-WL Isomorphism Test in Action

- 1-WL test: Hash aggregated color multisets of neighbors at each step
- Check if node color multisets of two given graphs match



1-WL Algorithm

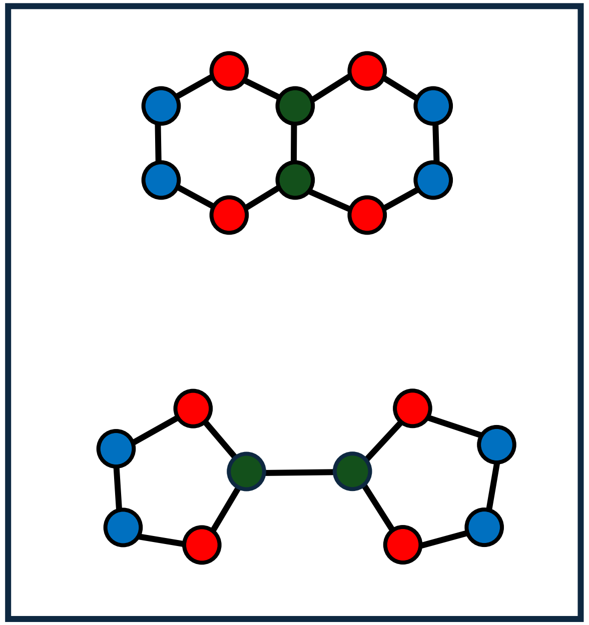


1-dimensional WL (1-WL) algorithm (a.k.a. color refinement)
Input: A pair of graphs $G = (V, E, \boldsymbol{X})$ and $H = (U, F, \boldsymbol{Y})$.

```
1. $c_v^{(0)} \leftarrow \textsc{Hash}(\boldsymbol{X}_v)$ $(\forall v \in V)$
2. $d_u^{(0)} \leftarrow \textsc{Hash}(\boldsymbol{Y}_u)$ $(\forall u \in U)$
3. for $l = 1, 2, \ldots$ (until convergence)
    (a) if $\{\{c_v^{(l-1)} \mid v \in V\}\} \neq \{\{d_u^{(l-1)} \mid u \in U\}\}$ then return "non-isomorphic"
    (b) $c_v^{(l)} \leftarrow \textsc{Hash}(c_v^{(l-1)}, \{\{c_w^{(l-1)} \mid w \in \mathcal{N}_G(v)\}\})$ $(\forall v \in V)$
    (c) $d_u^{(l)} \leftarrow \textsc{Hash}(d_u^{(l-1)}, \{\{d_w^{(l-1)} \mid w \in \mathcal{N}_H(u)\}\})$ $(\forall u \in U)$
4. return "possibly isomorphic"
```

1-WL Isomorphism Test

- Graph isomorphism is hard!
- 1-WL test fails to distinguish some pairs of non-isomorphic graphs
- A vanilla message-passing GNN also cannot distinguish such graphs



1-WL Limitations

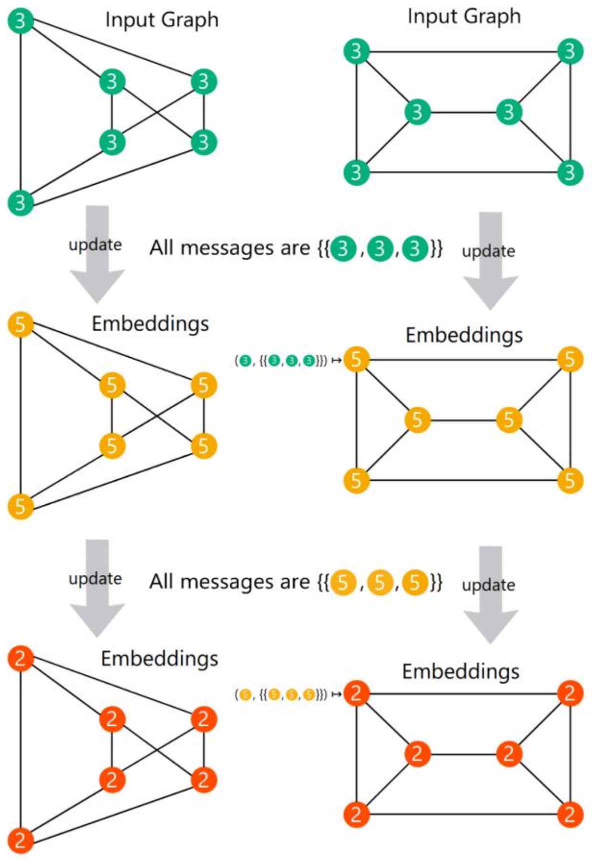


Figure 4: Message passing GNNs cannot distinguish any pair of regular graphs with the same degree and size even if they are not isomorphic.

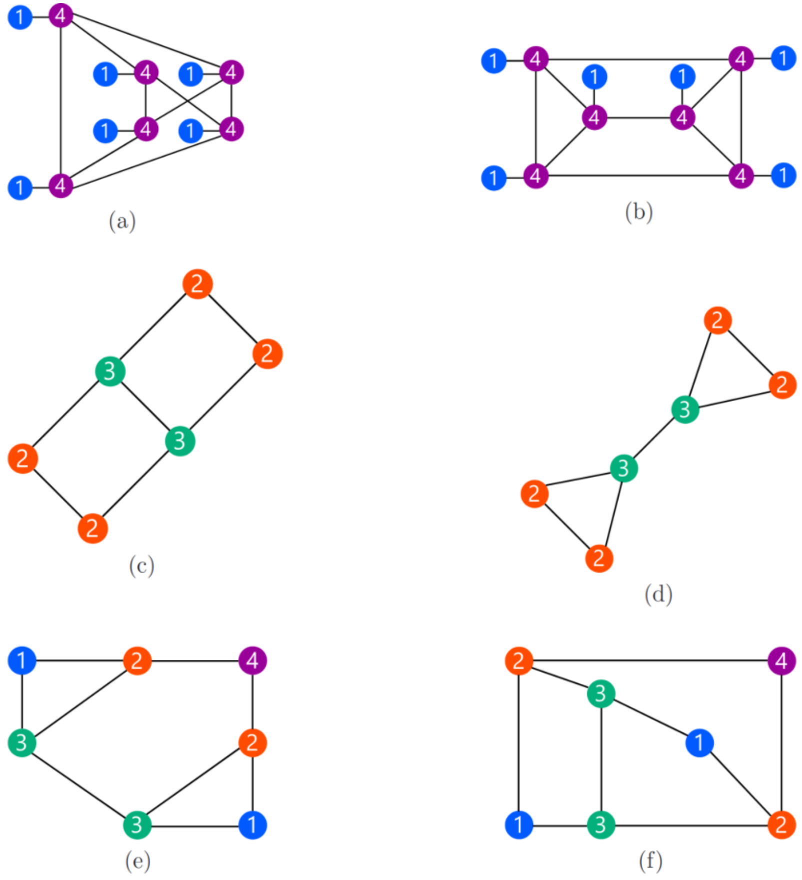
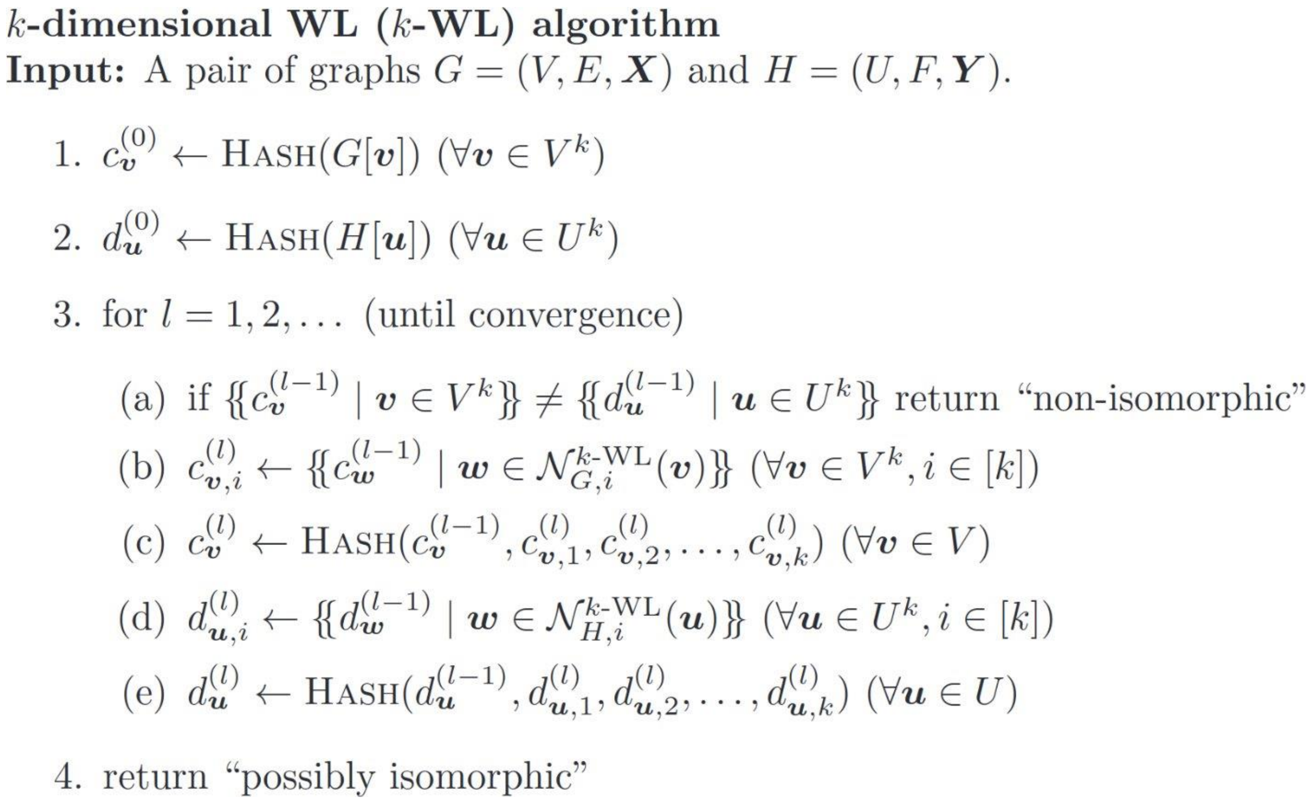


Figure 5: Although these graphs are not isomorphic or regular, GNNs cannot distinguish (a) from (b), (c) from (d), and (e) from (f)

k-WL Algorithm

- Assign colors to k-tuples of nodes



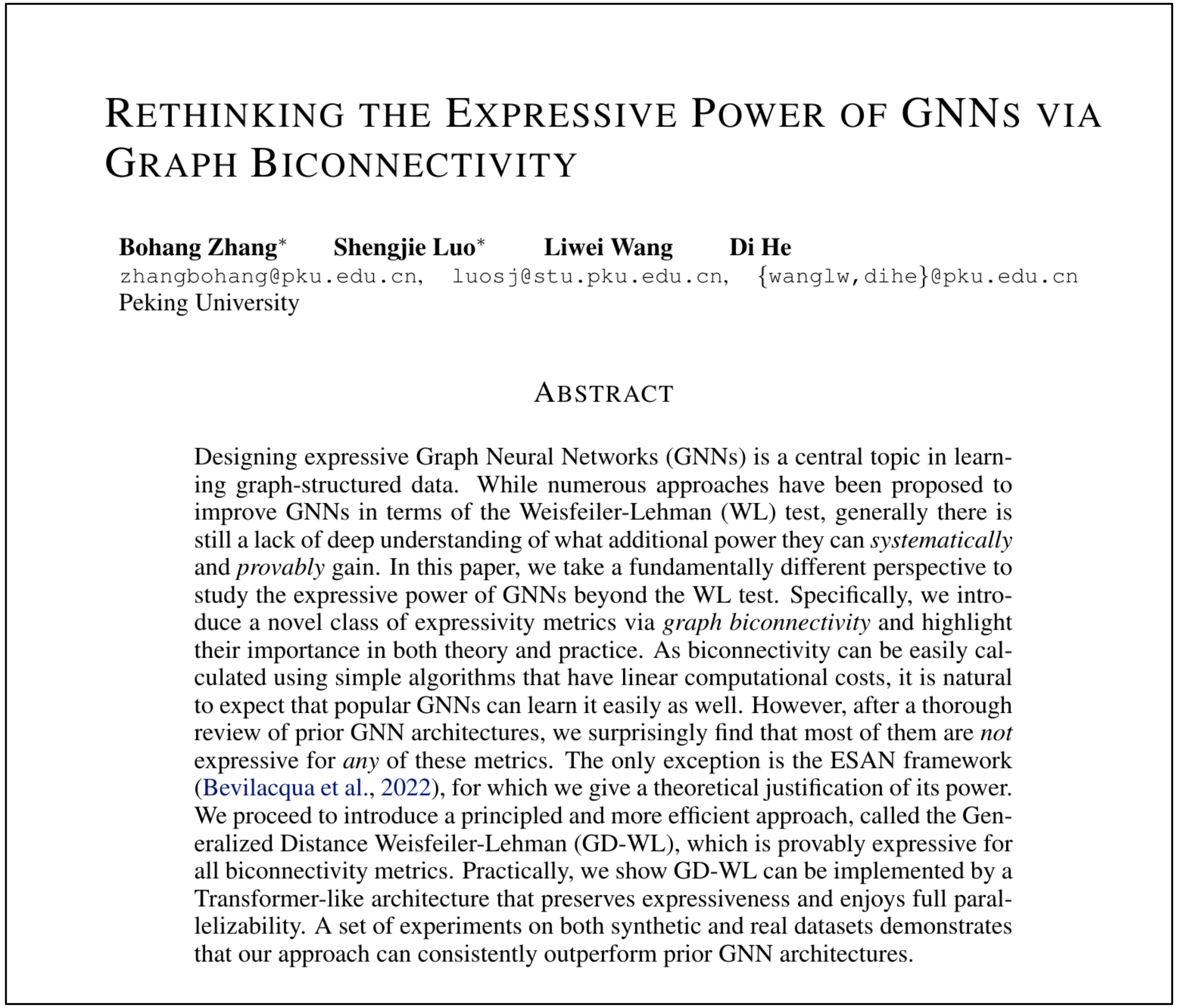
k-dimensional WL (k-WL) algorithm

Input: A pair of graphs $G = (V, E, \boldsymbol{X})$ and $H = (U, F, \boldsymbol{Y})$.

```
1. $c^{(0)}_{\boldsymbol{v}} \leftarrow \textsc{Hash}(G[\boldsymbol{v}])$ $(\forall \boldsymbol{v} \in V^k)$
2. $d^{(0)}_{\boldsymbol{u}} \leftarrow \textsc{Hash}(H[\boldsymbol{u}])$ $(\forall \boldsymbol{u} \in U^k)$
3. for $l = 1, 2, \ldots$ (until convergence)
    (a) if $\{\!\{c^{(l-1)}_{\boldsymbol{v}} \mid \boldsymbol{v} \in V^k\}\!\} \neq \{\!\{d^{(l-1)}_{\boldsymbol{u}} \mid \boldsymbol{u} \in U^k\}\!\}$ return "non-isomorphic"
    (b) $c^{(l)}_{\boldsymbol{v},i} \leftarrow \{\!\{c^{(l-1)}_{\boldsymbol{w}} \mid \boldsymbol{w} \in \mathcal{N}^{k\text{-WL}}_{G,i}(\boldsymbol{v})\}\!\}$ $(\forall \boldsymbol{v} \in V^k, i \in [k])$
    (c) $c^{(l)}_{\boldsymbol{v}} \leftarrow \textsc{Hash}(c^{(l-1)}_{\boldsymbol{v}}, c^{(l)}_{\boldsymbol{v},1}, c^{(l)}_{\boldsymbol{v},2}, \ldots, c^{(l)}_{\boldsymbol{v},k})$ $(\forall \boldsymbol{v} \in V)$
    (d) $d^{(l)}_{\boldsymbol{u},i} \leftarrow \{\!\{d^{(l-1)}_{\boldsymbol{w}} \mid \boldsymbol{w} \in \mathcal{N}^{k\text{-WL}}_{H,i}(\boldsymbol{u})\}\!\}$ $(\forall \boldsymbol{u} \in U^k, i \in [k])$
    (e) $d^{(l)}_{\boldsymbol{u}} \leftarrow \textsc{Hash}(d^{(l-1)}_{\boldsymbol{u}}, d^{(l)}_{\boldsymbol{u},1}, d^{(l)}_{\boldsymbol{u},2}, \ldots, d^{(l)}_{\boldsymbol{u},k})$ $(\forall \boldsymbol{u} \in U)$
4. return "possibly isomorphic"
```

Going Beyond WL Test?

- Going beyond WL by proposing expressivity metrics via *graph biconnectivity* ([Zhang et al., 2023])
- Generalized Distance WL (GD-WL)
- Uncovers limitations of many current GNN approaches



RETHINKING THE EXPRESSIVE POWER OF GNNS VIA GRAPH BICONNECTIVITY

Bohang Zhang* **Shengjie Luo*** **Liwei Wang** **Di He**
zhangbohang@pku.edu.cn, luosj@stu.pku.edu.cn, {wanglw,dihe}@pku.edu.cn
Peking University

ABSTRACT

Designing expressive Graph Neural Networks (GNNs) is a central topic in learning graph-structured data. While numerous approaches have been proposed to improve GNNs in terms of the Weisfeiler-Lehman (WL) test, generally there is still a lack of deep understanding of what additional power they can *systematically* and *provably* gain. In this paper, we take a fundamentally different perspective to study the expressive power of GNNs beyond the WL test. Specifically, we introduce a novel class of expressivity metrics via *graph biconnectivity* and highlight their importance in both theory and practice. As biconnectivity can be easily calculated using simple algorithms that have linear computational costs, it is natural to expect that popular GNNs can learn it easily as well. However, after a thorough review of prior GNN architectures, we surprisingly find that most of them are *not* expressive for *any* of these metrics. The only exception is the ESAN framework (Bevilacqua et al., 2022), for which we give a theoretical justification of its power. We proceed to introduce a principled and more efficient approach, called the Generalized Distance Weisfeiler-Lehman (GD-WL), which is provably expressive for all biconnectivity metrics. Practically, we show GD-WL can be implemented by a Transformer-like architecture that preserves expressiveness and enjoys full parallelizability. A set of experiments on both synthetic and real datasets demonstrates that our approach can consistently outperform prior GNN architectures.

How to Enhance Expressivity?

- Standard GNNs limited by 1-WL graph isomorphism test
- Ways to improve GNN expressivity
  - Add features
  - Modulate message-passing
  - Modify underlying graph

How to Enhance Expressivity?

- Standard GNNs limited by 1-WL graph isomorphism test
- Ways to improve GNN expressivity
  - **Add features**
  - Modulate message-passing
  - Modify underlying graph

Adding Features

- Add node, edge, or graph features that incorporate structural, positional, etc. information
- Often computed offline as a preprocessing step

Incorporate edge features

$$\mathbf{m}_{u,v}^{(t+1)} = M_t(\mathbf{h}_u^{(t)}, \mathbf{h}_v^{(t)}, \mathbf{e}_{u,v})$$

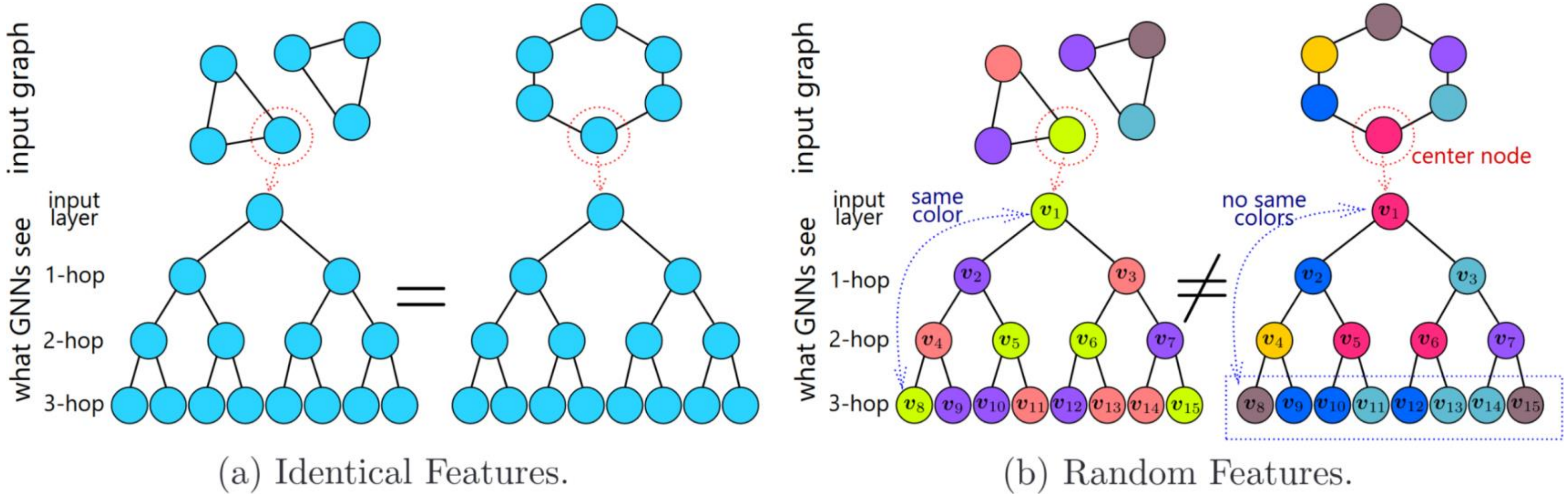
$$\mathbf{a}_u^{(t+1)} = f_{\text{aggregate}}(\{\!\{\mathbf{m}_{u,v}^{(t+1)} \mid v \in \mathcal{N}(u)\}\!\})$$

$$\mathbf{h}_u^{(t+1)} = f_{\text{update}}(\mathbf{h}_u^{(t)}, \mathbf{a}_u^{(t+1)})$$

- New node features can be added to $\mathbf{h}_u^{(0)}$
- Graph-level features can be incorporated in M_t and aggregation, update functions

Adding Features

- 1-WL limitation applies to very limited setting
- Inability to distinguish node identities
- <u>Simple tweak</u>: initializing nodes with random features goes beyond 1-WL ([Sato et al., 2021], [Abboud et al., 2021])

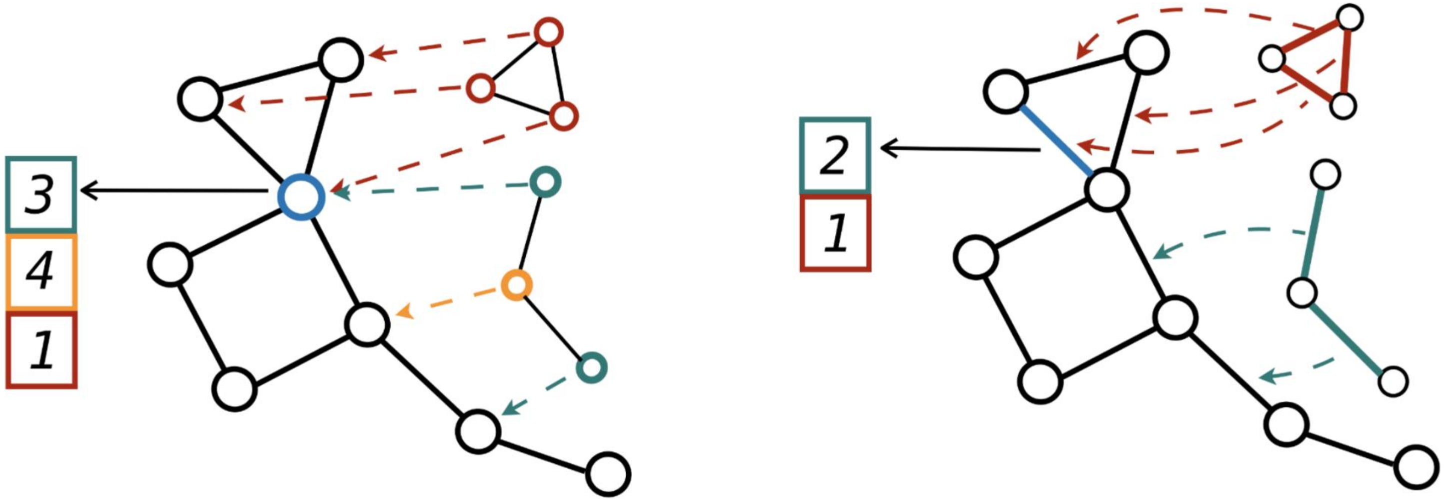


(a) Identical Features.

(b) Random Features.

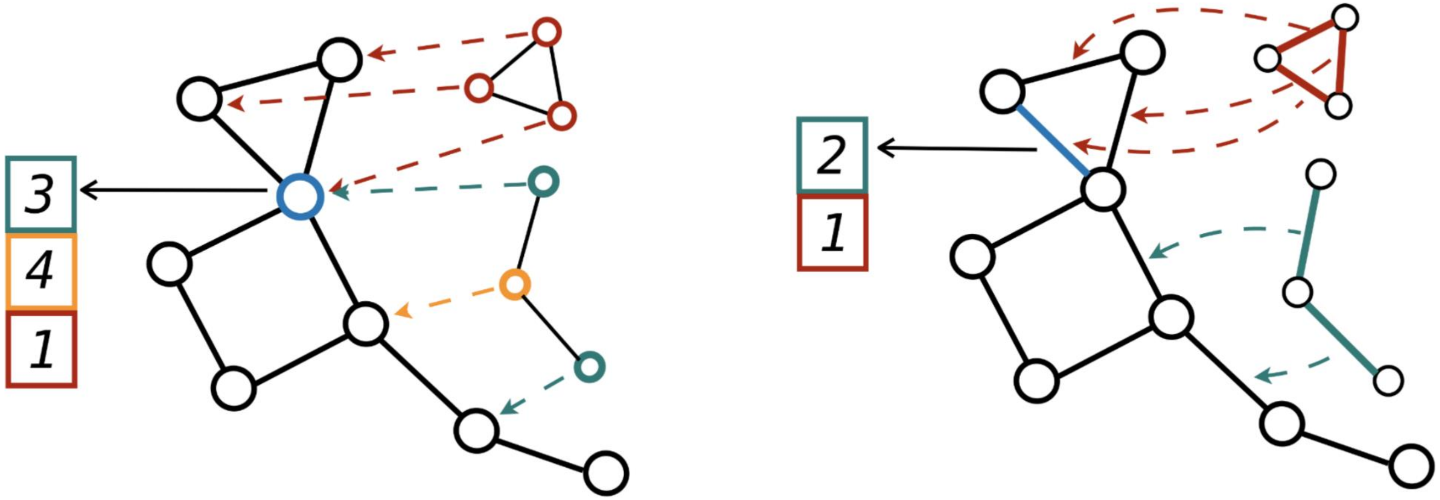
Graph Substructure Networks

- Features that encode topological structures/substructures
- Graph Substructure Networks (GSN) [Bouritsas et al., 2022]: Encode subgraph counts
  - Pick a set of graphs: $\{H_1, H_2, \ldots, H_K\}$
  - Node features: Count appearance of *v* in different orbits for each H_i
  - Edge features: Count appearance of *e* in different edge automorphism orbits



Graph Substructure Networks

- Graph Substructure Networks (GSN): Encode subgraph counts
  - Pick a set of graphs: $\{H_1, H_2, \ldots, H_K\}$ and encode node, edge orbits as features



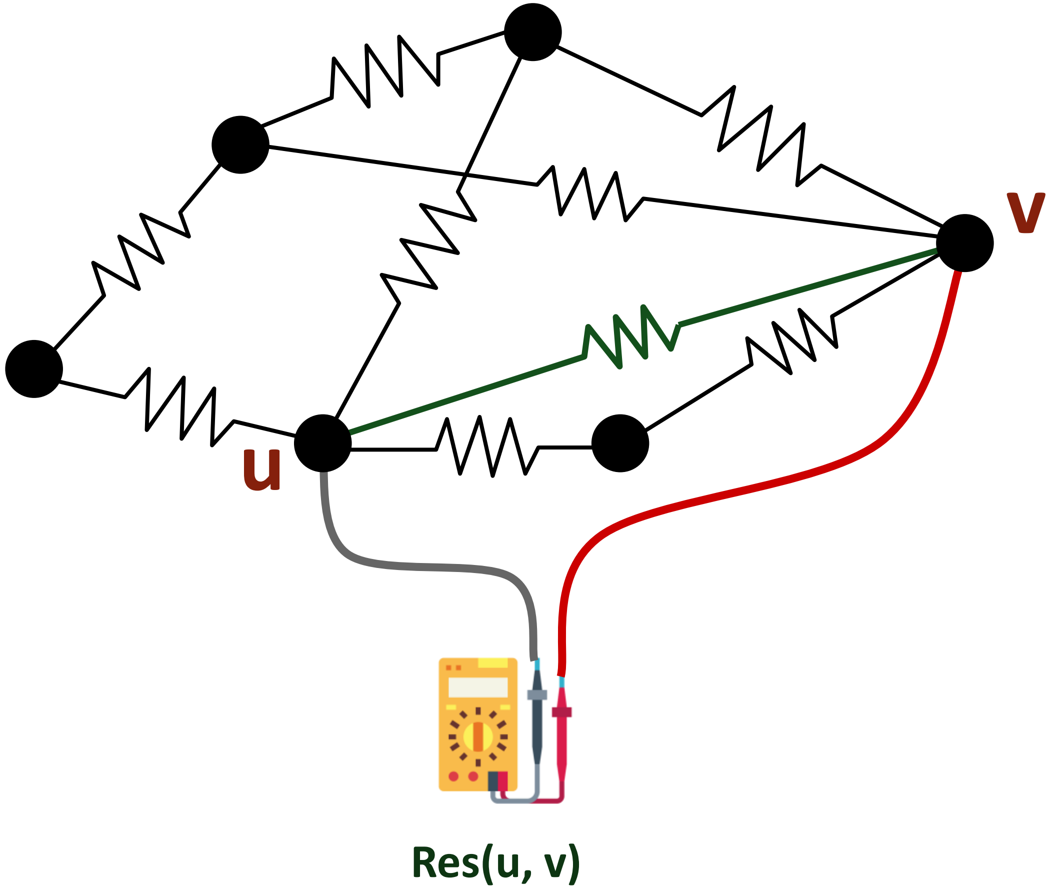
- Choice of substructures is domain-specific
- Chains and cycles in molecules

Adding Features

- GSN requires domain specific knowledge to know which substructures to use
- General-purpose (not domain-specific) ways of adding features
- Affinity Measures: capture structural information about graph ([Velingker et al., 2023])
  - Effective resistances (or commute time)
  - Hitting times
  - Resistive embeddings

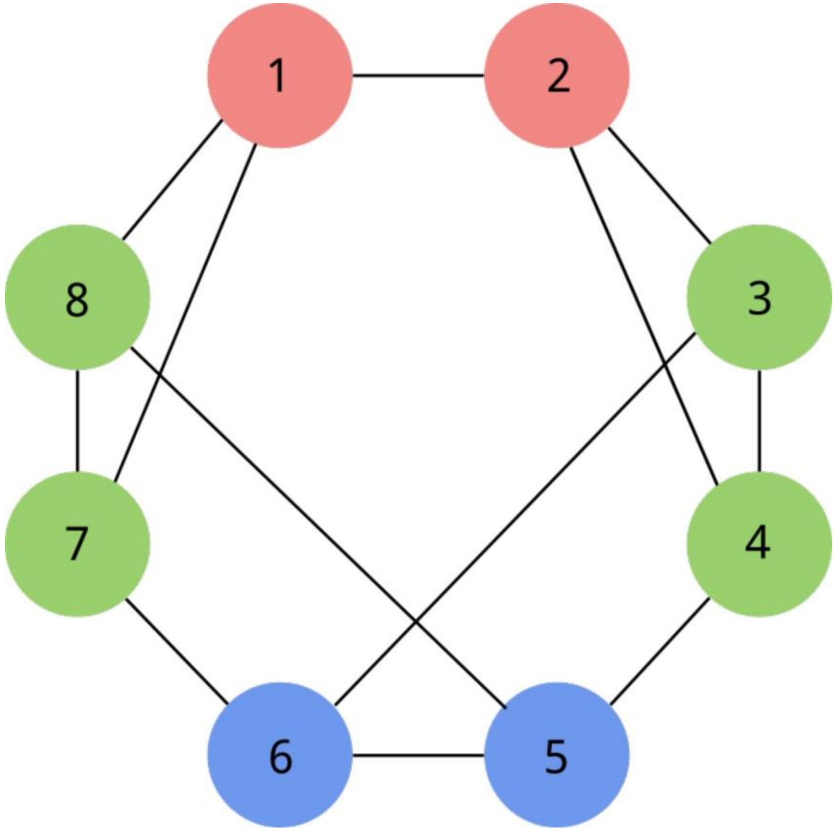
Effective Resistance

- ERs capture topological structure in graph
- Widely used in theoretical computer science
  - Graph sparsification
  - Linear system solvers
  - Graph clustering
- View graph as electrical circuit
  - Edges are resistors
  - Send current between two points and measure effective resistance



Effective Resistance: Going Beyond 1-WL

- ERs capture topological structure in graph
- Widely used in theoretical computer science
  - Graph sparsification
  - Linear system solvers
  - Graph clustering
- View graph as electrical circuit
  - Edges are resistors
  - Send current between two points and measure effective resistance



Resistive Embeddings

- ERs are scalar features along each edge
- Define richer *vector embeddings* that capture more structure
- **Resistive embedding** for each node satisfying:

$$\|\mathbf{r}_u - \mathbf{r}_v\|_2^2 = \mathsf{Res}(u, v)$$

- ***Efficient computation using dimensionality reduction techniques (JL Lemma)***

Incorportating Affinity Measures into GNNs

- Use affinity measures as edge features in aggregation step!
- ER, hitting time affinity measures are scalar features
- Use Resistive Embeddings as node features

$$\mathbf{m}_{u,v}^{(t+1)} = M_t(\mathbf{h}_u^{(t)}, \mathbf{h}_v^{(t)}, \mathbf{e}_{u,v})$$

$$\mathbf{a}_u^{(t+1)} = f_{\text{aggregate}}(\{\!\!\{\mathbf{m}_{u,v}^{(t+1)} \mid v \in \mathcal{N}(u)\}\!\!\})$$

$$\mathbf{h}_u^{(t+1)} = f_{\text{update}}(\mathbf{h}_u^{(t)}, \mathbf{a}_u^{(t+1)})$$

Large-Scale Molecular Graphs: PCQM4M-LSCv1

- PCQM4M-LSCv1 in KDD Cup 2021 Contest
- **Best published single model result** (validation MAE < 0.12)
- Outperforms without molecular geometric features or use of dense attention networks!!!

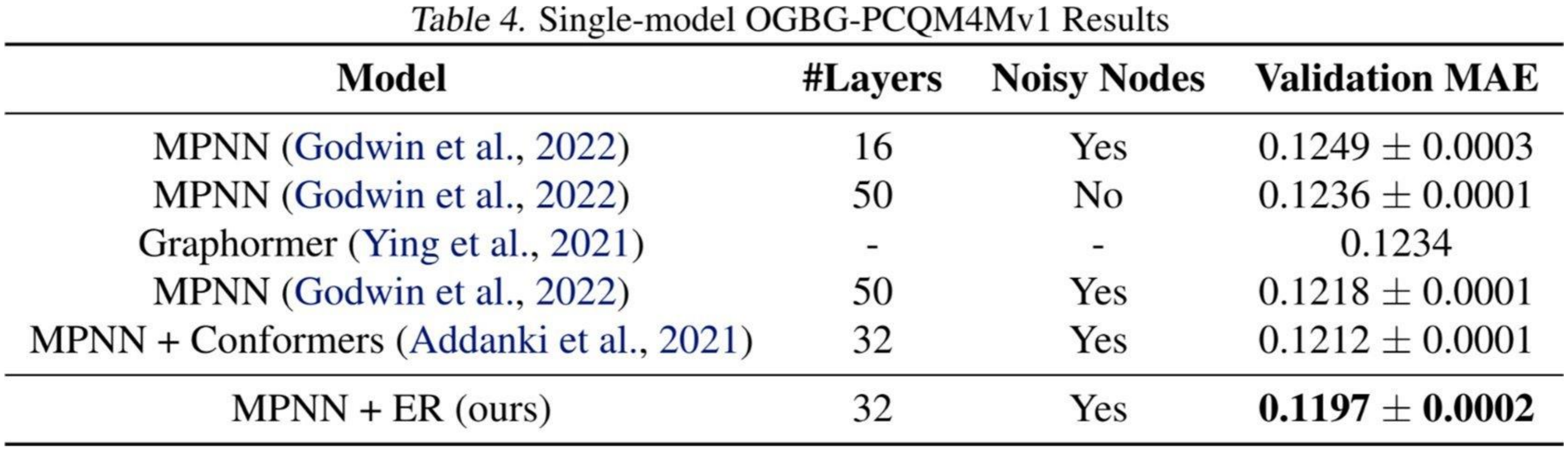


Table 4. Single-model OGBG-PCQM4Mv1 Results

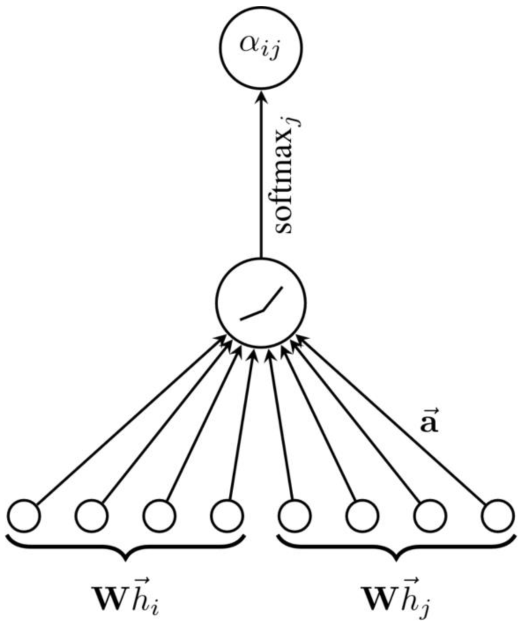
Model	#Layers	Noisy Nodes	Validation MAE
MPNN (Godwin et al., 2022)	16	Yes	0.1249 ± 0.0003
MPNN (Godwin et al., 2022)	50	No	0.1236 ± 0.0001
Graphormer (Ying et al., 2021)	-	-	0.1234
MPNN (Godwin et al., 2022)	50	Yes	0.1218 ± 0.0001
MPNN + Conformers (Addanki et al., 2021)	32	Yes	0.1212 ± 0.0001
MPNN + ER (ours)	32	Yes	**0.1197 ± 0.0002**

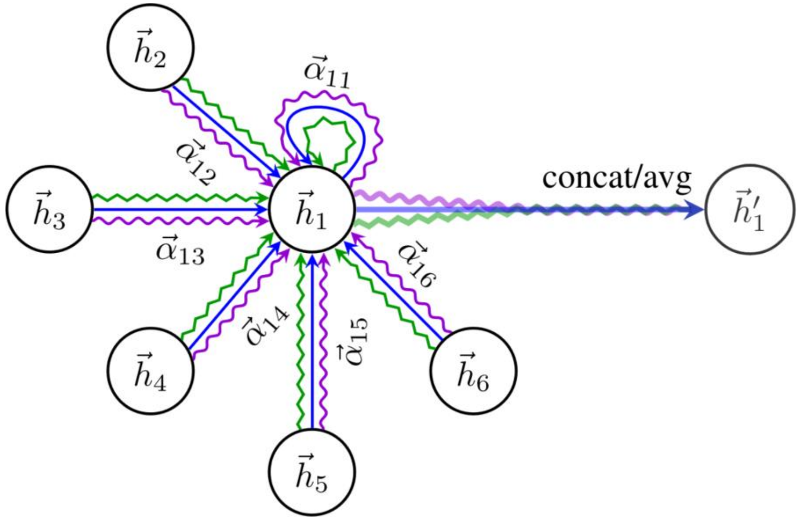
How to Enhance Expressivity?

- Standard GNNs limited by 1-WL graph isomorphism test
- Ways to improve GNN expressivity
  - Add features
  - **Modulate message-passing**
  - Modify underlying graph

Modulate the Message Passing

- Instead of adding features, modify the message passing mechanism itself
- Allow *anisotropic* aggregation of messages from neighbors
- We already saw one example: GAT

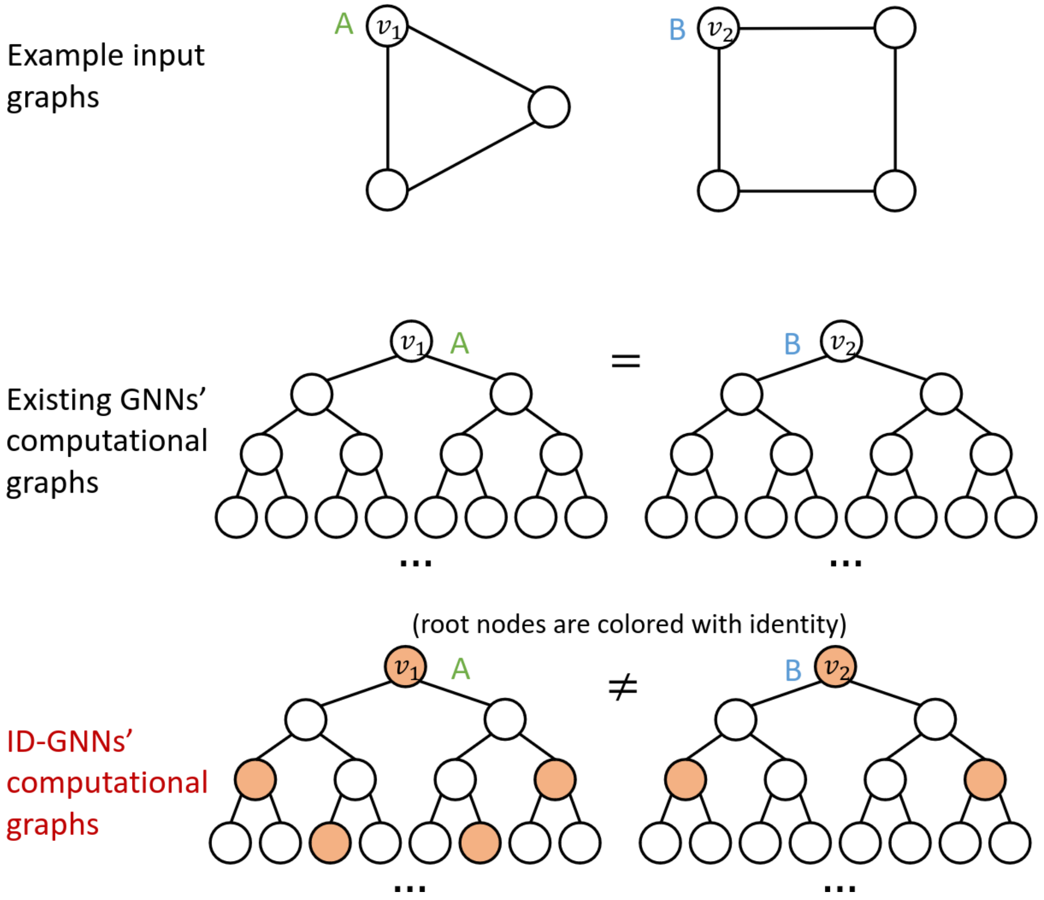




Identity-Aware Graph Networks

[You et al., 2021]

- Use heterogeneous message-passing to distinguish "root" node from other nodes
  - First compute ego network centered at a node of interest
  - Isolate instances of the center node in the computational graph
  - Apply message passing with different sets of parameters for center node vs. others
- Allows cycle detection



Example input graphs

Existing GNNs' computational graphs

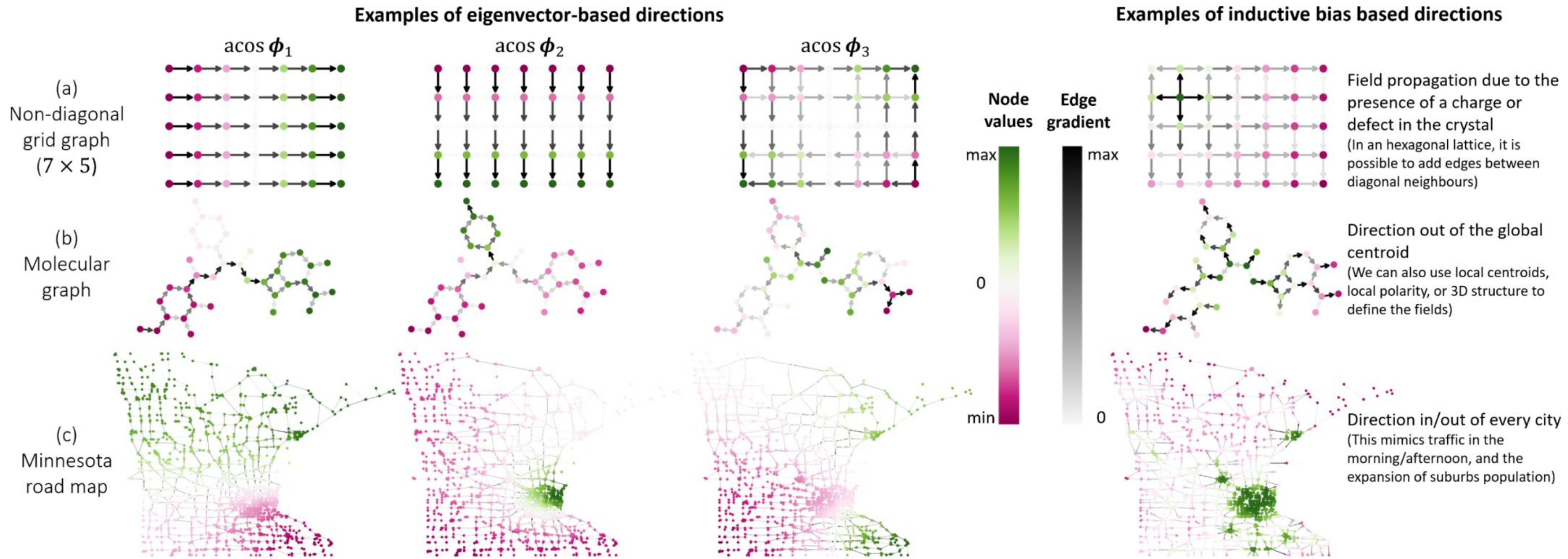
(root nodes are colored with identity)

ID-GNNs' computational graphs

Directional Graph Networks (DGN)

[Beaini et al., 2020]

- *Anisotropic* message passing using Laplacian flows



Examples of eigenvector-based directions
acos ϕ_1
acos ϕ_2
acos ϕ_3
Examples of inductive bias based directions
(a) Non-diagonal grid graph (7 × 5)
(b) Molecular graph
(c) Minnesota road map
Node values
Edge gradient
max
0
min
max
0

Field propagation due to the presence of a charge or defect in the crystal
(In an hexagonal lattice, it is possible to add edges between diagonal neighbours)

Direction out of the global centroid
(We can also use local centroids, local polarity, or 3D structure to define the fields)

Direction in/out of every city
(This mimics traffic in the morning/afternoon, and the expansion of suburbs population)

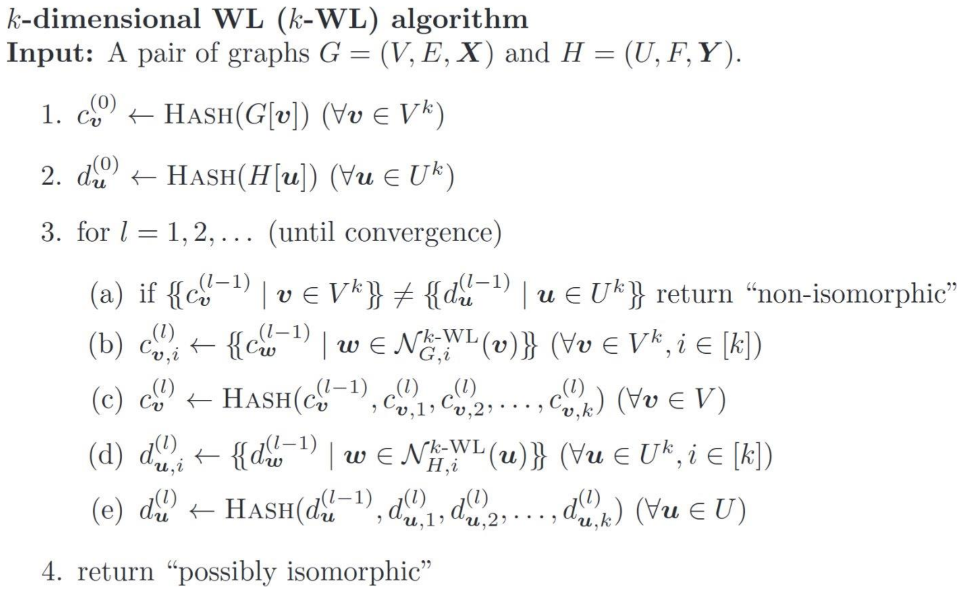
How to Enhance Expressivity?

- Standard GNNs limited by 1-WL graph isomorphism test
- Ways to improve GNN expressivity
  - Add features
  - Modulate message-passing
  - **Modify underlying graph**

Modify Underlying Graph

- Use a computation graph that is different from the input graph
  - Can add/remove edges or nodes to the input graph
  - Can be an altogether different graph
- Can be useful for datasets/tasks where the given input graph is noisy
- **Challenge**: Allow less restrictive computation while still maintaining the inductive bias of the input graph structure

Higher Order GNNs



k-dimensional WL (k-WL) algorithm
Input: A pair of graphs $G = (V, E, \boldsymbol{X})$ and $H = (U, F, \boldsymbol{Y})$.

```
1. $c_{\boldsymbol{v}}^{(0)} \leftarrow \textsc{Hash}(G[\boldsymbol{v}])$ $(\forall \boldsymbol{v} \in V^k)$
2. $d_{\boldsymbol{u}}^{(0)} \leftarrow \textsc{Hash}(H[\boldsymbol{u}])$ $(\forall \boldsymbol{u} \in U^k)$
3. for $l = 1, 2, \ldots$ (until convergence)
    (a) if $\{\!\{ c_{\boldsymbol{v}}^{(l-1)} \mid \boldsymbol{v} \in V^k \}\!\} \neq \{\!\{ d_{\boldsymbol{u}}^{(l-1)} \mid \boldsymbol{u} \in U^k \}\!\}$ return "non-isomorphic"
    (b) $c_{\boldsymbol{v},i}^{(l)} \leftarrow \{\!\{ c_{\boldsymbol{w}}^{(l-1)} \mid \boldsymbol{w} \in \mathcal{N}_{G,i}^{k\text{-WL}}(\boldsymbol{v}) \}\!\}$ $(\forall \boldsymbol{v} \in V^k, i \in [k])$
    (c) $c_{\boldsymbol{v}}^{(l)} \leftarrow \textsc{Hash}(c_{\boldsymbol{v}}^{(l-1)}, c_{\boldsymbol{v},1}^{(l)}, c_{\boldsymbol{v},2}^{(l)}, \ldots, c_{\boldsymbol{v},k}^{(l)})$ $(\forall \boldsymbol{v} \in V)$
    (d) $d_{\boldsymbol{u},i}^{(l)} \leftarrow \{\!\{ d_{\boldsymbol{w}}^{(l-1)} \mid \boldsymbol{w} \in \mathcal{N}_{H,i}^{k\text{-WL}}(\boldsymbol{u}) \}\!\}$ $(\forall \boldsymbol{u} \in U^k, i \in [k])$
    (e) $d_{\boldsymbol{u}}^{(l)} \leftarrow \textsc{Hash}(d_{\boldsymbol{u}}^{(l-1)}, d_{\boldsymbol{u},1}^{(l)}, d_{\boldsymbol{u},2}^{(l)}, \ldots, d_{\boldsymbol{u},k}^{(l)})$ $(\forall \boldsymbol{u} \in U)$
4. return "possibly isomorphic"
```

- Recall the WL test of order k, i.e., k-WL
- Hierarchy in expressivity/distinguishing power

1-WL = 2-WL < 3-WL < 4-WL < ...

- Build GNN architectures that mimic k-WL
- Hierarchy in expressivity/distinguishing power

Higher Order GNNs

[Morris et al., 2019]

- Input graph G = (V, E)
- $[V(G)]^k$ = set of k-element subsets of V
- Neighborhoods on $[V(G)]^k$:

$$\mathcal{N}(s) = \{t \in [V(G)]^k \mid |s \cap t| = k - 1\}$$

- *Local* neighborhoods:

$$\mathcal{N}_L(s) = \{t \in \mathcal{N}(s) \mid (v, w) \in E \text{ where } \{v\} = s \setminus t \text{ and } \{w\} = t \setminus s\}$$

Higher Order GNNs

[Morris et al., 2019]

- Input graph G = (V, E)
- $[V(G)]^k$ = set of k-element subsets of V

$$\mathcal{N}(s) = \{t \in [V(G)]^k \mid |s \cap t| = k - 1\}$$

$$\mathcal{N}_L(s) = \{t \in \mathcal{N}(s) \mid (v, w) \in E \text{ where } \{v\} = s \setminus t \text{ and } \{w\} = t \setminus s\}$$

- Aggregation and update rule:

$$\mathbf{h}^{(t+1)}(s) = \sigma\left(\mathbf{W}_1^{(t)} \cdot \mathbf{h}^{(t)}(s) + \sum_{u \in \mathcal{N}_L(s)} \mathbf{W}_2^{(t)} \cdot \mathbf{h}^{(t)}(u)\right)$$

High Order Hierarchy

- Variety of high-order WL variants
- Various new GNN architectures that are as expressive as k-WL ([Azizian and Lelarge, 2020], [Geerts, 2020], [Maron et al., 2019])
- Sparse variants (e.g., SpeqNets [Morris et al., 2022])
- High-order GNNs still suffer from a O(n^k) dependence – k > 3 impractical for larger graphs

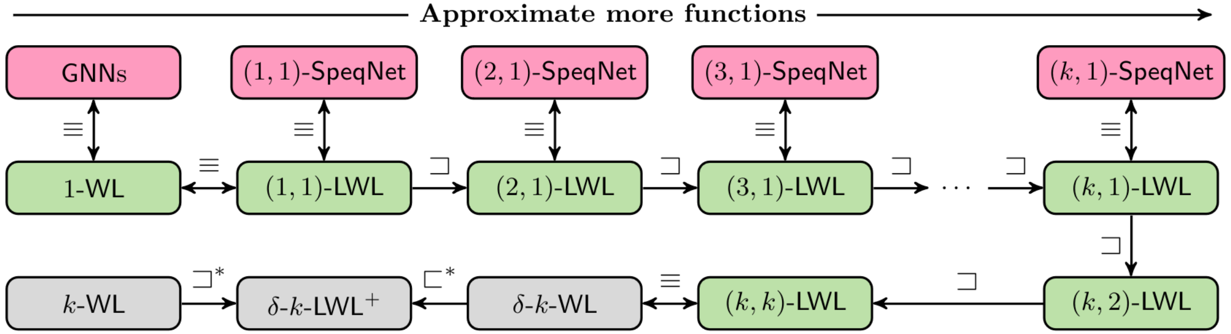
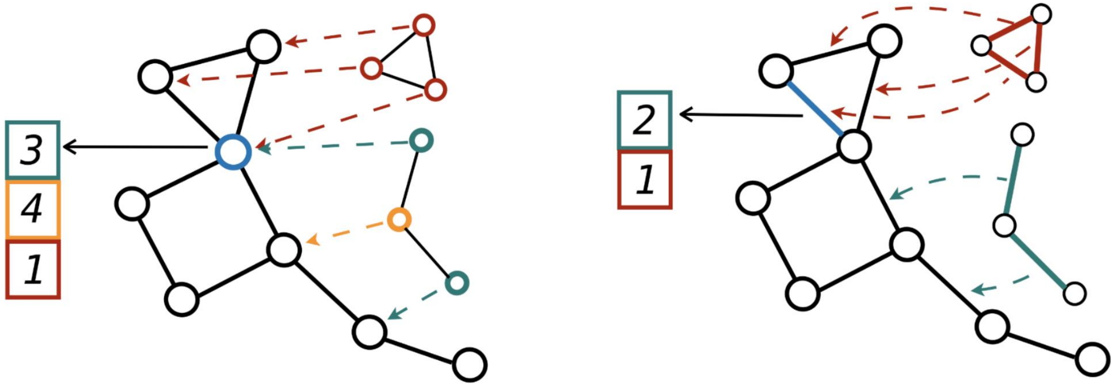


Figure 1: Overview of the power of the proposed algorithms and neural architectures. The green and red nodes represent algorithms proposed in the present work. Forward arrows point to more powerful algorithms or neural architectures. *—Proven in [Morris et al., 2020]. $A \sqsubset B$ ($A \equiv B$): algorithm A is strictly more powerful than (equally powerful as) B.

GENERALIZABILITY

Generalizability of GNNs

- Expressivity vs. generalizability
- More expressive networks lead to overfitting?
- In practice, no! Expressivity and generalizability often go hand in hand
- Subgraph-based enhancements: Graph Substructure Networks (GSN)



Graph Substructure Networks (GSN)

[Bouritsas, Frasca, Zafeiriou, Bronstein '22]

- Experimental results show improvements

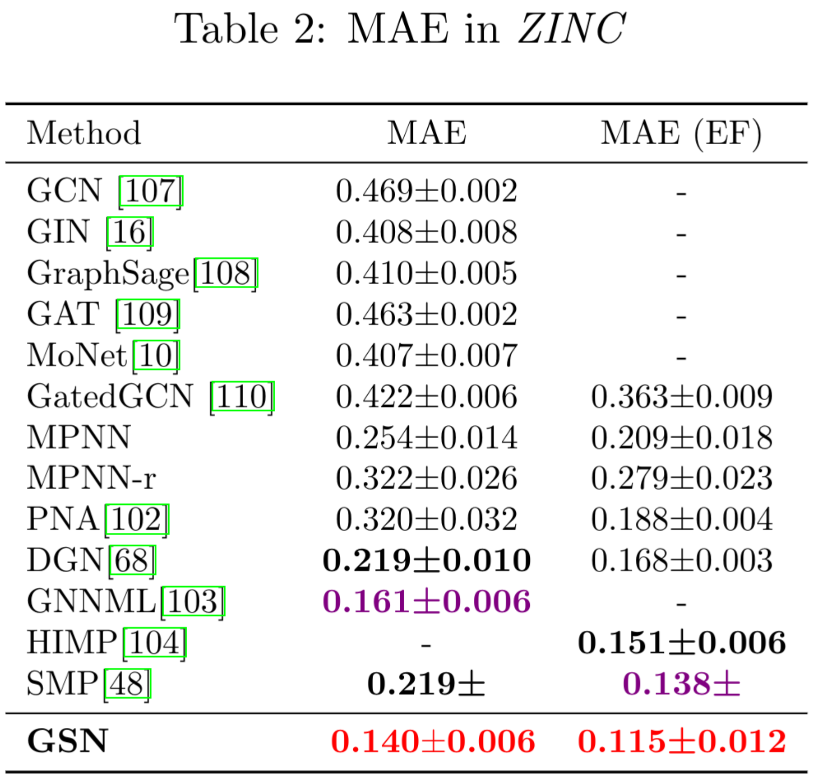


Table 2: MAE in *ZINC*

Method	MAE	MAE (EF)
GCN [107]	0.469±0.002	-
GIN [16]	0.408±0.008	-
GraphSage[108]	0.410±0.005	-
GAT [109]	0.463±0.002	-
MoNet[10]	0.407±0.007	-
GatedGCN [110]	0.422±0.006	0.363±0.009
MPNN	0.254±0.014	0.209±0.018
MPNN-r	0.322±0.026	0.279±0.023
PNA[102]	0.320±0.032	0.188±0.004
DGN[68]	**0.219±0.010**	0.168±0.003
GNNML[103]	**0.161±0.006**	-
HIMP[104]	-	**0.151±0.006**
SMP[48]	**0.219±**	**0.138±**
GSN	**0.140±0.006**	**0.115±0.012**

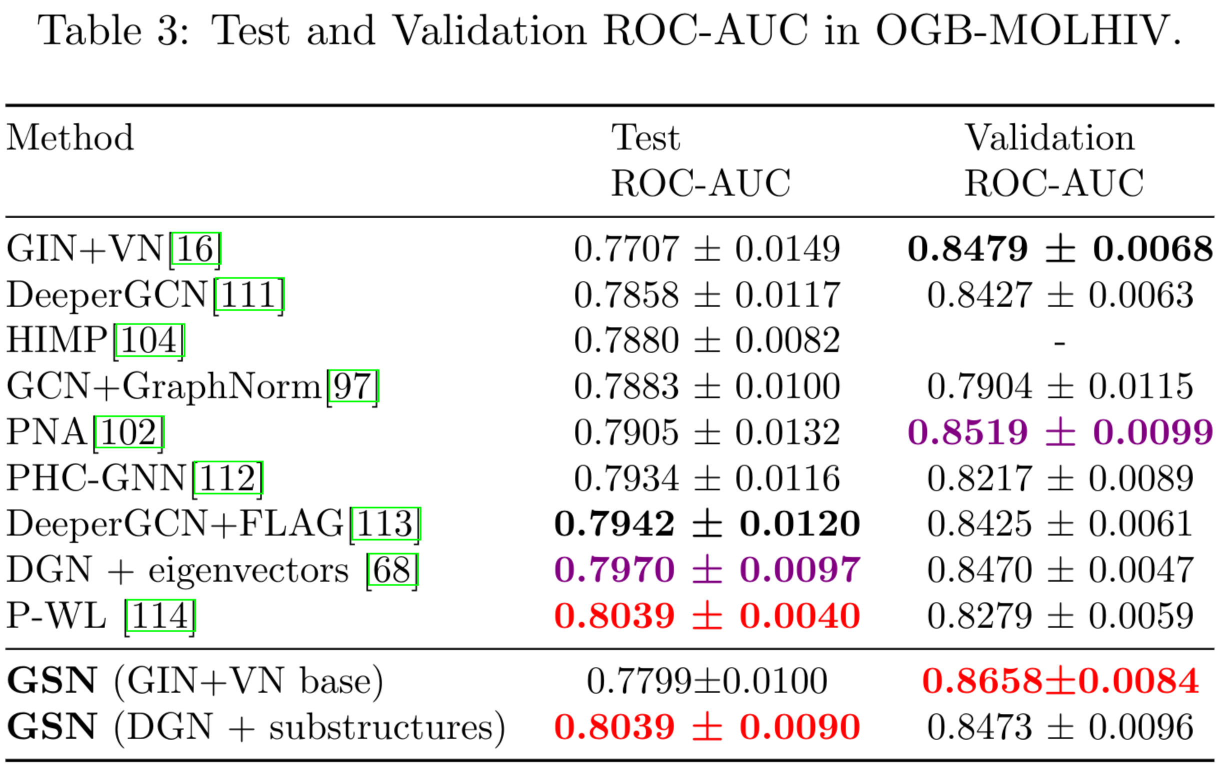
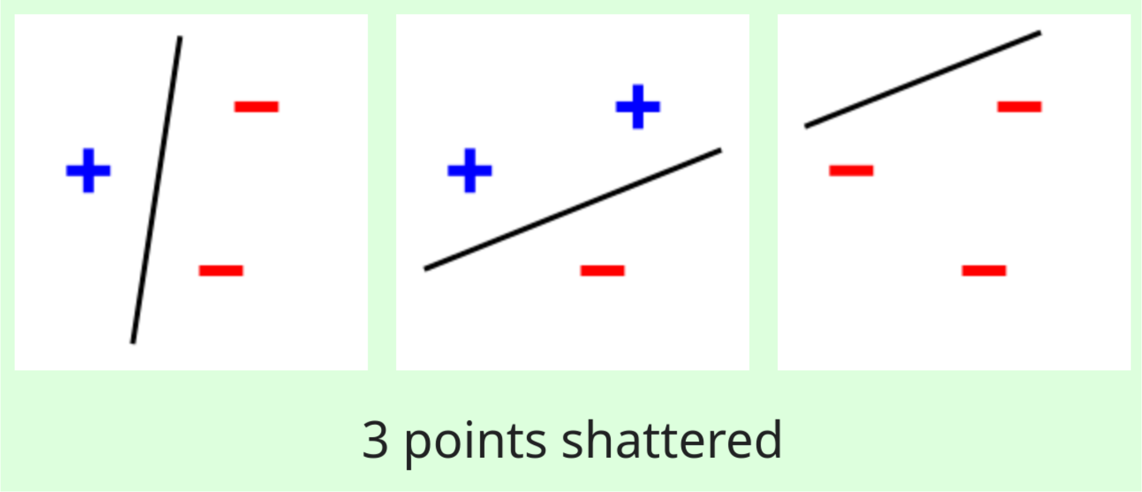


Table 3: Test and Validation ROC-AUC in OGB-MOLHIV.

Method	Test	Validation
	ROC-AUC	ROC-AUC
GIN+VN[16]	0.7707 ± 0.0149	**0.8479 ± 0.0068**
DeeperGCN[111]	0.7858 ± 0.0117	0.8427 ± 0.0063
HIMP[104]	0.7880 ± 0.0082	-
GCN+GraphNorm[97]	0.7883 ± 0.0100	0.7904 ± 0.0115
PNA[102]	0.7905 ± 0.0132	**0.8519 ± 0.0099**
PHC-GNN[112]	0.7934 ± 0.0116	0.8217 ± 0.0089
DeeperGCN+FLAG[113]	**0.7942 ± 0.0120**	0.8425 ± 0.0061
DGN + eigenvectors [68]	**0.7970 ± 0.0097**	0.8470 ± 0.0047
P-WL [114]	**0.8039 ± 0.0040**	0.8279 ± 0.0059
GSN (GIN+VN base)	0.7799±0.0100	**0.8658±0.0084**
GSN (DGN + substructures)	**0.8039 ± 0.0090**	0.8473 ± 0.0096

Vapnik-Chervonenkis (VC) Dimension

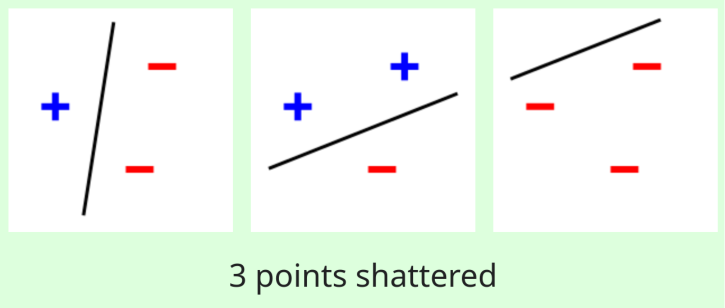
- Binary classification model f
- Model f (with params θ) shatters data points x1, x2, …, xn if for every assignment of labels, there exists θ for which f correctly classifies all xi
- VC dim = max number of points that are shattered by f



VC Dimension

- Statistical learning theory: VC dimension gives a bound on test error in terms of training error
- Hypothesis class *H* (output {-1, 1}), with *h* in H. VC dim = *d.* Training set size = m. Then, with probability ≥ 1 - δ, test error is not too big compared to training error:

$$E_{out}(h) \leq E_{in}(h) + \sqrt{\frac{8d \log(m/d) + 8 \log \frac{4}{\delta}}{m}}$$



WL Meets VC

[Morris, Geerts, Tönshoff, Grohe - ICML '23]

- Consider binary graph classification
- Class C of GNNs
- G_1, G_2, …, G_m are *shattered* by C if for any τ in $\{0, 1\}^m$, there exists a gnn in C such that:

$$\mathsf{gnn}(\mathbf{G}_i) = \begin{cases} \geq 2/3 & \text{if } \tau_i = 1, \text{ and} \\ \leq 1/3 & \text{if } \tau_i = 0. \end{cases}$$

WL Meets VC

[Morris, Geerts, Tönshoff, Grohe - ICML '23]

- VC dimension bounds for a variety of settings:

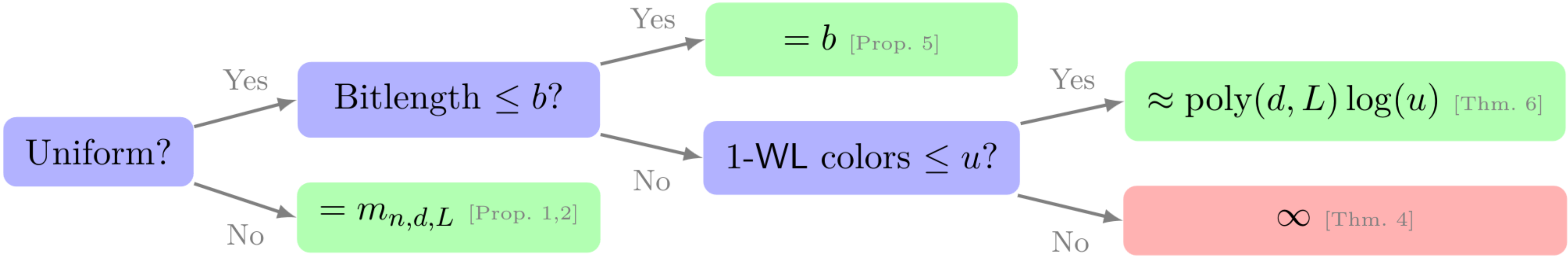


Figure 1: Overview of our results for bounded-width GNNs. Green and red boxes denote VC dimension bounds. Here, $m_{n,d,L}$ denotes the number of graphs of order at most n with boolean d-dimensional features distinguishable by 1-WL after L iterations.

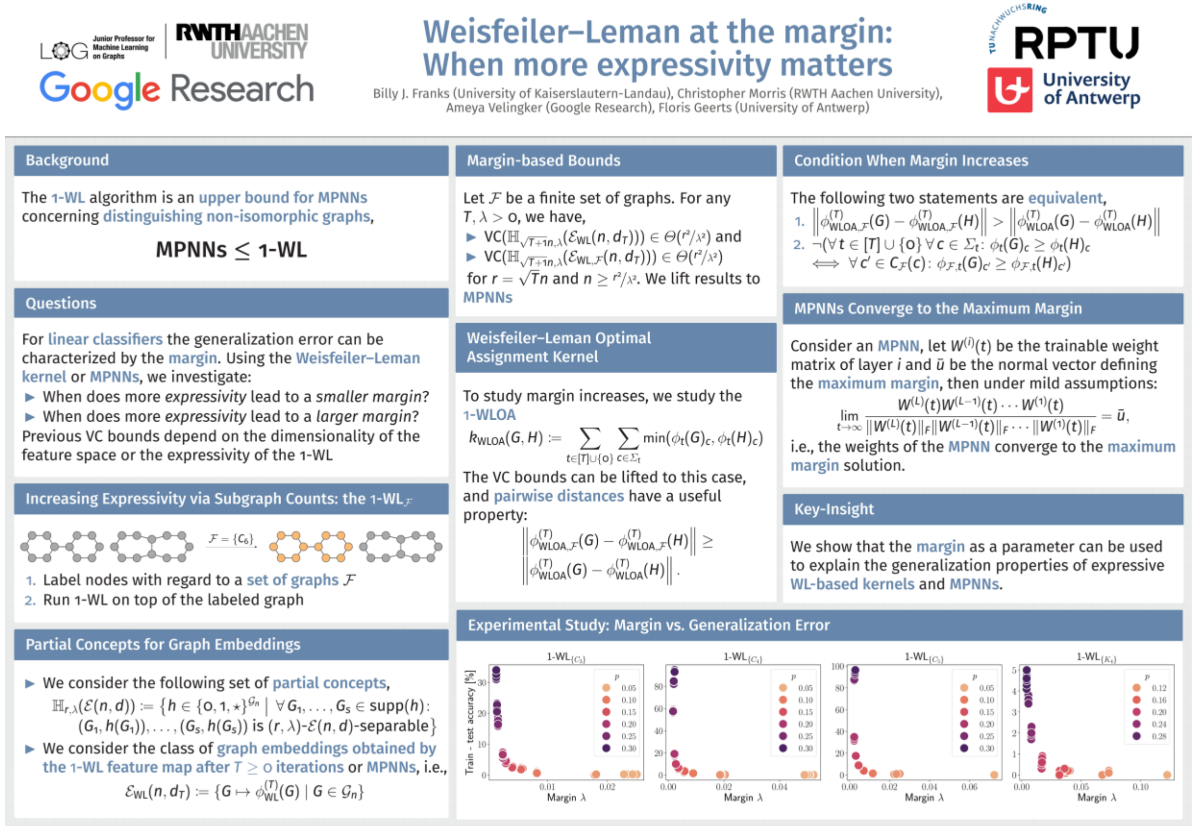
Expressivity vs. Generalizability

- Empirical results (e.g., Graph Substructure Networks (GSN)) show added expressivity results in improved predictive performance
- Upper and lower bounds on VC dimensions of message-passing GNNs ([Morris, Geerts, Tönshoff, Grohe - ICML '23])
- *Question*: Why does increased expressivity correspond to better generalization while keeping the training set equal?
  - [Morris et al. '23] demonstrated correlation between VC dimension and the number of non-isomorphic graphs that 1-WL can differentiate
  - Increased expressivity ==> higher VC dimension

Expressivity vs. Generalizability

[Franks, Morris, Velingker, Geerts – ICML '24]

- Initial work in ICLR 2024 in Vienna: *Bridging the Gap Between Practice and Theory in Deep Learning (BGPT)* workshop
- Poster on Thursday!



Weisfeiler–Leman at the margin: When more expressivity matters

Billy J. Franks (University of Kaiserslautern-Landau), Christopher Morris (RWTH Aachen University), Ameya Velingker (Google Research), Floris Geerts (University of Antwerp)

Background

The 1-WL algorithm is an upper bound for MPNNs concerning distinguishing non-isomorphic graphs,

MPNNs ≤ 1-WL

Questions

For linear classifiers the generalization error can be characterized by the margin. Using the Weisfeiler–Leman kernel or MPNNs, we investigate:

- When does more *expressivity* lead to a *smaller margin*?
- When does more *expressivity* lead to a *larger margin*?

Previous VC bounds depend on the dimensionality of the feature space or the expressivity of the 1-WL

Increasing Expressivity via Subgraph Counts: the 1-WL$_\mathcal{F}$

1. Label nodes with regard to a set of graphs $\mathcal{F}$
2. Run 1-WL on top of the labeled graph

Partial Concepts for Graph Embeddings

- We consider the following set of partial concepts,

$$\mathbb{H}_{r,\lambda}(\mathcal{E}(n,d)) := \{h \in \{0,1,\star\}^{\mathcal{G}_n} \mid \forall G_1,\ldots,G_s \in \text{supp}(h): (G_1,h(G_1)),\ldots,(G_s,h(G_s)) \text{ is } (r,\lambda)\text{-}\mathcal{E}(n,d)\text{-separable}\}$$

- We consider the class of graph embeddings obtained by the 1-WL feature map after $T \geq 0$ iterations or MPNNs, i.e.,

$$\mathcal{E}_{\text{WL}}(n,d_T) := \{G \mapsto \phi^{(T)}_{\text{WL}}(G) \mid G \in \mathcal{G}_n\}$$

Margin-based Bounds

Let $\mathcal{F}$ be a finite set of graphs. For any $T, \lambda > 0$, we have,

- $\text{VC}(\mathbb{H}_{\sqrt{T+1}n,\lambda}(\mathcal{E}_{\text{WL}}(n,d_T))) \in \Theta(r^2/\lambda^2)$ and
- $\text{VC}(\mathbb{H}_{\sqrt{T+1}n,\lambda}(\mathcal{E}_{\text{WL},\mathcal{F}}(n,d_T))) \in \Theta(r^2/\lambda^2)$

for $r = \sqrt{T}n$ and $n \geq r^2/\lambda^2$. We lift results to MPNNs

Weisfeiler–Leman Optimal Assignment Kernel

To study margin increases, we study the 1-WLOA

$$k_{\text{WLOA}}(G,H) := \sum_{t\in[T]\cup\{0\}} \sum_{c\in\Sigma_t} \min(\phi_t(G)_c, \phi_t(H)_c)$$

The VC bounds can be lifted to this case, and pairwise distances have a useful property:

$$\left\|\phi^{(T)}_{\text{WLOA},\mathcal{F}}(G) - \phi^{(T)}_{\text{WLOA},\mathcal{F}}(H)\right\| \geq \left\|\phi^{(T)}_{\text{WLOA}}(G) - \phi^{(T)}_{\text{WLOA}}(H)\right\|.$$

Condition When Margin Increases

The following two statements are equivalent,

1. $\left\|\phi^{(T)}_{\text{WLOA},\mathcal{F}}(G) - \phi^{(T)}_{\text{WLOA},\mathcal{F}}(H)\right\| > \left\|\phi^{(T)}_{\text{WLOA}}(G) - \phi^{(T)}_{\text{WLOA}}(H)\right\|$
2. $\neg(\forall\, t \in [T]\cup\{0\}\ \forall\, c \in \Sigma_t\colon \phi_t(G)_c \geq \phi_t(H)_c$
   $\iff \forall\, c' \in C_\mathcal{F}(c)\colon \phi_{\mathcal{F},t}(G)_{c'} \geq \phi_{\mathcal{F},t}(H)_{c'})$

MPNNs Converge to the Maximum Margin

Consider an MPNN, let $W^{(i)}(t)$ be the trainable weight matrix of layer i and $\bar{u}$ be the normal vector defining the maximum margin, then under mild assumptions:

$$\lim_{t\to\infty} \frac{W^{(L)}(t)W^{(L-1)}(t)\cdots W^{(1)}(t)}{\|W^{(L)}(t)\|_F\|W^{(L-1)}(t)\|_F\cdots\|W^{(1)}(t)\|_F} = \bar{u},$$

i.e., the weights of the MPNN converge to the maximum margin solution.

Key-Insight

We show that the margin as a parameter can be used to explain the generalization properties of expressive WL-based kernels and MPNNs.

Experimental Study: Margin vs. Generalization Error

PREVIEW: Expressivity vs. Generalizability

[Franks, Morris, Velingker, Geerts – ICML '24]

- Consider linear classifiers
- Generalization error is characterized by the margin
- Address the question: When does expressivity lead to a larger vs smaller margin?
- Extend theory of partial concepts ([Alon et al., FOCS’21]) to MPNNs to get margin-based VC bounds

PREVIEW: Gradient Flow Convergence to Max Margin

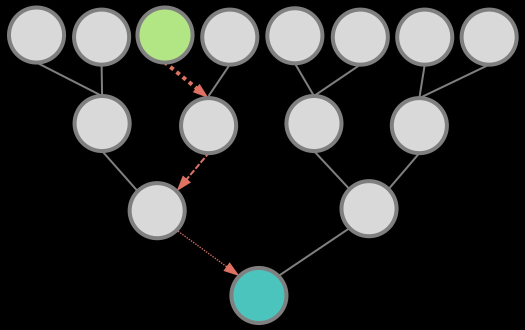
[Franks, Morris, Velingker, Geerts – ICML '24]

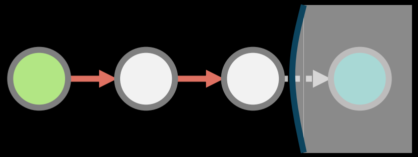
- MPNNs exhibit an "alignment" property
- Gradient flow pushes network weights toward the maximum margin solution
- Builds on results of [Ji and Telgarsky '19]

$$\mathbf{X}^{(i+1)} = \mathbf{W}^{(i+1)}\mathbf{X}^{(i)}\mathbf{A}'(G),$$

$$\hat{y} = \mathsf{READOUT}(\mathbf{X}^{(L)}) = \mathbf{X}^{(L)} \cdot \mathbf{1}_n,$$

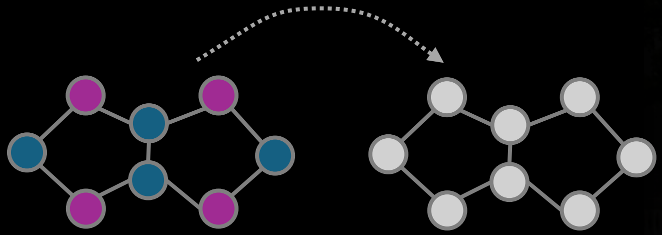
$$\lim_{t\to\infty} \frac{\mathbf{W}^{(L)}(t)\mathbf{W}^{(L-1)}(t)\cdots\mathbf{W}^{(1)}(t)}{\|\mathbf{W}^{(L)}(t)\|_F\|\mathbf{W}^{(L-1)}(t)\|_F\cdots\|\mathbf{W}^{(1)}(t)\|_F} = \bar{\mathbf{u}}.$$





Challenges for GNNs

Under-reaching, Over-smoothing and Over-squashing



Common origin of the problems

GNNs arise to leverage **information on the graph topology** to improve inference

HOW?

- Diffusion of information over the structure ***A***
  - → **Locality** nature or **Smoothness principle**
- Repeated computation over ***X*** to reach information over the **k-hop** neighborhood

Source: D. Zelle et al. GNNs in TensorFlow. Google Research Blog. 2024

$$\mathbf{h}_u^{(l+1)} = \phi\left(\mathbf{h}_u^l, \bigoplus_{v\in\mathcal{N}_u} \psi\left(\mathbf{h}_u^l, \mathbf{h}_v^l\right)\right)$$

Random Walks	Graph Diffusion
$\boldsymbol{P} = \boldsymbol{D}\boldsymbol{A}$	$\boldsymbol{S} = \sum_{k=0}^{\infty} \theta_k \boldsymbol{T}^k$

Lovasz 1993; Chung 1997; Kondor 2002

Common origin of the problems

GNNs arise to leverage **information on the graph topology** to improve inference

HOW?

- Diffusion of information over the structure ***A***
  - → **Locality** nature or **smoothness principle**
- Repeated computation over ***X*** to reach information over the **k-hop** neighborhood

Source: D. Zelle et al. GNNs in TensorFlow. Google Research Blog. 2024

$$\mathbf{h}_u^{(l+1)} = \phi\left(\mathbf{h}_u^l, \bigoplus_{v\in\mathcal{N}_u} \psi\left(\mathbf{h}_u^l, \mathbf{h}_v^l\right)\right)$$

Difficulties

☹ Long range dependencies

☹ Heterophily

☹ Uneven location distribution of labeled nodes

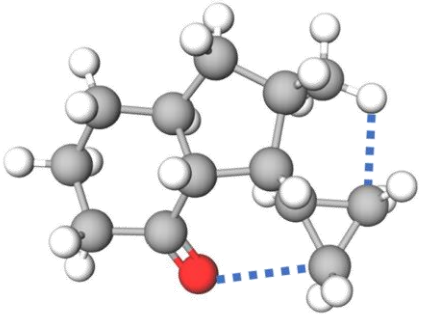
Random Walks	Graph Diffusion
$\boldsymbol{P} = \boldsymbol{D}\boldsymbol{A}$	$\boldsymbol{S} = \sum_{k=0}^{\infty} \theta_k \boldsymbol{T}^k$

Lovasz 1993; Chung 1997; Kondor 2002

Long-Range and Heterophily

Long Range tasks depend on interactions between distant nodes

[Alon. et al., 2020]



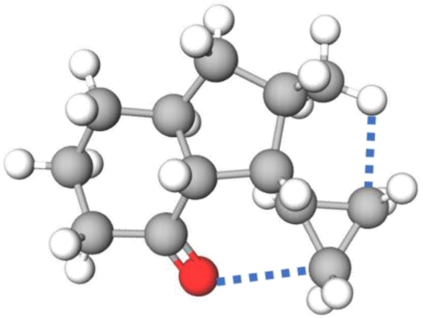
Long-range 3D atomic contact not captured by the structure

[Dwivedi et al., 2022]

Long-Range and Heterophily

Long Range tasks depend on interactions between distant nodes

[Alon. et al., 2020]



Long-range 3D atomic contact not captured by the structure

[Dwivedi et al., 2022]

Homophily metrics measure how the **graph structure aligns with the nodes' signals**

Most widely used in the literature: based on *1-hop neighbors*.

[Zhu, J., et al., 2020]

$$h_{\boldsymbol{edges}} = \frac{|\{(u,v) \in E: y_u = y_v\}|}{|E|}$$

[Pei, H. et al., 2019]

$$h_{\boldsymbol{nodes}} = \frac{1}{|V|} \sum_{v \in V} \frac{|\{u \in N(v): y_u = y_v\}|}{|N(v)|}$$

$$\mathrm{H}_{ij}(E) = \frac{|\{(u,v) \in E: y_u i \wedge y_v = j\}|}{|\{(u,v) \in E: y_u = i\}|}$$

$$h_{\boldsymbol{class}} = \frac{1}{|C| - 1} \sum_{c \in \mathrm{C}} \left[h_c - \frac{|C_c|}{n} \right]_+ , h_c = \frac{\sum_{v \in c} |\{u \in N(v): y_u = y_v\}|}{\sum_{v \in c} |N(v)|}$$

[Lim, D. et al., 2021]

Problems briefly

r = problem radius
k = n layers
d = graph diameter

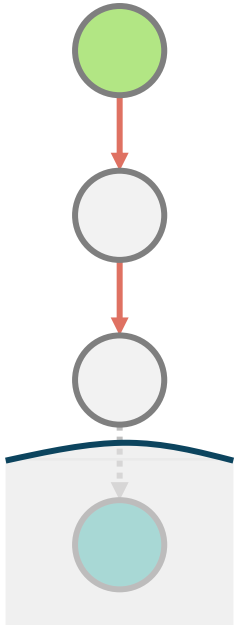
Under-reaching $\uparrow k$ **Over-smoothing**

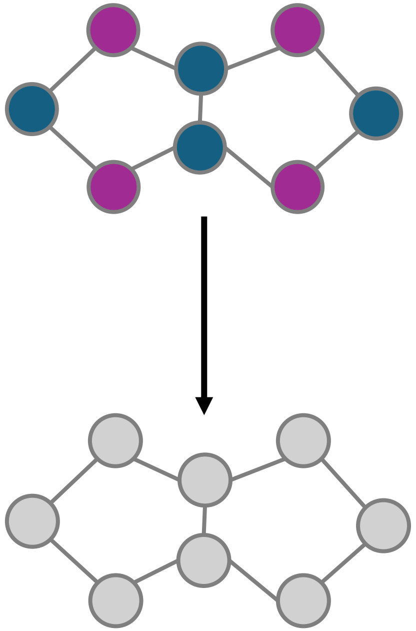
Over-squashing

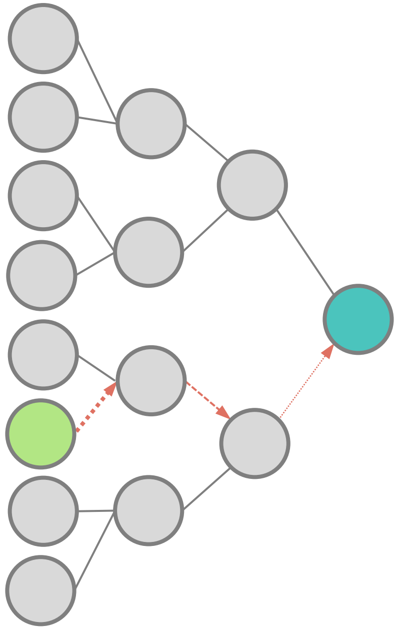
$k < r$

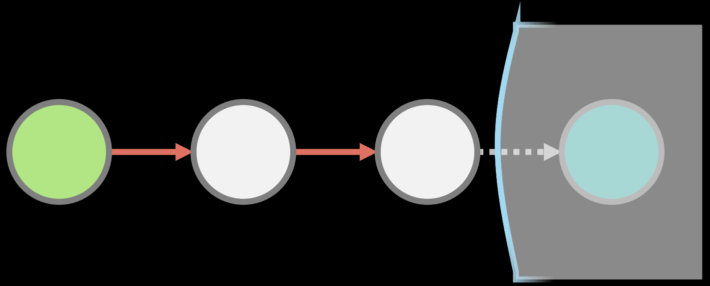
$k > r \approx d$

Node's receptive field increases exponentially with k



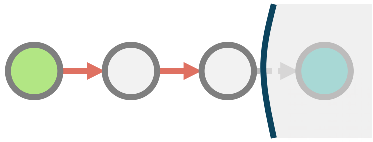






Under-reaching

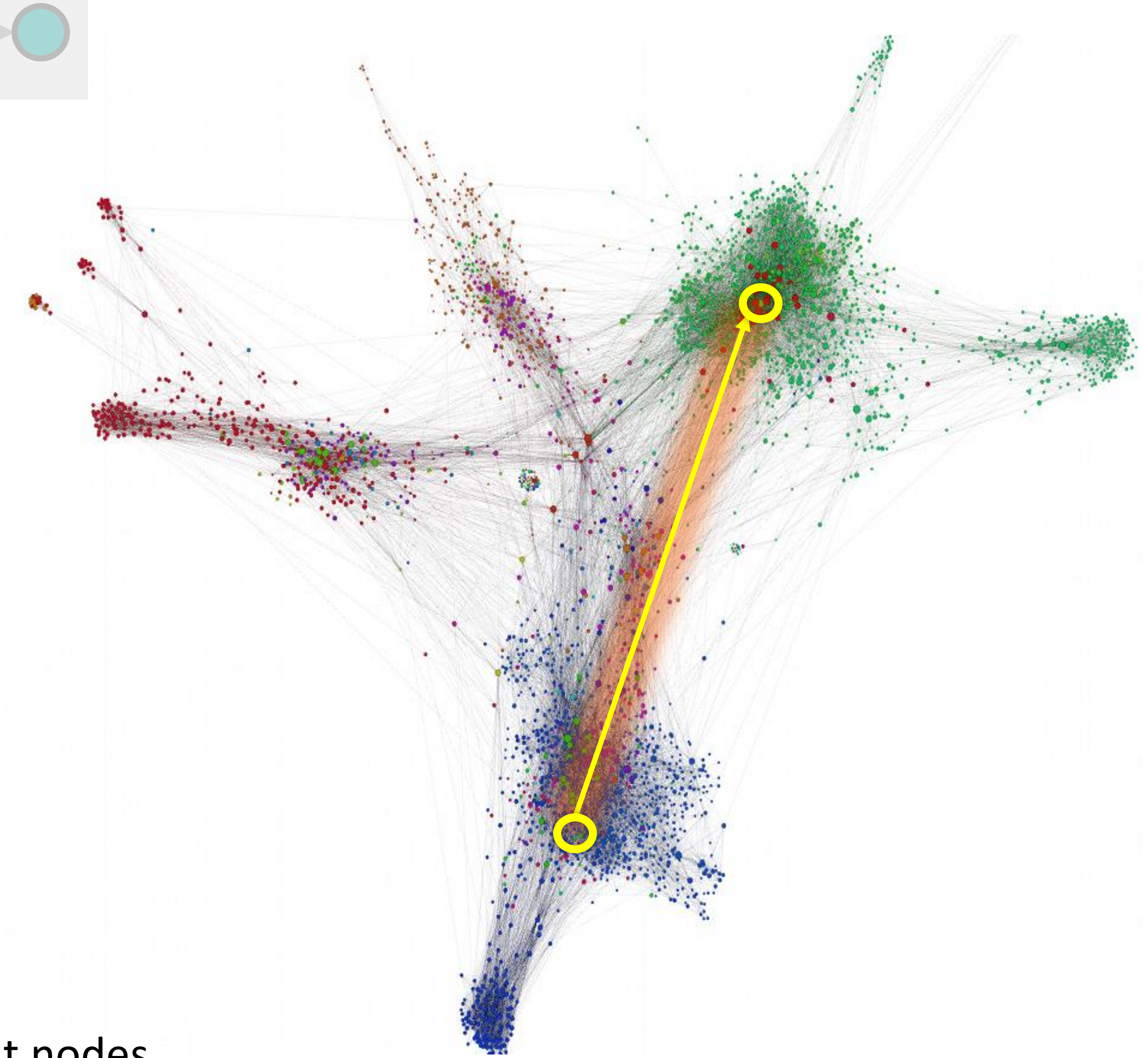
Under-reaching



- Inability of nodes to be aware of nodes that are farther away than the number of layers k [Barceló 2022]

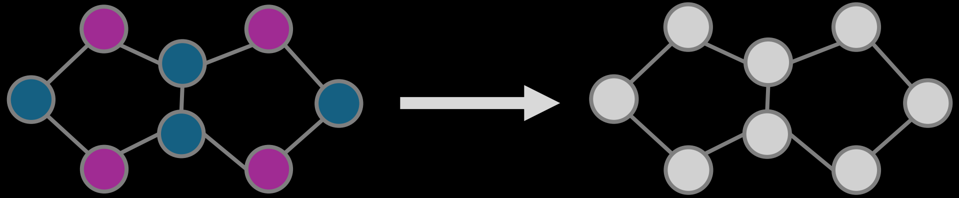
- **Inability of information to propagate further than $\boldsymbol{k}$ layers of the GNN** [Alon 2022]

- **Number of layers smaller than problem radius**
  - $\boldsymbol{k < r}$
- r typically grows with n → k dependent on the graph size



Solution

Stack $k > r$ layers so information is exchanged among distant nodes



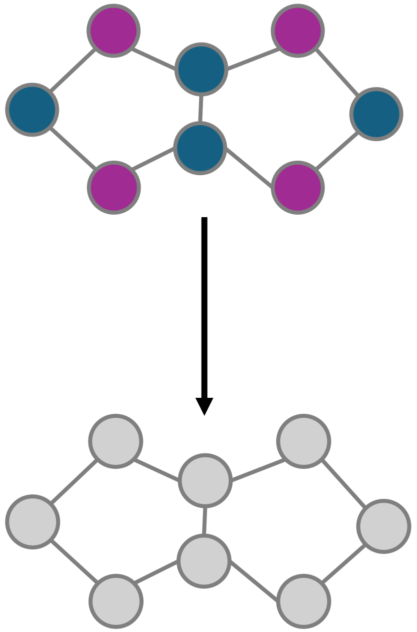
Over-smoothing

OSM

Over-smoothing

- When **stacking many layers** in a GNN, node representations can become **indistinguishable** [Li et al 2018; Oono and Suzuki 2020; Cai et al 2020; Chen et al 2020; Zhou et al 2020; Zhou et al 2020 ; Rusch et al 2023]

$$\sum_{(u,v)\in E} \left| h_u^k - h_n^k \right| \rightarrow 0 \text{ as } k \rightarrow \infty$$

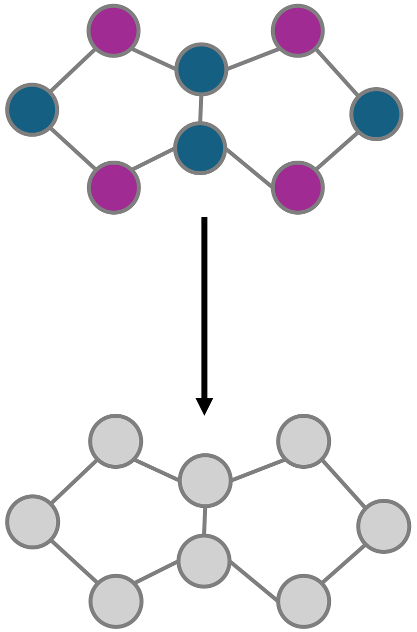


Convergence of the node embeddings as the number (k) of message passing layers increases

Over-smoothing

- When **stacking many layers** in a GNN, node representations can become **indistinguishable** [Li et al 2018; Oono and Suzuki 2020; Cai et al 2020; Chen et al 2020; Zhou et al 2020; Zhou et al 2020 ; Rusch et al 2023]

$$\sum_{(u,v)\in E} \left|h_u^k - h_n^k\right| \rightarrow 0 \text{ as } k \rightarrow \infty$$



- Conceptual origin of the problems: too much mixing

How many times do we mix
(*k layers*)

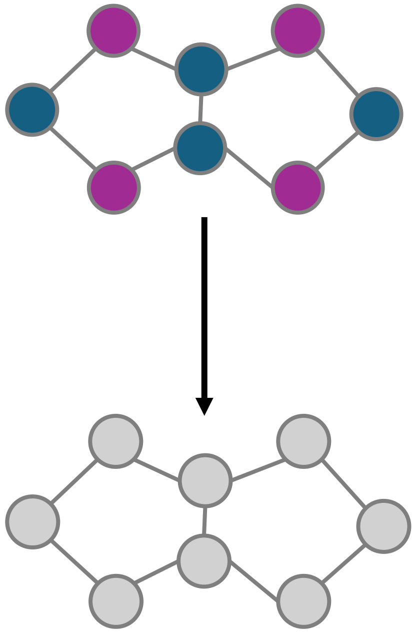
How information is mixed
A) **Connectivity of *G***
B) **GNN architecture**

Convergence of the node embeddings as the number (k) of message passing layers increases

Over-smoothing

- When **stacking many layers** in a GNN, node representations can become **indistinguishable** [Li et al 2018; Oono and Suzuki 2020; Cai et al 2020; Chen et al 2020; Zhou et al 2020; Zhou et al 2020 ; Rusch et al 2023]

$$\sum_{(u,v)\in E} \left|h_u^k - h_n^k\right| \to 0 \text{ as } k \to \infty$$



- Conceptual origin of the problems: too much mixing

How many times do we mix
(*k layers*)

How information is mixed
A) **Connectivity of *G***
B) **GNN architecture**

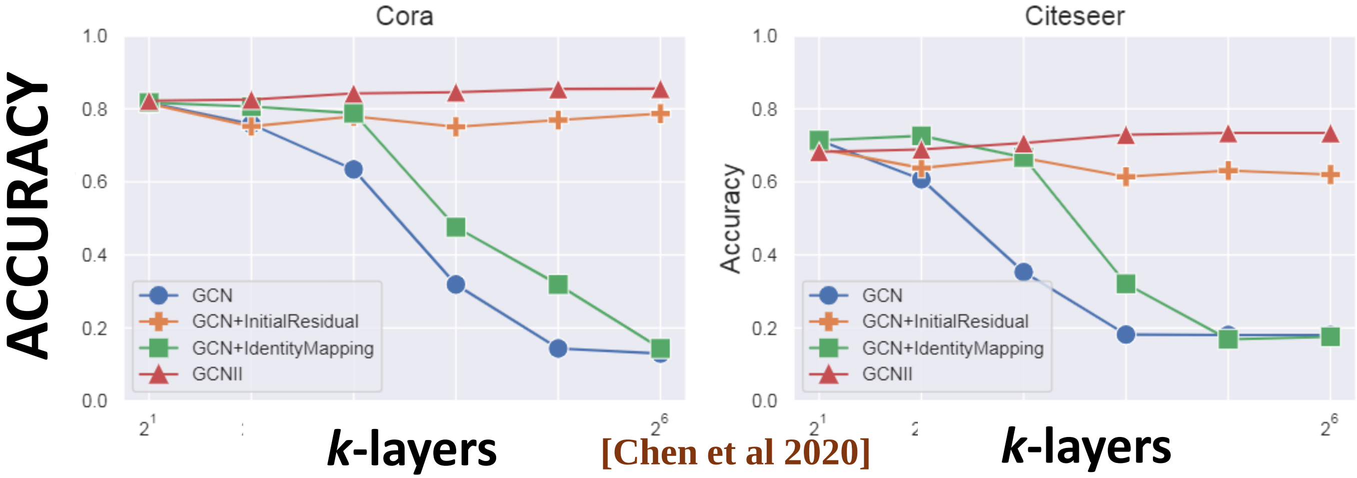
- “Independent” of the problem radius

Convergence of the node embeddings as the number (k) of message passing layers increases

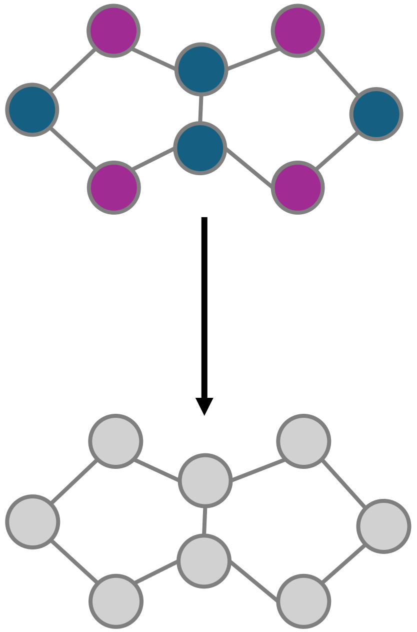
Over-smoothing

- When **stacking many layers** in a GNN, node representations can become **indistinguishable** [Li et al 2018; Oono and Suzuki 2020; Cai et al 2020; Chen et al 2020; Zhou et al 2020; Zhou et al 2020 ; Rusch et al 2023]

$$\sum_{(u,v)\in E} \left|h_u^k - h_n^k\right| \to 0 \text{ as } k \to \infty$$



[Chen et al 2020]



Convergence of the node embeddings as the number (k) of message passing layers increases

How do we measure Over-smoothing?

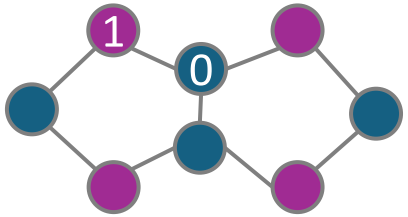
- Other metrics such as $\mathrm{MAD}_G(H^l) = \frac{1}{n} \sum_{v \in \mathcal{V}} \sum_{u \in \mathcal{N}_v} 1 - \frac{{h_v^l}^T h_u^l}{\|h_v^l\| \, \|h_u^l\|}$

- Consensus in using **Dirichlet Energy** of a signal on the Graph [Chung, 1997; Cai et al 2020; Rusch et al 2023]

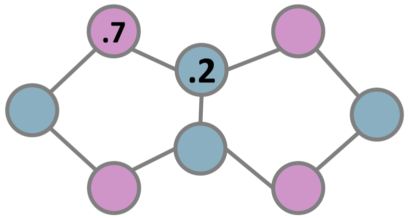
Principled

$$\mathcal{E}_G(H^l) = \mathrm{Tr}({H^l}^T L H^l) = \frac{1}{2} \sum_{u,v \in E} \left\| \frac{h_u}{\sqrt{d_u}} - \frac{h_v}{\sqrt{d_v}} \right\|_2^2$$

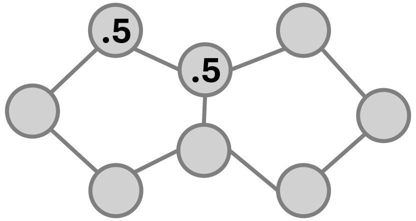
Constrained variability of signal *H* wrt *G*



$\mathcal{E}_G(H^l) = 1.66$



$\mathcal{E}_G(H^l) = 0.38$



$\mathcal{E}_G(H^l) = 0$

$$\Delta \mathcal{L}_i = f_i - \sum_{u \sim v} \frac{f_v}{\sqrt{d_u d_v}}$$

How do we measure Over-smoothing?

Recap on Eigenvectors [Chung, 1997]

Eigenvectors are a set orthonormal functions that minimize the Rayliegh Quotient (normalized DE) on *G*

$$R_G(f) = \frac{\mathrm{Tr}(f^T L f)}{\|f\|_2^2}$$

$$\phi_n = \mathrm{argmin}_{f \perp (f_1, \ldots, f_{n-1})} R_G(f)$$

$$\lambda_n = R_G(\phi_n) = \frac{\mathcal{E}(\phi_n)}{\phi_n^T \phi_n}$$

Eigenvalues are the Rayliegh Quotient (**normalized DE**)
of the **eigenvectors** of the graph (**orthonormal functions that minimizes DE**)

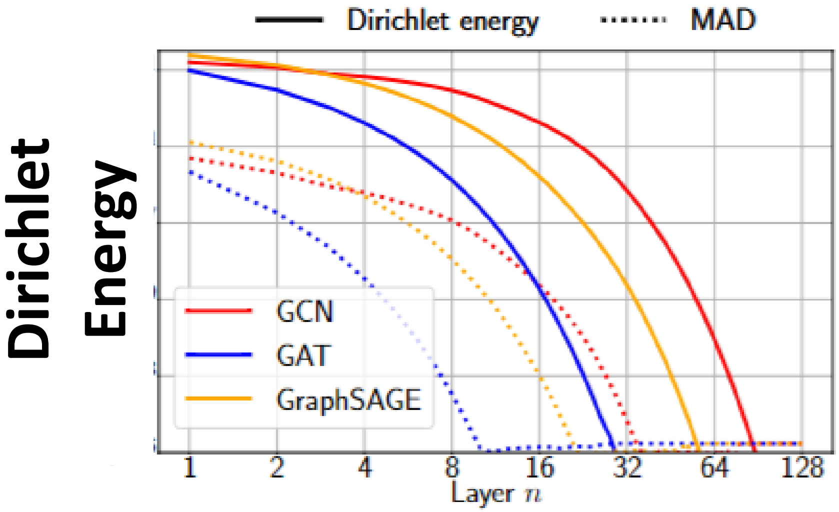
$$\mathcal{E}_G(H^l) = \mathrm{Tr}({H^l}^T L H^l)$$

How do we measure Over-smoothing?

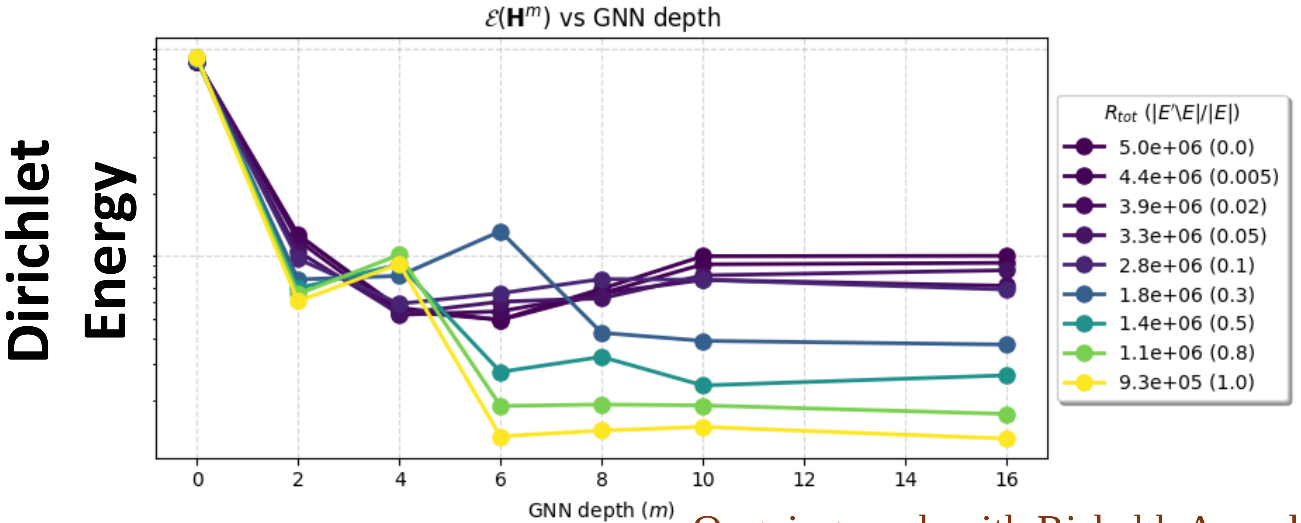
- **Dirichlet Energy** of a signal on the Graph
  [Chung, 1997; Cai et al 2020; Rusch et al 2023]

$$\mathcal{E}_G(H^l) = \mathrm{Tr}({H^l}^T L H^l) = \frac{1}{2}\sum_{u,v\in E}\left\|\frac{h_u}{\sqrt{d_u}} - \frac{h_v}{\sqrt{d_v}}\right\|_2^2$$

Constrained variability of signal *H* wrt *G*



[Rusch et al 2023]



Ongoing work with Rishabh Anand

Reasons for Over-smoothing

Random Walk perspective: stationary (stable) distribution

Connection with λ_2

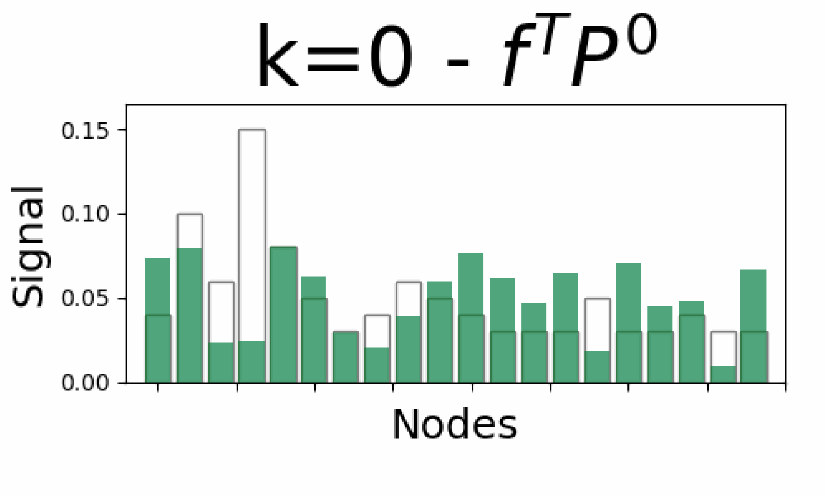
- **Over-smoothing as the stationary distribution π of a random walk in a Graph** [Chung, 1997; Spielman; 2018; Giraldo et al 2023]

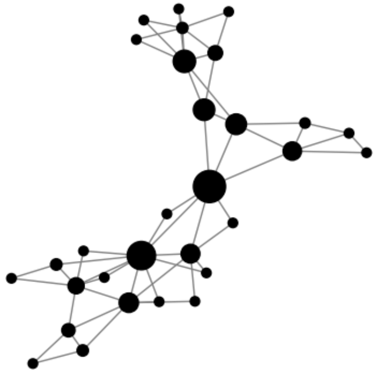
$$P = D^{-1}A, \quad f : \mathcal{V} \to \mathbb{R} \text{ with } \sum_v f(v) = 1$$ Distribution over nodes in *G*

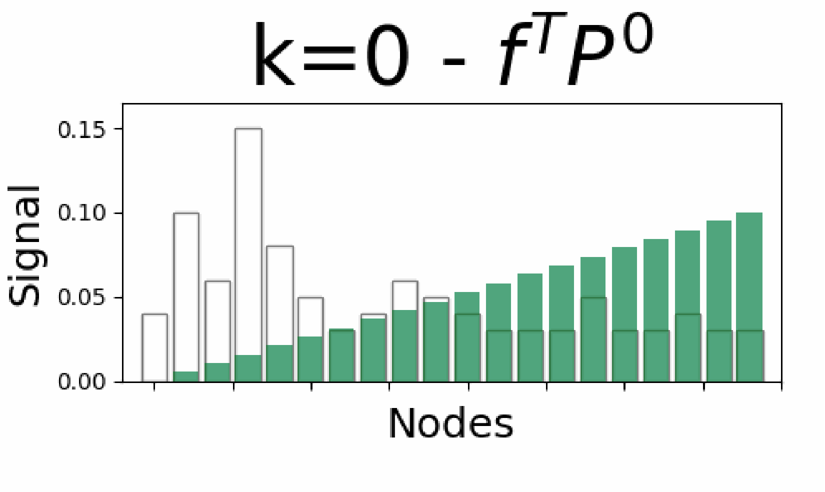
$$f^T P^k$$ Distribution over nodes in *G* after *k* steps

$$\lim_{k\to\infty} f^T P^k = \pi \longrightarrow \pi_u = \frac{d_u}{\sum_{v\in\mathcal{V}} d_v}$$

Converges to stationary distribution (no feature information)







Reasons for Over-smoothing

Random Walk perspective: stationary (stable) distribution

Connection with λ_2

- **Over-smoothing as the stationary distribution π of a random walk in a Graph**
  [Chung, 1997; Spielman; 2018; Giraldo et al 2023]

$$P = D^{-1}A, \qquad f : \mathcal{V} \to \mathbb{R} \text{ with } \sum_v f(v) = 1$$ Distribution over nodes in *G*

$f^T P^k$ Distribution over nodes in *G* after *k* steps

$$\lim_{k\to\infty} f^T P^k = \pi \longrightarrow \pi_u = \frac{d_u}{\sum_{v\in\mathcal{V}} d_v}$$

Converges to stationary distribution
(no feature information)

- $\boldsymbol{\lambda}_2$ denotes the **rate of convergence** → The higher the *spectral gap*, the faster the convergence to $\boldsymbol{\pi}$

$$\left\|f^T P^k - \pi\right\|_2 \leq e^{-k\boldsymbol{\lambda}'} \log\left(\frac{\max_v \sqrt{d_x}}{\min_u \sqrt{d_u}}\right)$$

Apply RW smoothing too many times → stationary point

The more connectivity, the higher the rate of convergence

Reasons for Over-smoothing

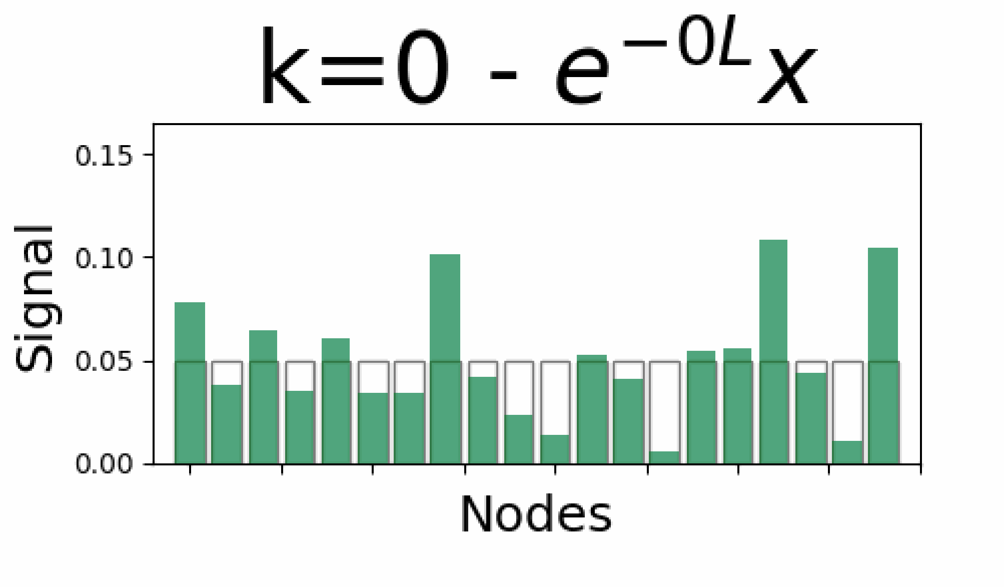
Random Walk perspective: averaging network

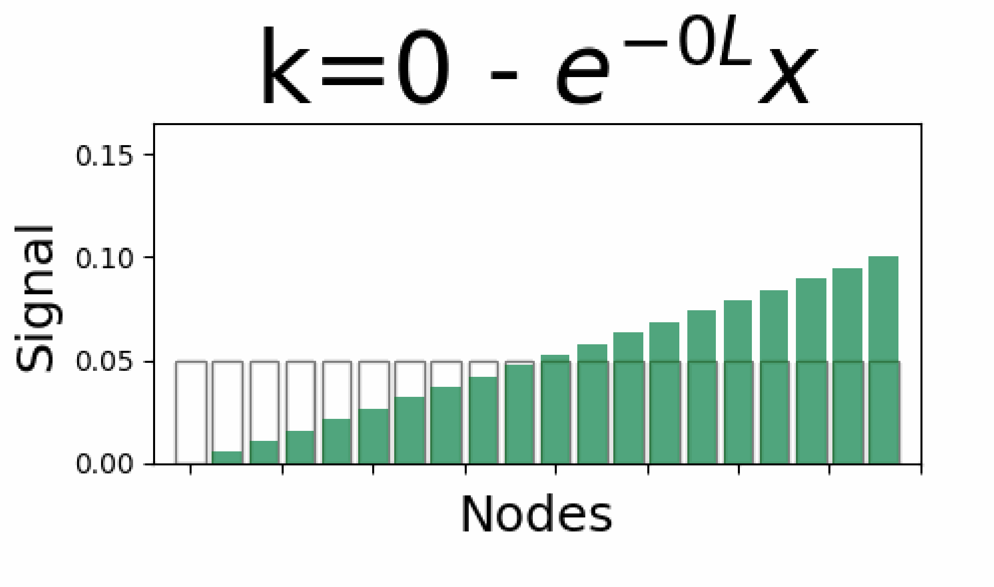
Connection with R_{uv}

- **Over-smoothing as averaging network** [Ghosh et al 2008]
- Discrete diffusion (heat) equation converges to the averaging network at infinite steps

$$\frac{d}{dt}x = -Lx, \quad \text{with solution } x(t) = e^{-tL}x(0)$$

$$\lim_{t\to\infty} x(t) = \lim_{t\to\infty} e^{-tL}x(0) = \frac{\mathbf{1}\mathbf{1}^T x(0)}{n}$$





Reasons for Over-smoothing

Random Walk perspective: averaging network

Connection with R_{uv}

- **Over-smoothing as averaging network** [Ghosh et al 2008]
- Discrete diffusion (heat) equation converges to the averaging network at infinite steps

$$\frac{d}{dt}x = -Lx, \quad \text{with solution } x(t) = e^{-tL}x(0)$$

$$\lim_{t\to\infty} x(t) = \lim_{t\to\infty} e^{-tL}x(0) = \frac{\mathbf{1}\mathbf{1}^T x(0)}{n}$$

- **Rate of convergence**
  - $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$ determine the **rate** at which averaging takes place
  - $\Phi = [\phi_1, \phi_2, \ldots, \phi_n]$ are the **mode** of the system
  - **Time constant** for ϕ_k to decay by a factor e

$$T_k = \frac{1}{\lambda_k}$$

  - **Total effective resistance** is proportional to the sum of time constants → the lower, the faster convergence

$$R_{tot} = \sum_{u\sim v} R_{u,v} = n \sum_{k=2}^{n} \frac{1}{\lambda_k} = n \sum_{k=2}^{n} T_k$$

Does this analysis answer the question?

GCN is augmented heat diffusion process

→*W*

→*σ*

$$PX$$

$$PXW$$

Feature Transformation

$$\sigma(PXW)$$

Non-linear feature transformation

$$\sigma(\sigma(PX)W)$$

Non-linear aggregation

$$\mathrm{GCN}(X, L) = \sigma(\underbrace{\sigma(\cdots\sigma(\sigma(}_{H\ \text{times}} PX)W^1)W^2\cdots)W^k)$$

Reasons for Over-smoothing

GNN architecture perspective

- **Over-smoothing in a GCN with feature transformation and non-linear activation functions (GCN)** [Cai et al 2020]

$$P = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \qquad \mathrm{GCN}(X, L) = \sigma(\underbrace{\sigma(\cdots\sigma(\sigma(}_{H\ \text{times}} PX)W^1)W^2\cdots)W^k)$$

- How does **each component affect the DE** between one layer and the next one? [Oono et al 2019; Cai et al 2020]

$$\mathcal{E}(PX) \leq (1-\lambda_2)^2 \mathcal{E}(X) \qquad \mathcal{E}(XW) \leq \left\|W^T\right\|^2 \mathcal{E}(X) \qquad \mathcal{E}(\sigma(X)) \leq \mathcal{E}(X)$$

ReLu
LeakyReLu

Structure

Weight matrix

Activation

Reasons for Over-smoothing

GNN architecture perspective

- **Over-smoothing in a GCN with feature transformation and non-linear activation functions (GCN)** [Cai et al 2020]

$$P = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \qquad \mathrm{GCN}(X, L) = \sigma(\underbrace{\sigma(\cdots\sigma(\sigma(}_{H\ \text{times}} PX)W^1)W^2\cdots)W^k)$$

- How does each component affect the DE between one layer and the next one? [Oono et al 2019; Cai et al 2020]

$$\mathcal{E}(PX) \leq (1-\lambda_2)^2 \mathcal{E}(X) \qquad \mathcal{E}(XW) \leq \left\|W^T\right\|^2 \mathcal{E}(X) \qquad \mathcal{E}(\sigma(X)) \leq \mathcal{E}(X)$$

ReLu
LeakyReLu

- **DE of one layer is upper-bounded by the previous layer DE** [Zhou et al 2021]
  - **The upper bound depends on the graph connectivity and the structure of the weights**

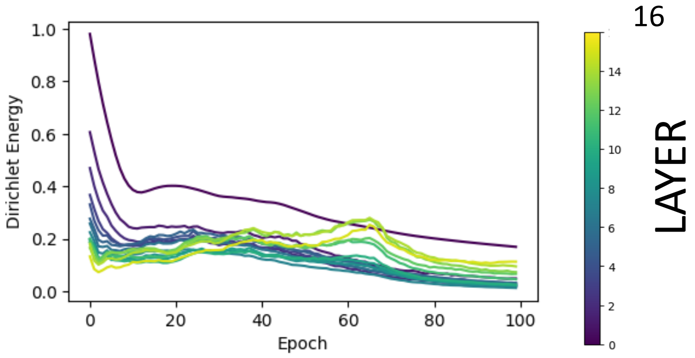
$$\mathcal{E}(H^k) \leq (1-\lambda_2)^2 s^k_{\max} \mathcal{E}(H^{k-1})$$

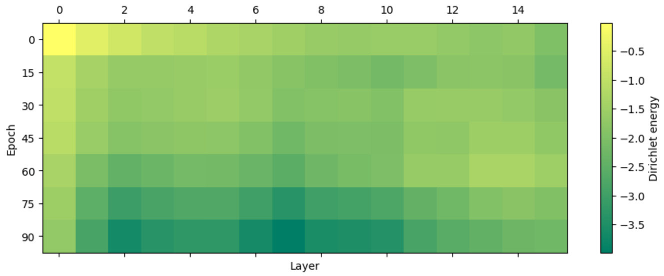
→ Square of maximum singular value of W^k

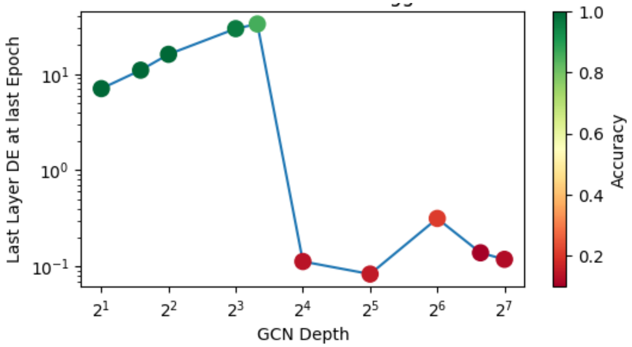
OSM in GCN wrt aggregation function

MEAN aggregation function

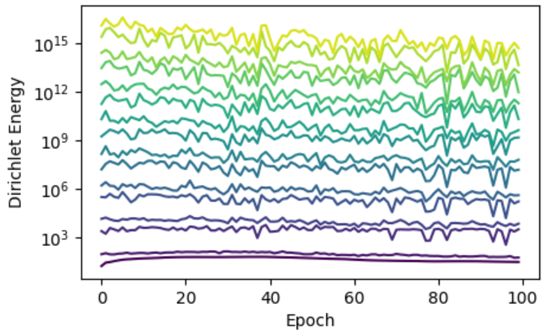
DE converges per epoch and layer

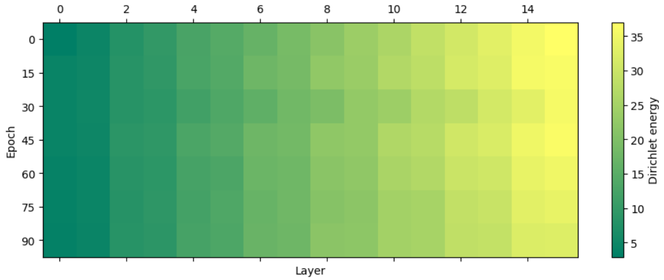


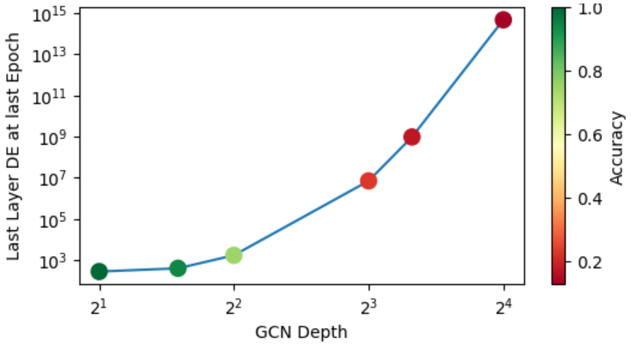




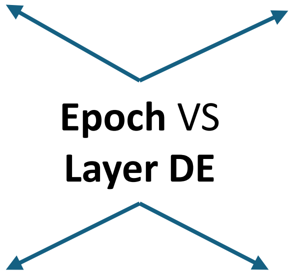
DE slightly converges per epoch but explodes per layer







ADD aggregation function



Epoch VS
Layer DE

GCN Depth VS
Last-layer DE VS
Accuracy

DE does not always correlate with accuracy

Ongoing work with R. Anand

How is OSM manifested in practice?

GNN architecture perspective

- But… Main intuition of Laplacian smoothing (low-pass filters) only proven for
  - non-linear ReLU or LReLU
  - Small weight matrices (measured by their singular values)
  - No residual connections, no normalization, no for all aggregation functions…

- **Dominant frequency explanation** [Di Giovanni 2023, TMLR]
  - Low-Frequency-Dominant (LFD) MPNNs

$$\lim_{k\to\infty} \frac{\mathcal{E}_G(H^k)}{\|H^k\|_2^2} \to 0$$

When?

$$|\mu_{\min}|(\lambda_{\max} - 1) < \mu_{\max}$$

Small eigenvectors are related to smoothing *Homophily*

$\uparrow$ For some *X* $\downarrow$

  - High-Frequency-Dominant (HFD) MPNNs

$$\lim_{k\to\infty} \frac{\mathcal{E}_G(H^k)}{\|H^k\|_2^2} \to \lambda_n$$

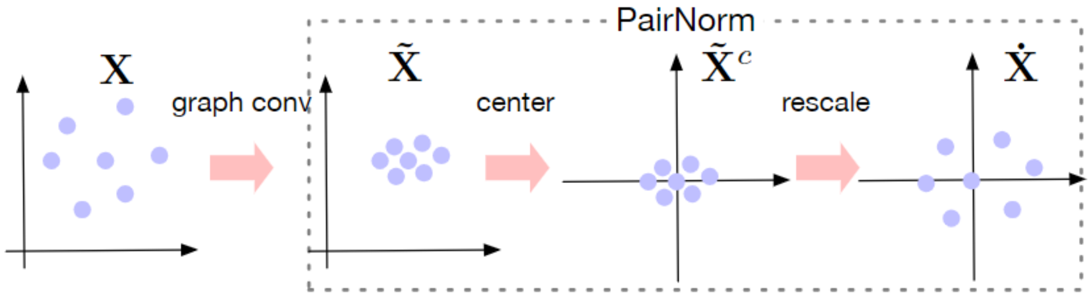
$$|\mu_{\min}|(\lambda_{\max} - 1) > \mu_{\max}$$

High eigenvectors are related to sharpness *Heterophily*

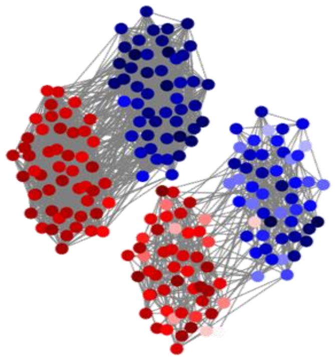
In principle, OSM is mitigated by choosing message passing functions that do not act as low-pass filters

Solutions

- **Normalization of node-embeddings**



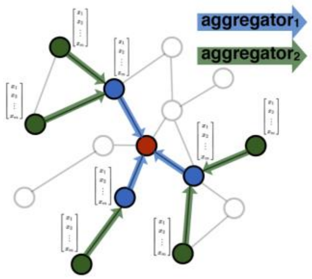
- **Graph Sparsification**



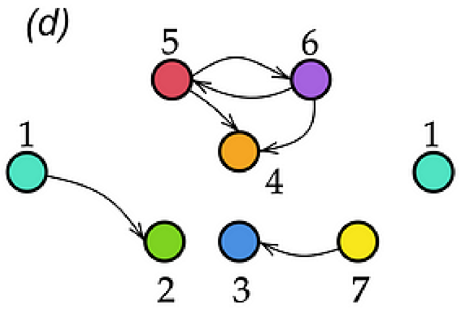
- **Regularization of weight matrix**

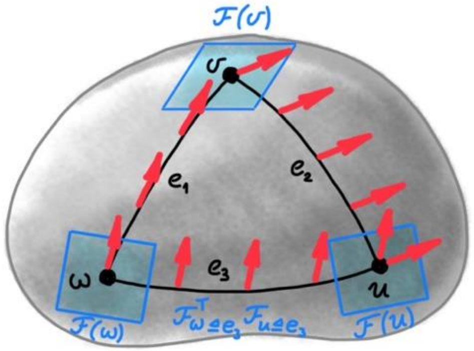
- **Skip-connections**

- **Change GNN Dynamics**
  - GAT, GraphSage
  - Physics inspired GNN
  - Adaptative GNNs



2. Aggregate feature information from neighbors



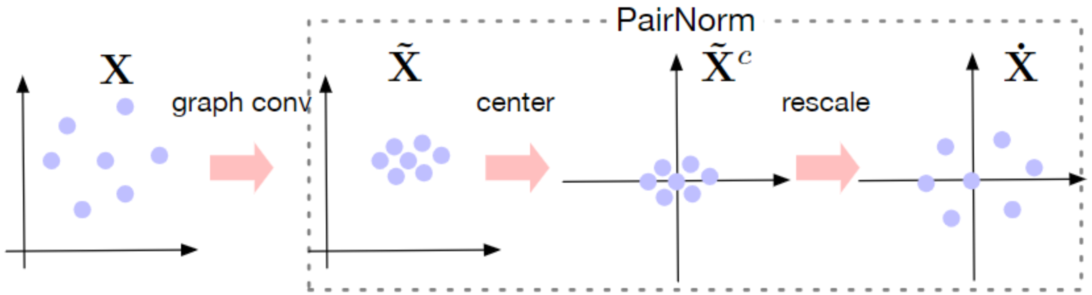


Empirical review of some of the methods and tricks in Chen, et al 2022. “Bag of tricks”

Solutions to Over-smoothing

Normalization of embeddings

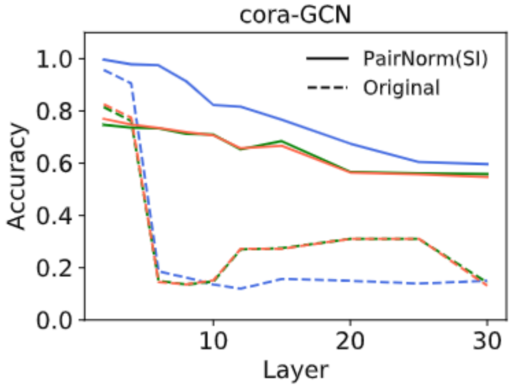
- **Node embedding normalization techniques**.
  - Set distances to be constant throughout every layer in the GNN: **PairNorm** [Zhao et al 2020] or **NodeNorm** [Zhou et al 2021b]



[Zhao et al 2020]

$$\tilde{X}_u^c = \tilde{X}_u - \frac{1}{n}\sum_{u\in\mathcal{V}} \tilde{X}_v$$

$$\dot{X}_u = s\sqrt{n}\frac{\tilde{X}_u^c}{\sqrt{\|\tilde{X}^c\|_2^2}}$$



- Also extensible to group-normalization: DGN [Zhou et al 2020]
  - Learn to maintain the node pair distance in the node batch or group

Solutions to Over-smoothing

Graph Sparsification

- To sparsify a graph reduces the spectral gap, therefore the rate of convergence of node features

$$\frac{1}{R_{uv}}\left(\frac{1}{d_u}+\frac{1}{d_v}\right) \leq \lambda_2$$

[Lovász 1993]

$$\lambda_2 \leq 1 - 2\frac{\sqrt{d_{\max}-1}}{d_{\max}}c$$

Rayleigh Monotonicity principle
When removing edges of the graph, all R_{uv}'s are equal or higher
Sparser graphs → Higher R_{uv}'s → **decreased lower bound**

Max degree lower bound of spectral gap
Sparser graphs → **decreased upper bound**

Solutions to Over-smoothing

Graph Sparsification

- To sparsify a graph reduces the spectral gap, therefore the rate of convergence of node features

$$\frac{1}{R_{uv}}\left(\frac{1}{d_u}+\frac{1}{d_v}\right) \leq \lambda_2$$

[Lovász 1993]

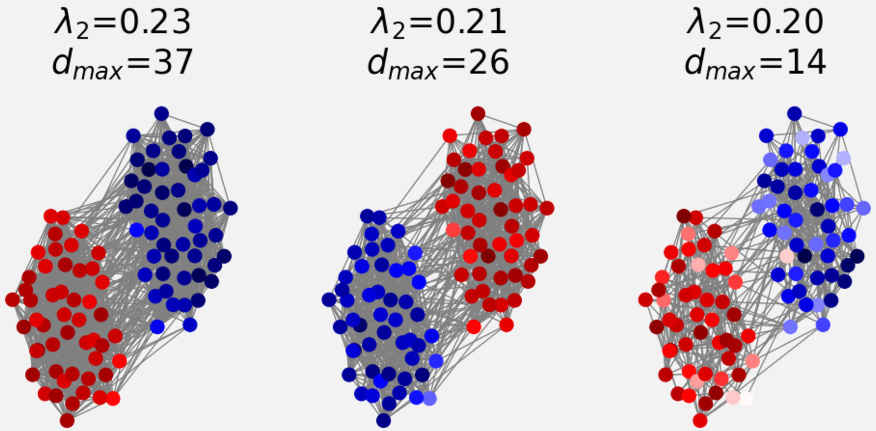
Rayleigh Monotonicity principle
When removing edges of the graph, all R_{uv}'s are equal or higher
Sparser graphs → Higher R_{uv}'s → **decreased lower bound**

$$\lambda_2 \leq 1 - 2\frac{\sqrt{d_{\max}-1}}{d_{\max}}c$$

Max degree lower bound of spectral gap
Sparser graphs → **decreased upper bound**

Strategies

- Random sparsification [Rong et al 2020]
- Drop different edges per feature and learn it via posterior inference [Hasanzadeh et al 2020]
- Neural sparsification [Zheng et al 2020]



Solutions to Over-smoothing

Skip connections

- Skip connections to alleviate information loss [Li et al 2019]

$$H^{l+1} = H^l + \sigma(PH^lW) \qquad\qquad H^{l+1} = H^0 + \sigma(PH^lW)$$

Solutions to Over-smoothing

Skip connections and trainable weights regularization

- Skip connections to alleviate information loss [Li et al 2019]

$$H^{l+1} = H^l + \sigma(PH^lW) \qquad H^{l+1} = H^0 + \sigma(PH^lW)$$

- Parametrized skip connection and identity mapping to reduce the singular values of W.
  GCNII [Chen et al 2020] and **EGNN** [Zhou et al 2021]. Initial Residual connections also present in **APPNP** [Gasteiger et al 2018]

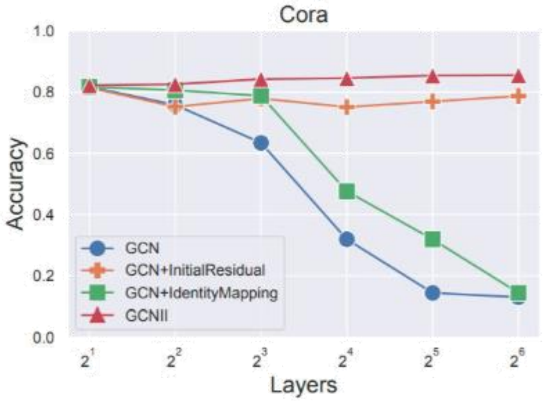
GCNII →

$$H^{l+1} = \sigma\Big(\underbrace{\big((1-\alpha)PH^l + \alpha X\big)}_{\text{Initial residual}} \ \underbrace{\big((1-\beta)I_n + \beta W\big)}_{\text{Identity mapping}} \Big)$$

↓

Reduce the norm and eigenvalues of *W*

$$\mathcal{E}(H^k) \leq (1-\lambda_2)^2 s^k_{\max}\mathcal{E}(H^{k-1})$$



Solutions to Over-smoothing

Skip connections and trainable weights regularization

- Skip connections to alleviate information loss [Li et al 2019]

$$H^{l+1} = H^l + \sigma(PH^lW) \qquad H^{l+1} = H^0 + \sigma(PH^lW)$$

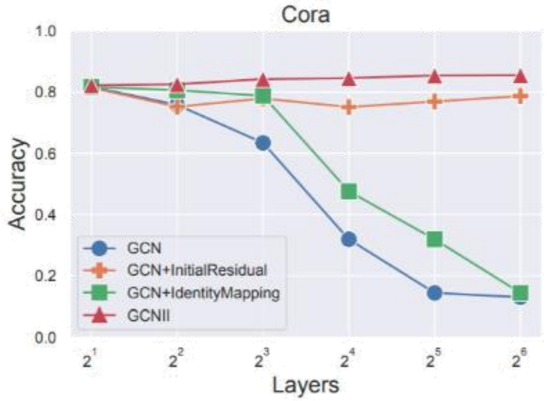
- Parametrized skip connection and identity mapping to reduce the singular values of W. **GCNII** [Chen et al 2020] and **EGNN** [Zhou et al 2021]. Initial Residual connections also present in **APPNP** [Gasteiger et al 2018]

GCNII →

$$H^{l+1} = \sigma\left(\underbrace{((1-\alpha)PH^l + \alpha X)}_{\text{Initial residual}} \; \underbrace{((1-\beta)I_n + \beta W)}_{\text{Identity mapping}} \right)$$

Reduce the norm and eigenvalues of *W*

$$\mathcal{E}(H^k) \leq (1-\lambda_2)^2 s_{\max}^k \mathcal{E}(H^{k-1})$$



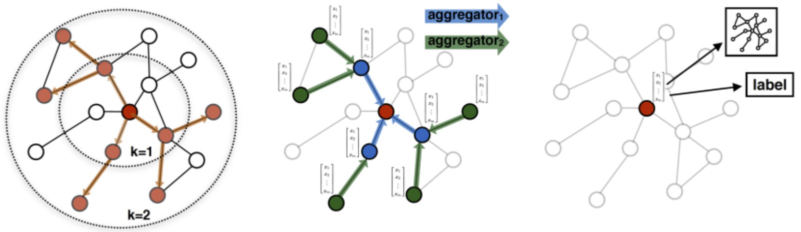
- Combine all hidden node embeddings at the last layer **JKNet** [Xu et al 2018] and **DAGNNs** [Liu et al 2020]

Skip connections can enhance the High-Frequency-Dominant (HFD) MPNNs [Di Giovanni 2023, TMLR]

Advanced GNN Architectures

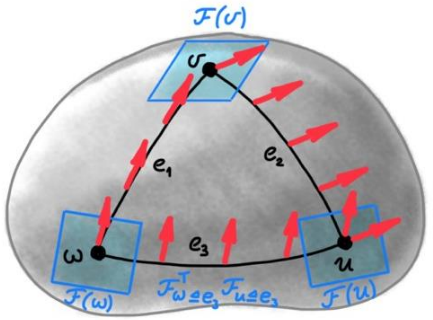
Change GNN Dynamics

- **GraphSAGE** [Hamilton et al., '17], **GAT** [Veličković et al '17]
    - Modify the dynamic of the GNN message passing sampling or learning the messages to aggregate

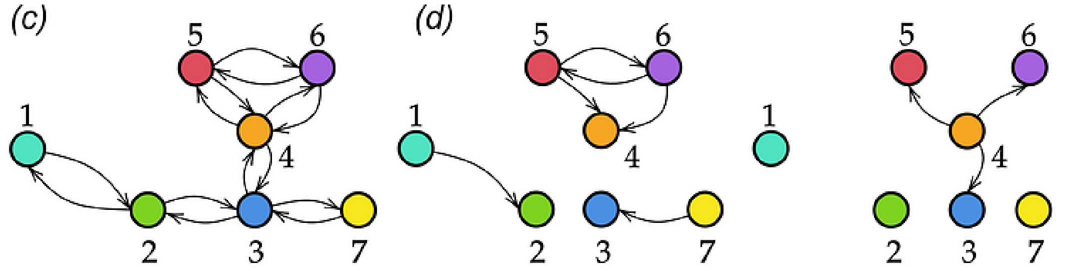


1. Sample neighborhood 2. Aggregate feature information from neighbors 3. Predict graph context and label using aggregated information

- **Physics informed GNNs (PDEs and ODEs) – Time-continuous dynamical GNNs**
    - CGNN [Xhonneux et al 2020], PDE-GCN [Eliasof et al 2021], GRAND [Chamberlain et al 2021b], Neural Sheaf [Bodnar et al 2022], HKGCN [Zhao et al 21], GraphCON [Rusch et al 22], GRAFF [Di Giovanni et al 22], BLEND [Chamberlain et al 22], G-MHKG [Shao et al 23], FLODE [Maskey et al 2024]



- **Adaptative GNNs** [Errica et al 2023]
    - Learn the depth of the network during training
    - Differentiable message filtering



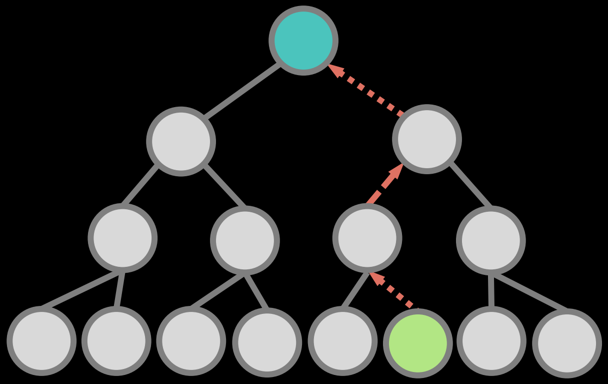
Standard MP (ℓ=1,2) Adaptive MP (ℓ=1) Adaptive MP (ℓ=2)

Recap on Over-smoothing

- Theoretically proved to be caused by
  - Stacking many layers
  - High Graph conductance
  - Structure of the trainable weight matrix

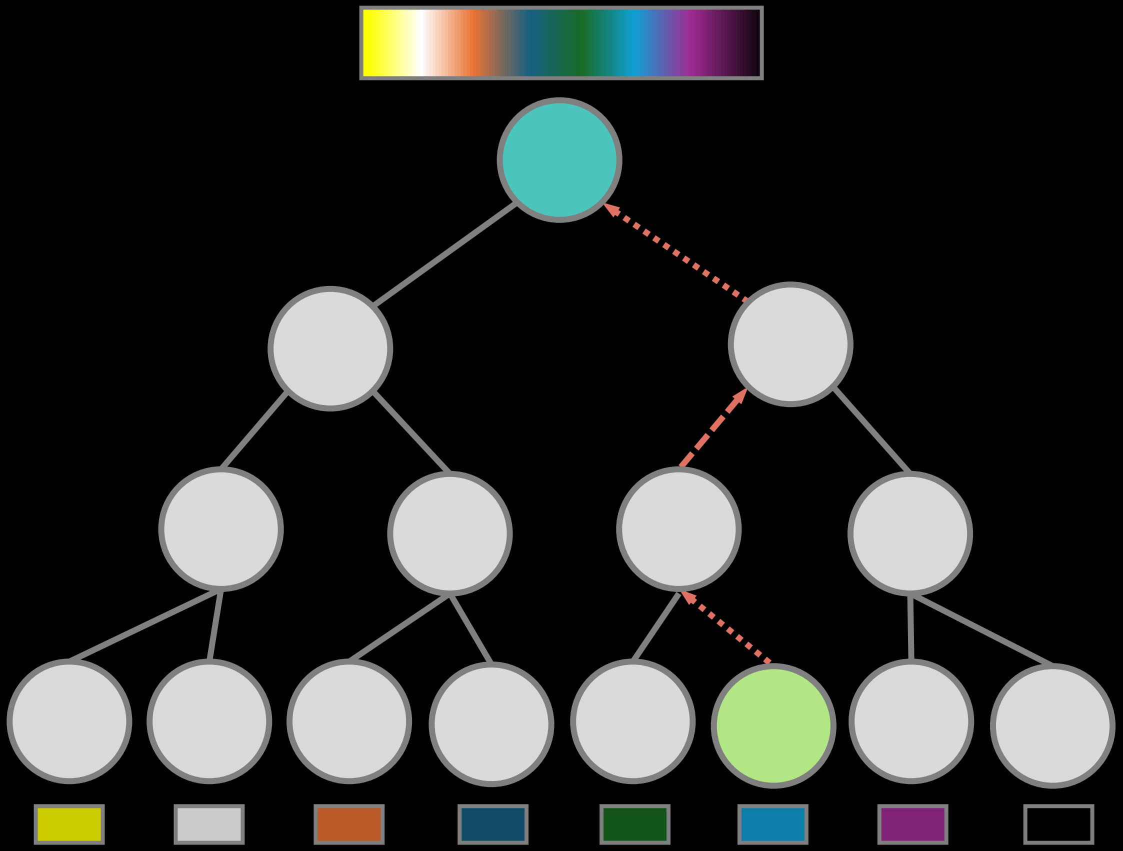
- If the architecture acts as a **low-pass filter** (LFD MPNNs) then there will be OSM
  - Empirically, all the ML tricks (aggregation functions, normalization, self-loops, bias, skip-connections) have a different effect on different graphs

- Mitigated by
  - Sparsification
  - Node embedding normalization and trainable weights regularization
  - Skip-connections
  - Changed GNN dynamics



Over-squashing
OSQ

Over-squashing

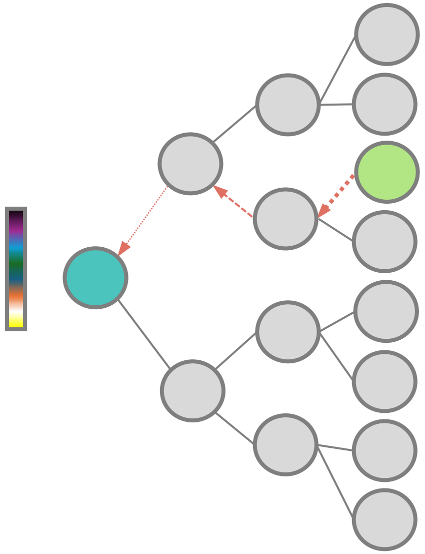


Over-squashing

- **Number of nodes in the receptive field increases exponentially with the depth**
- Neighbors in the *k-hop* increment exponentially with *k*
  - How much does node ***u*** influences node ***v*** when considering all paths of length ***k***? if $\mathrm{SP}(u,v) = r$ in a binary tree, then: [Alon and Yahav 2021; Topping et al 2022]

$$\hat{A} = D^{1/2}(A+I)D^{1/2} \qquad \hat{A}^{r}_{uv} = \frac{1}{2} \cdot 3^{-(r-1)}$$

Normalized connection strength between u and v

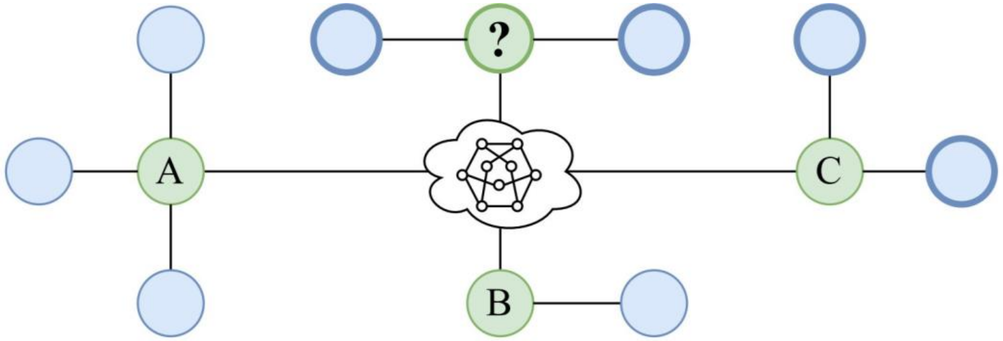


- Compressed into fixed-size vector $X_i =$
  - No longer sensitive in relative terms

- Therefore**, if there is bottlenecks in the graph**, all information is compressed and have to pass through that bottleneck → exponential compression
  - Long-range fails **X**

Over-squashing

NeighborsMatch problem



[Alon and Yahav 2021]

- Synthetic benchmark for controlling over-squashing
- Tree problem with controllable depth
- Training accuracy drops with depth
- Some types of GNNs more susceptible to over-squashing
- *+FA* propose a Full Connected graph in the last MP layer

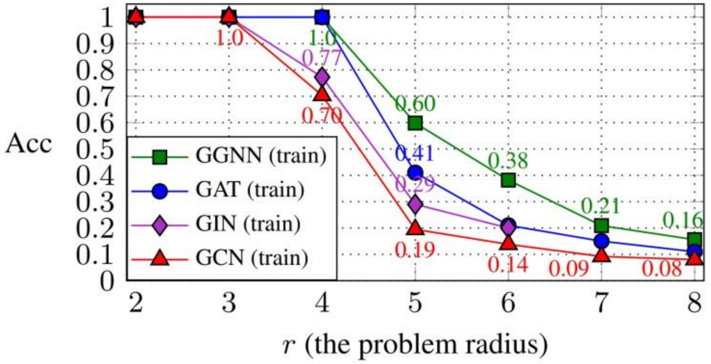


Figure 3: Accuracy across *problem radius* (tree depth) in the NEIGHBORSMATCH problem. Over-squashing starts to affect GCN and GIN even at $r = 4$.

How to Measure Over-squashing?

How to measure the bottlenecks in the graph?

- **Cheeger constant and bottleneck**
  [Topping et al ‘22; Arnaiz-Rodriguez et al ‘22; Banjeree et al ‘22]

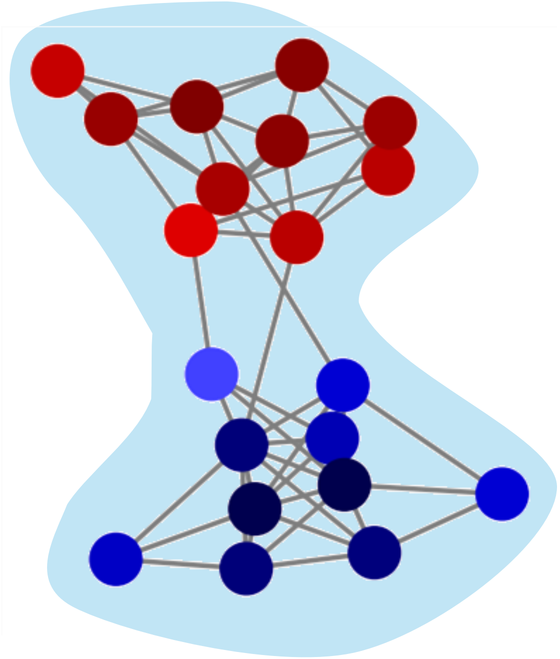
- **Effective resistance / Commute Times**
  [Arnaiz-Rodriguez et al '22 ; Banjeree et al ‘22; Di Giovanni et al., '23; Black et al., ‘23]

- **Jacobian** *(Sensitivity analysis)*
  [Xu et al '18; Di Giovanni et al '23; Black et al '23]

- **Curvature** *(Balanced Forman, Ollivier)*
  [Topping et al '22]

- **Hessian measure**
  [Di Giovanni and Rusch et al. '24]

Betweenness centrality?



Measures of OSQ

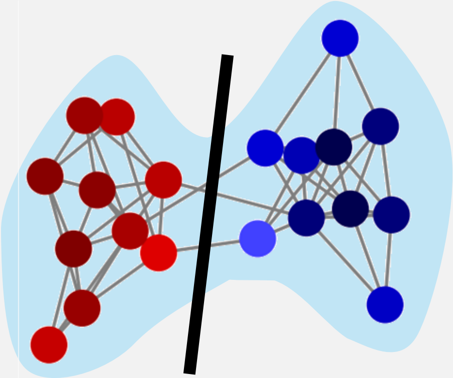
Cheeger constant, Spectral Gap and Effective Resistance

- **Cheeger constant**
  measures the **MinCut** to disconnect the graph

$$\partial S = \{e = (u, v) : u \in S, v \in \bar{S}\}$$

$$h_S = \frac{|\partial S|}{\min\{\mathrm{vol}(S),\ \mathrm{vol}(\bar{S})\}}$$

$$h_G = \min_{S \subset V} h_S$$



[Lovász 1993; Chung 1997; Qiu and Hancock 2007] [Topping et al ‘22; Arnaiz-Rodriguez et al ‘22; Banjeree et al ‘22]

Measures of OSQ

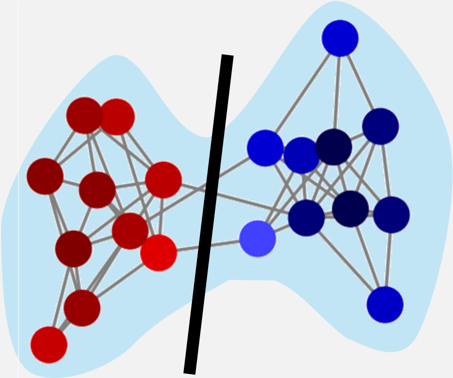
Cheeger constant, Spectral Gap and Effective Resistance

- **Cheeger constant**
  measures the **MinCut** to disconnect the graph

$$\partial S = \{e = (u, v) : u \in S, v \in \bar{S}\}$$

$$h_S = \frac{|\partial S|}{\min\{\mathrm{vol}(S),\ \mathrm{vol}(\bar{S})\}}$$

$$h_G = \min_{S \subset V} h_S$$



- **Cheeger Inequality**

$$\frac{h_G^2}{2} \le \lambda_2 \le 2h_G \qquad \frac{\lambda_2}{2} \le h_G \le \sqrt{2\lambda_2}$$

[Lovász 1993; Chung 1997; Qiu and Hancock 2007] [Topping et al '22; Arnaiz-Rodriguez et al '22; Banjeree et al '22]

Measures of OSQ

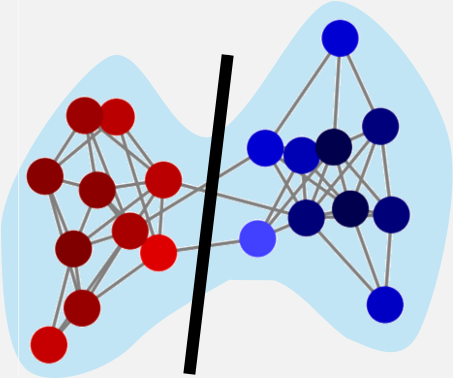
Cheeger constant, Spectral Gap and Effective Resistance

- **Cheeger constant**
  measures the **MinCut** to disconnect the graph

$$\partial S = \{e = (u,v) : u \in S, v \in \bar{S}\}$$

$$h_S = \frac{|\partial S|}{\min\{\mathrm{vol}(S),\ \mathrm{vol}(\bar{S})\}}$$

$$h_G = \min_{S \subset V} h_S$$



- **Cheeger Inequality**

$$\frac{h_G^2}{2} \leq \lambda_2 \leq 2h_G \qquad \frac{\lambda_2}{2} \leq h_G \leq \sqrt{2\lambda_2}$$

- **Cheeger constant and Effective Resistance**

$$R_{\max} \leq \frac{1}{h_G^2}$$

[Lovász 1993; Chung 1997; Qiu and Hancock 2007] [Topping et al ‘22; Arnaiz-Rodriguez et al ‘22; Banjeree et al ‘22]

Measures of OSQ

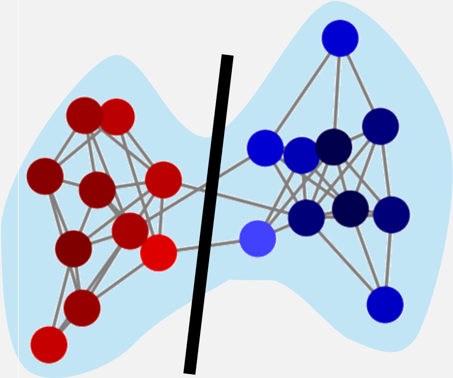
Cheeger constant, Spectral Gap and Effective Resistance

- **Cheeger constant**
  measures the **MinCut** to disconnect the graph

$$\partial S = \{e = (u,v) : u \in S, v \in \bar{S}\}$$

$$h_S = \frac{|\partial S|}{\min\{\mathrm{vol}(S),\ \mathrm{vol}(\bar{S})\}}$$

$$h_G = \min_{S \subset V} h_S$$



- **Cheeger Inequality**

$$\frac{h_G^2}{2} \leq \lambda_2 \leq 2h_G \qquad \frac{\lambda_2}{2} \leq h_G \leq \sqrt{2\lambda_2}$$

- **Cheeger constant and Effective Resistance**

$$R_{\max} \leq \frac{1}{h_G^2}$$

- **Spectral Gap and Effective Resistance**
  - *Lovász Bound*

$$\left| R_{uv} - \left(\frac{1}{d_u} - \frac{1}{d_v} \right) \right| \leq \frac{2}{\lambda_2 d_{min}}$$

  - *More ER-related bounds*

$$\frac{1}{n\lambda_2} \leq R_{\max} \leq \frac{2}{\lambda_2} \qquad \frac{n}{\lambda_2} \leq R_{\mathrm{tot}} \leq \frac{n(n-1)}{\lambda_2}$$

[Lovász 1993; Chung 1997; Qiu and Hancock 2007] [Topping et al '22; Arnaiz-Rodriguez et al '22; Banjeree et al '22]

Measure Over-squashing

Sensitivity between node embeddings

- How much do the original features of the ***u*** node affects the features of node ***v*** after ***m*** layers? [Xu et al 2018; Hamilton 2020]
  Influence Score

$$\left\| \frac{\partial \mathbf{h}_v^{(r)}}{\partial \mathbf{h}_u^{(0)}} \right\| \propto P_{G,K}(u|v)$$

Measure Over-squashing

Sensitivity between node embeddings

- How much do the original features of the ***u*** node affects the features of node ***v*** after ***m*** layers? ***Influence Score***

[Xu et al 2018; Hamilton 2020]

$$\left\|\frac{\partial \mathbf{h}_v^{(r)}}{\partial \mathbf{h}_u^{(0)}}\right\| \propto P_{G,K}(u|v)$$

$$\min\left\{r : \left\|\frac{\partial \mathbf{h}_v^{(r)}}{\partial \mathbf{h}_u^{(l_0)}}\right\| \neq 0\right\} \geq l_0 + \mathrm{SP}(u,v)$$

Connection to under-reaching

[Gutteridge et al 2023]

Measure Over-squashing

Sensitivity between node embeddings

- How much do the original features of the ***u*** node affects the features of node ***v*** after ***m*** layers?
  Influence Score [Xu et al 2018; Hamilton 2020]

$$\left\| \frac{\partial \mathbf{h}_v^{(r)}}{\partial \mathbf{h}_u^{(0)}} \right\| \propto P_{G,K}(u|v)$$

- How to measure the overall sensitivity between both nodes? [Di Giovanni et al 2023; Black et al 2023]

$$\tilde{\mathbf{J}}_k^{(m)}(v,u) := \boxed{\left(\frac{1}{d_v} \frac{\partial \mathbf{h}_v^{(m)}}{\partial \mathbf{h}_v^{(k)}} - \frac{1}{\sqrt{d_v d_u}} \frac{\partial \mathbf{h}_v^{(m)}}{\partial \mathbf{h}_u^{(k)}} \right)} + \cdots$$

Δ *node* ***v*** *self-sensitivity* VS ***u->v*** *sensitivity*
from layer ***k*** *features to layer* ***m*** *features*

Measure Over-squashing

Sensitivity between node embeddings

- How much do the original features of the ***u*** node affects the features of node ***v*** after ***m*** layers? [Xu et al 2018; Hamilton 2020]
  Influence Score

$$\left\| \frac{\partial \mathbf{h}_v^{(r)}}{\partial \mathbf{h}_u^{(0)}} \right\| \propto P_{G,K}(u|v)$$

- How to measure the overall sensitivity between both nodes? [Di Giovanni et al 2023; Black et al 2023]

$$\tilde{\mathbf{J}}_k^{(m)}(v,u) := \boxed{\left(\frac{1}{d_v} \frac{\partial \mathbf{h}_v^{(m)}}{\partial \mathbf{h}_v^{(k)}} - \frac{1}{\sqrt{d_v d_u}} \frac{\partial \mathbf{h}_v^{(m)}}{\partial \mathbf{h}_u^{(k)}} \right)} + \left(\frac{1}{d_u} \frac{\partial \mathbf{h}_u^{(m)}}{\partial \mathbf{h}_u^{(k)}} - \frac{1}{\sqrt{d_v d_u}} \frac{\partial \mathbf{h}_u^{(m)}}{\partial \mathbf{h}_v^{(k)}} \right)$$

Δ ***node v self-sensitivity*** VS ***u->v sensitivity from layer k features to layer m features***

Same but reversed for v->u

Measure Over-squashing

Sensitivity between node embeddings

- How much do the original features of the ***u*** node affects the features of node ***v*** after ***m*** layers?
  Influence Score [Xu et al 2018; Hamilton 2020]

$$\left\| \frac{\partial \mathbf{h}_v^{(r)}}{\partial \mathbf{h}_u^{(0)}} \right\| \propto P_{G,K}(u|v)$$

- How to measure the overall sensitivity between both nodes? [Di Giovanni et al 2023; Black et al 2023]

$$\underbrace{\tilde{\mathbf{J}}_k^{(m)}(v,u) := \boxed{\left(\frac{1}{d_v}\frac{\partial \mathbf{h}_v^{(m)}}{\partial \mathbf{h}_v^{(k)}} - \frac{1}{\sqrt{d_v d_u}}\frac{\partial \mathbf{h}_v^{(m)}}{\partial \mathbf{h}_u^{(k)}}\right)} + \left(\frac{1}{d_u}\frac{\partial \mathbf{h}_u^{(m)}}{\partial \mathbf{h}_u^{(k)}} - \frac{1}{\sqrt{d_v d_u}}\frac{\partial \mathbf{h}_u^{(m)}}{\partial \mathbf{h}_v^{(k)}}\right)}$$

Δ ***node* v *self-sensitivity*** VS **u->v *sensitivity from layer* k *features to layer* m *features***

***Same but reversed for* v->u**

Symmetric Jacobian Obstruction:

Symmetric Δ *self-sensitivity* VS *pairwise sensitivity from layer-**k**’s features to layer-**m**’s features*

Measure Over-squashing

Sensitivity between node embeddings

- How much do the original features of the ***u*** node affects the features of node ***v*** after ***m*** layers?
  Influence Score [Xu et al 2018; Hamilton 2020]

$$\left\| \frac{\partial \mathbf{h}_v^{(r)}}{\partial \mathbf{h}_u^{(0)}} \right\| \propto P_{G,K}(u|v)$$

- How to measure the overall sensitivity between both nodes? [Di Giovanni et al 2023; Black et al 2023]

$$\tilde{\mathbf{J}}_k^{(m)}(v,u) := \underbrace{\boxed{\left(\frac{1}{d_v}\frac{\partial \mathbf{h}_v^{(m)}}{\partial \mathbf{h}_v^{(k)}} - \frac{1}{\sqrt{d_v d_u}}\frac{\partial \mathbf{h}_v^{(m)}}{\partial \mathbf{h}_u^{(k)}}\right)} + \left(\frac{1}{d_u}\frac{\partial \mathbf{h}_u^{(m)}}{\partial \mathbf{h}_u^{(k)}} - \frac{1}{\sqrt{d_v d_u}}\frac{\partial \mathbf{h}_u^{(m)}}{\partial \mathbf{h}_v^{(k)}}\right)}$$

Δ ***node v self-sensitivity*** VS ***u->v sensitivity from layer k features to layer m features***

Same but reversed for v->u

Symmetric Jacobian Obstruction:
Symmetric Δ *self-sensitivity* VS *pairwise sensitivity from layer-**k**'s features to layer-**m**'s features*

- Extension to *m* layers

$$\tilde{\mathrm{O}}^m(u,v) = \sum_{k=0}^{m} \left\| \tilde{\mathbf{J}}_k^{(m)}(v,u) \right\|$$

Symmetric Jacobian Obstruction after *m* layers

Measure Over-squashing

Sensitivity between node embeddings – Bounds

- How much do the original features of the ***u*** node affects the features of node ***v*** after ***m*** layers? [Di Giovanni et al 2023; Black et al 2023] → Bounded by topology

$$\left\| \frac{\partial \mathbf{h}_v^{(r)}}{\partial \mathbf{h}_u^{(0)}} \right\| \leq \left(\prod_{k=1}^{r} \|\phi_k\| \, \|\psi_k\| \right) (\hat{\mathbf{A}}^{\mathbf{r}})_{uv} \qquad \left\| \frac{\partial \mathbf{h}_u^{(r)}}{\partial \mathbf{h}_v^{(0)}} \right\| \leq c \, (\hat{\mathbf{A}}^{\mathbf{r}})_{uv}$$

Normalized # paths of length ***r*** between ***u*** and ***v***

Measure Over-squashing

$$R_{u,v} = \sum_{j=0}^{\infty} \left(\frac{(\hat{A}^i)_{uu}}{d_u} + \frac{(\hat{A}^i)_{vv}}{d_v} - \frac{2}{\sqrt{d_u d_v}} (\hat{A}^i)_{uv} \right)$$

Sensitivity between node embeddings – Bounds

- How much do the original features of the ***u*** node affects the features of node ***v*** after ***m*** layers? [Di Giovanni et al 2023; Black et al 2023] → Bounded by topology

$$\left\| \frac{\partial \mathbf{h}_v^{(r)}}{\partial \mathbf{h}_u^{(0)}} \right\| \leq \left(\prod_{k=1}^{r} \|\phi_k\| \, \|\psi_k\| \right) (\hat{\mathbf{A}}^{\mathbf{r}})_{uv} \qquad \left\| \frac{\partial \mathbf{h}_u^{(r)}}{\partial \mathbf{h}_v^{(0)}} \right\| \leq c \; (\hat{\mathbf{A}}^{\mathbf{r}})_{uv}$$

Normalized # paths of length ***r*** between ***u*** and ***v***

- **Jacobian obstruction and sum of pairwise jacobians** → Bounded by ER

$$\tilde{\mathrm{O}}^m(u,v) = \sum_{k=0}^{m} \left\| \tilde{\mathbf{J}}_k^{(m)}(v,u) \right\| \leq c \; R_{u,v} \qquad \sum_{u,v \in V \times V} \left\| \frac{\partial h_v^{(r)}}{\partial h_u^{(0)}} \right\| \leq c(b - R_{tot})$$

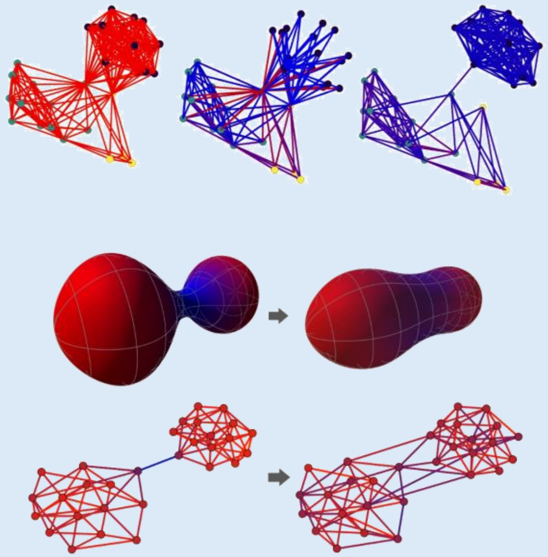
The larger **Effective Resistance** is,
the higher the **Symmetric Jacobian Obstruction**
[Di Giovanni et al 2023]

The larger **Total Effective Resistance** is,
the lower the **sum of pairwise jacobians**
[Black et al 2023]

Connection to what functions can be learned by a MPNN in [Di Giovanni et al 2024]

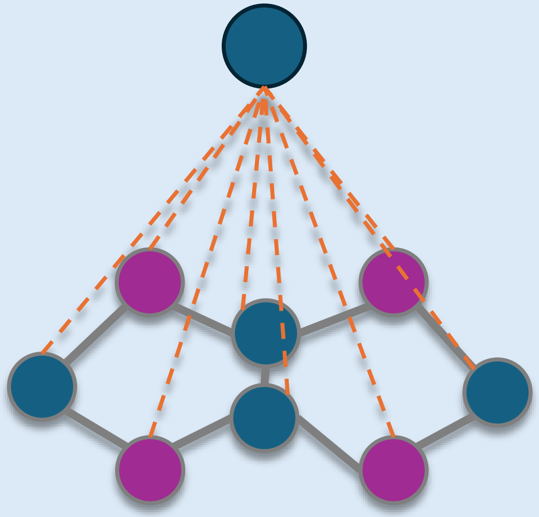
Solutions

GRAPH REWIRING

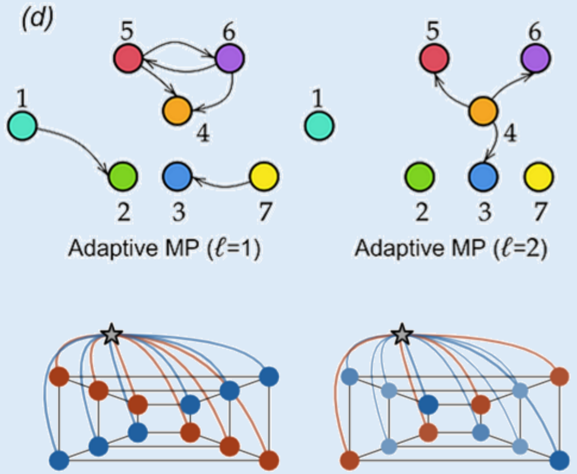


More:
- Multi-hop architectures
- Transformers

Virtual Nodes



Advanced GNN Architectures



- Adaptative
- Transformers
- Advanced Spectral Filters
- Physics-Informed Methods

Figures from Topping et al 2022, Arnaiz-Rodriguez et al 2022, Errica et al 2023, Geisler et al 2024

Graph Rewiring

- Change the edges of the graph such that message passing mechanism is affected

$$\mathcal{R}(G) = (V, \mathcal{R}(E))$$

$$Y = \mathrm{GNN}(\mathcal{R}(G))$$

$$\mathbf{h}_u^{(l+1)} = \phi\left(\mathbf{h}_u^l, \bigoplus_{(u,v)\in\mathcal{R}(E)} \psi\left(\mathbf{h}_u^l, \mathbf{h}_v^l\right)\right)$$

Connecting long-distance nodes

Reduce bottlenecks

Increase sensitivity between distant and relevant nodes

Graph Rewiring

- Change the edges of the graph such that message passing mechanism is affected

$$\mathcal{R}(G) = (V, \mathcal{R}(E))$$

$$Y = \mathrm{GNN}(\mathcal{R}(G))$$

$$\mathbf{h}_u^{(l+1)} = \phi\left(\mathbf{h}_u^l, \bigoplus_{(u,v)\in\mathcal{R}(E)} \psi\left(\mathbf{h}_u^l, \mathbf{h}_v^l\right)\right)$$

Connecting long-distance nodes

Reduce bottlenecks

Increase sensitivity between distant and relevant nodes

- **Spatial** vs **Spectral**

Spatial Add edges within a certain ***k-hop*** **(locality)**	$\mathrm{diam}(\mathcal{R}(G)) \leq \mathrm{diam}(G)$

Spectral Add edges based on a **global spectral measure (connectivity)**	$h_{\mathcal{R}(G)} \geq h_G$ $R_{\mathrm{tot}}(\mathcal{R}(G)) \leq R_{\mathrm{tot}}(G)$

Both Reduce the ER of G

[Di Giovanni et al 2023]

Graph Rewiring

- Change the edges of the graph such that message passing mechanism is affected

$$\mathcal{R}(G) = (V, \mathcal{R}(E))$$

$$Y = \mathrm{GNN}(\mathcal{R}(G))$$

$$\mathbf{h}_u^{(l+1)} = \phi\left(\mathbf{h}_u^l, \bigoplus_{(u,v)\in\mathcal{R}(E)} \psi\left(\mathbf{h}_u^l, \mathbf{h}_v^l\right)\right)$$

Connecting long-distance nodes

Reduce bottlenecks

Increase sensitivity between distant and relevant nodes

- **Spatial** vs **Spectral**

Spatial Add edges within a certain ***k-hop*** **(locality)**	$\mathrm{diam}(\mathcal{R}(G)) \leq \mathrm{diam}(G)$

Spectral Add edges based on a **global spectral measure (connectivity)**	$h_{\mathcal{R}(G)} \geq h_G$ $R_{\mathrm{tot}}(\mathcal{R}(G)) \leq R_{\mathrm{tot}}(G)$

Both Reduce the ER of G [Di Giovanni et al 2023]

- **Static** vs **Dynamic** $\mathcal{R}^t(G) = (V, \mathcal{R}^t(E))$ Nodes do not always interact with the same delay [Gutteridge et al 2023]

Graph Rewiring

- Change the edges of the graph such that message passing mechanism is affected

$$\mathcal{R}(G) = (V, \mathcal{R}(E))$$

$$Y = \mathrm{GNN}(\mathcal{R}(G))$$

$$\mathbf{h}_u^{(l+1)} = \phi\left(\mathbf{h}_u^l, \bigoplus_{(u,v)\in\mathcal{R}(E)} \psi\left(\mathbf{h}_u^l, \mathbf{h}_v^l\right)\right)$$

Connecting long-distance nodes

Reduce bottlenecks

Increase sensitivity between distant and relevant nodes

- **Spatial** vs **Spectral**

Spatial Add edges within a certain ***k-hop*** **(locality)**	$\mathrm{diam}(\mathcal{R}(G)) \leq \mathrm{diam}(G)$

Spectral Add edges based on a **global spectral measure (connectivity)**	$h_{\mathcal{R}(G)} \geq h_G$ $R_{\mathrm{tot}}(\mathcal{R}(G)) \leq R_{\mathrm{tot}}(G)$

Both Reduce the ER of G [Di Giovanni et al 2023]

- **Static** vs **Dynamic** $\mathcal{R}^t(G) = (V, \mathcal{R}^t(E))$ Nodes do not always interact with the same delay [Gutteridge et al 2023]

- **Pre-processing** vs **In-processing** (differentiable and data-driven) [Arnaiz-Rodriguez et al 2022]

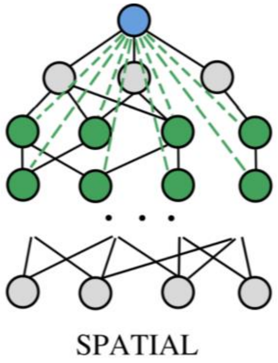
$$Y = \mathrm{GNN}(G) = \cdots \mathcal{R}(G) \cdots$$

Rewiring is learned during the GNN training

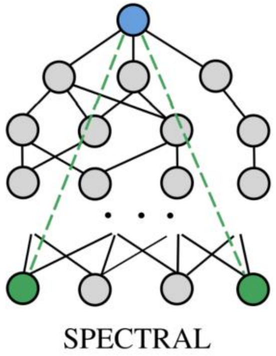
Graph Rewiring

Spatial vs Spectral

$$\mathrm{diam}(\mathcal{R}(G)) \leq \mathrm{diam}(G)$$



$$R_{\mathrm{tot}}(\mathcal{R}(G)) \leq R_{\mathrm{tot}}(G)$$



$$h_{\mathcal{R}(G)} \geq h_G$$

Spatial

Add edges within a certain ***k-hop*** (**locality**). Also, **multi-hop architectures** (A^k) and **transformers** (full connected)

❌ Need for very dense graphs to solve OSQ

✅ Preserve locality

Spectral

Add edges based on a **global spectral measure** (**connectivity**)

✅ Preserve sparsity

❌ Does not maintain the locality information

Figure from [Barbero et al 2024]

Graph Rewiring

Spatial vs Spectral

Spatial

Rewiring:

Based on ***Curvature***: SDRF [Topping et al 22]
Based on ***random edges***: G-RLEF [Banerjee et al 22]

High-order networks:

SPN [Abboud et al. 22], Mix-Hop [Abu-El-Haija et al 19], H2GNN [Zhue et al 20], DHGR [Bi et al 22], DRew [Gutteridge et al 23], GREET [Liu et al 23b]

Transformers and Positoinal Encodings (PE):

PE [Brüel-Gabrielsson et al 23], Graphormer [Ying et al, '21], SAN [Kreuzer et al, '21], GraphGPS [Rampášek et al, '22], Exphormer [Shirzad, Velingker, Venkatachalam et al, '23]

Spectral

Rewiring:

*Based on **PageRank smoothing***: DIGL [Gasteiger et al 19]

Learnable Effective Resistance: DiffWire (Data-driven rewiring) [Arnaiz-Rodriguez et al 22]

Increase approximately λ_2: FOSR [Karhadkar et al. 22]

Cayley expander graphs: EGP [Deac et al. 22]

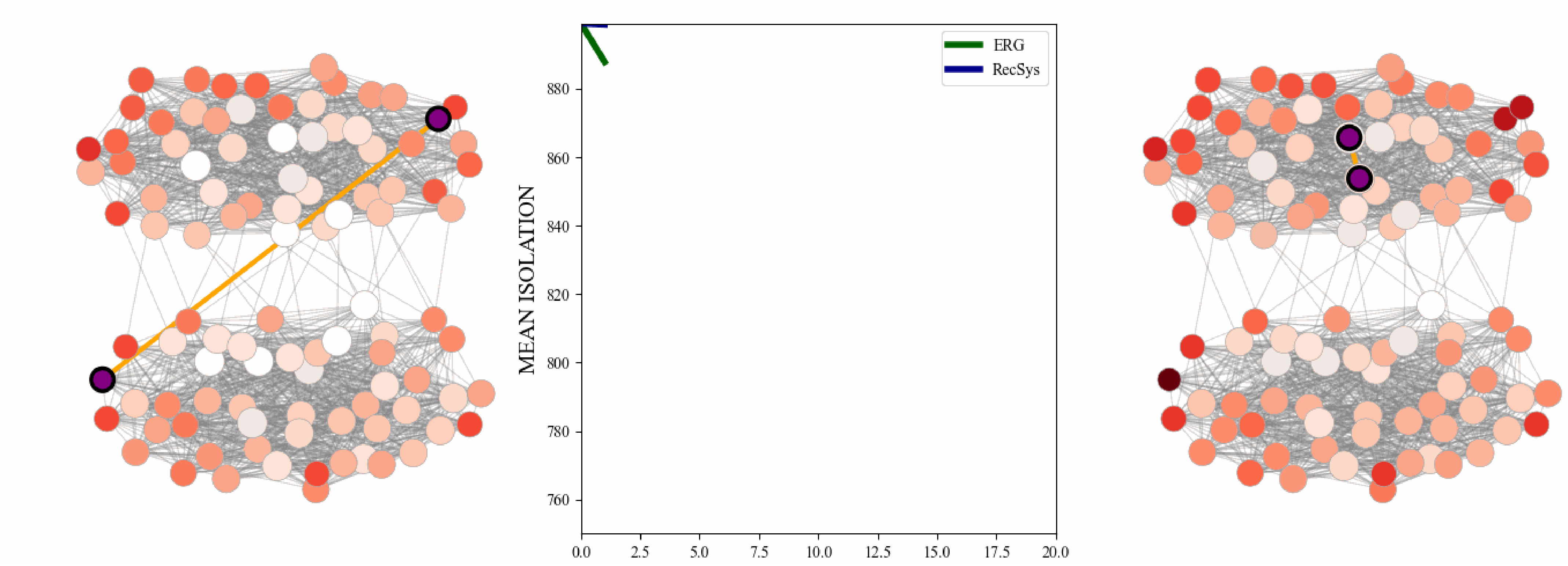
Precomputed Effective Resistance: GTR [Black et al 23]

Spatio-Spectral

LASER [Barbero et al 2023]
Spatio-Spectal GNNs[Geisler et al 2024] – GNN

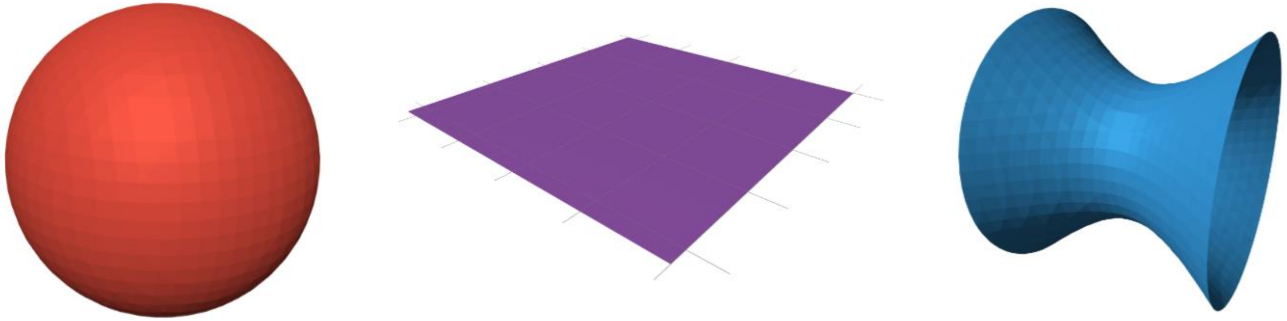
Graph Rewiring

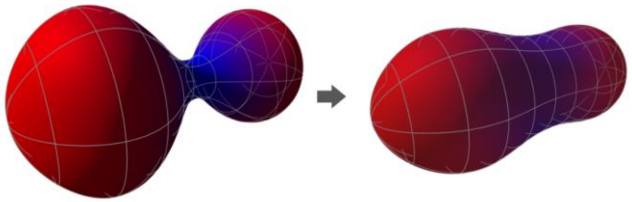
Spectral VS Spatial



Spatial Rewiring - Curvature

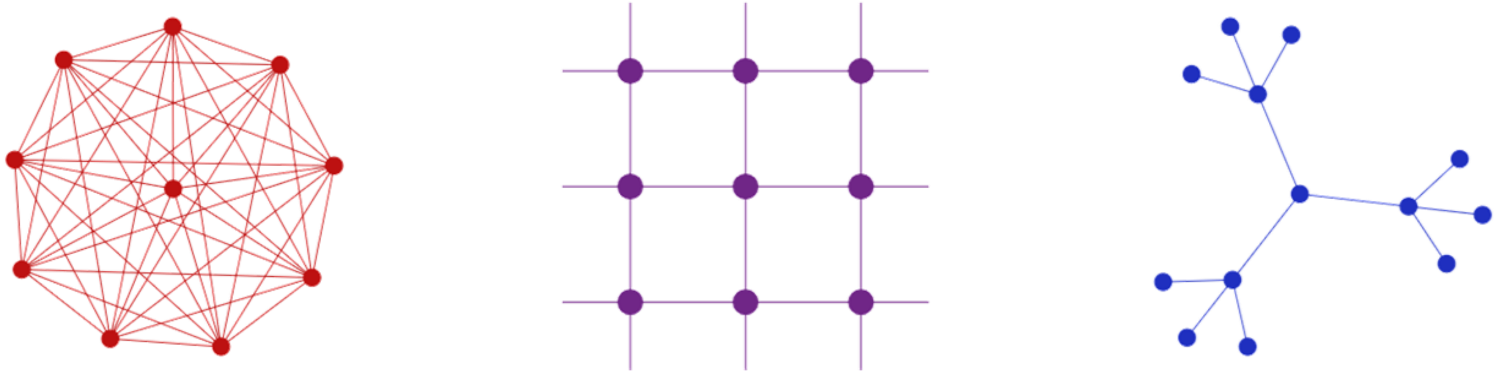
SDRF [Topping et al 2022]

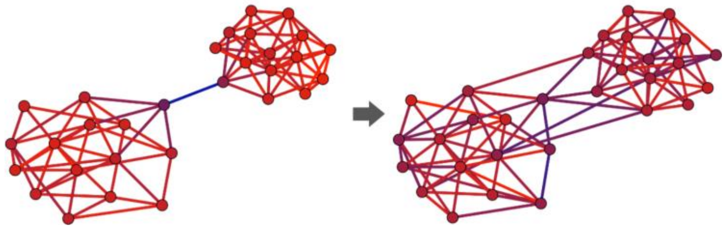




Spatial Rewiring - Curvature

SDRF [Topping et al 2022]





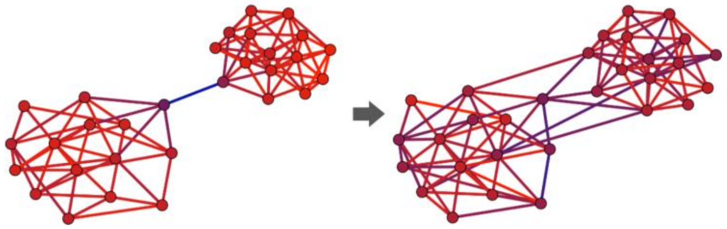
Spatial Rewiring - Curvature

SDRF [Topping et al 2022]

- How to identify bottleneck? **Edges with lowest Ricci Curvature** (Balanced Forman as lower bound)

$$\mathrm{Ric}(i,j) := \frac{2}{d_i} + \frac{2}{d_j} - 2 + 2\frac{|\#_\Delta(i,j)|}{\max\{d_i,d_j\}} + \frac{|\#_\Delta(i,j)|}{\min\{d_i,d_j\}} + \frac{(\gamma_{\max})^{-1}}{\max\{d_i,d_j\}}\left(|\#^i_\square| + |\#^j_\square|\right)$$

- How to fix bottlenecks? **Add edges around edges with low curvature**
  1. Identify edge $\boldsymbol{e_{min}}$ with lowest Ricci Curvature
  2. Add edge between 2 $(\boldsymbol{k}, \boldsymbol{l})$ neighbors of the endpoints $\boldsymbol{e_{min}}$
     sampled with probability proportional to the improvement of the curvature of $\boldsymbol{e_{min}}$ after adding $(\boldsymbol{k}, \boldsymbol{l})$
  3. Remove edge $\boldsymbol{e_{max}}$ with highest Ricci Curvature



Spectral Rewiring – Differentiable

DiffWire - Background

[Doyle and Snell, 1984]

Effective Resistance

$$R_{u,v} = \sum_{i=1}^{n} \frac{1}{\lambda_i} \left(\frac{\phi_i(u)}{\sqrt{d_u}} - \frac{\phi_i(v)}{\sqrt{d_v}} \right)^2$$

$$R_{u,v} = (\mathbf{e_i} - \mathbf{e_j})^T L^+ (\mathbf{e_i} - \mathbf{e_i})$$

Commute Times

$$\mathrm{CT}(u,v) = H(u,v) + H(v,u)$$

$$R_{u,v} = \frac{\mathrm{CT}(u,v)}{\mathrm{vol}(G)}$$

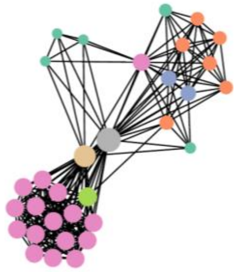
[Qiu and Hancock, 2006]

Commute Times Embedding

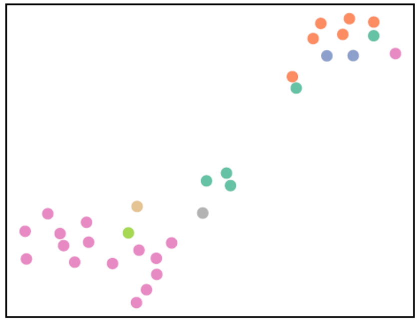
$$\mathbf{Z} = \sqrt{vol(G)}\mathbf{\Lambda}^{-1/2}\mathbf{F}^T \text{ given } \mathbf{L} = \mathbf{F\Lambda F}^T$$

$$\mathbf{Z} = \arg \min_{s.t.\, \mathbf{Z}^T\mathbf{Z}=\mathbb{I}} \frac{Tr[\mathbf{Z}^T\mathbf{L}_G\mathbf{Z}]}{Tr[\mathbf{Z}^T\mathbf{D}_G\mathbf{Z}]}$$

$$\mathrm{CT}(u,v) = \|\mathbf{z}_u - \mathbf{z}_v\|_2^2$$



Spectral CTE



Adrián Arnaiz-Rodríguez, Ahmed Begga, Francisco Escolano, and Nuria Oliver.
DiffWire: Inductive Graph Rewiring via the Lovász Bound. *In The First Learning on Graphs Conference, 2022.*

Spectral Rewiring – Differentiable

DiffWire – CT Layer

- **Learn to rewire in a GNN layer**
  - **Differentiable pipeline**
  - **Data-Driven**

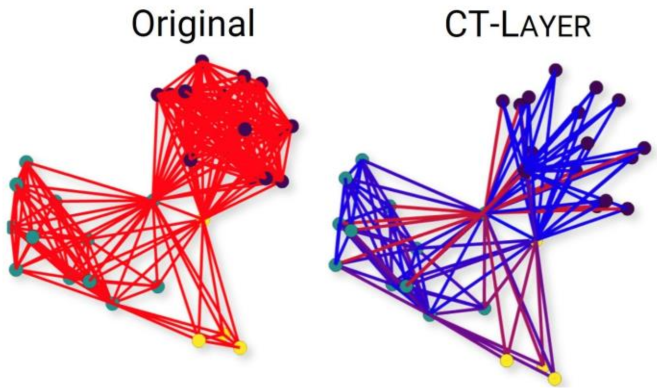
$$\mathbf{Z} = \arg \min_{s.t.\, \mathbf{Z}^T\mathbf{Z}=\mathbb{I}} \frac{Tr[\mathbf{Z}^T \mathbf{L}_G \mathbf{Z}]}{Tr[\mathbf{Z}^T \mathbf{D}_G \mathbf{Z}]}$$

$$\mathrm{CT}(u,v) = \|\mathbf{z}_u - \mathbf{z}_v\|_2^2$$

$$R_{u,v} = \frac{\mathrm{CT}(u,v)}{\mathrm{vol}(G)}$$

- GNN Layer learns the **Commute Time Embedding** between nodes (therefore, it learns the Effective Resistance distance)

- Modifies the message passing (adjacency) using the learned CT → prioritized edges between nodes at large CT



Original

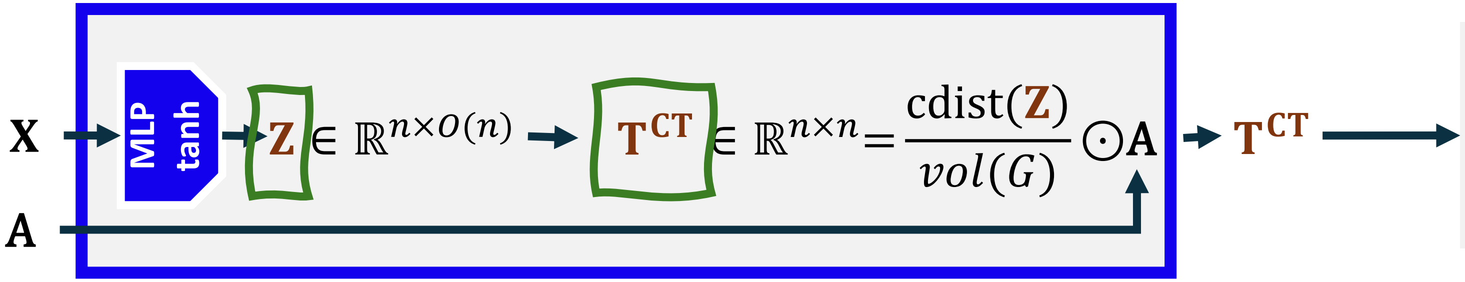
CT-LAYER

Adrián Arnaiz-Rodríguez, Ahmed Begga, Francisco Escolano, and Nuria Oliver.
DiffWire: Inductive Graph Rewiring via the Lovász Bound. *In The First Learning on Graphs Conference, 2022.*

Spectral Rewiring – Differentiable

DiffWire

$$\mathbf{Z} = \sqrt{vol(G)}\boldsymbol{\Lambda}^{-1/2}\mathbf{F}^{T} \rightarrow \mathbf{Z} = \arg\min_{s.t.\,\mathbf{Z}^{T}\mathbf{Z}=\mathbb{I}} \frac{Tr[\mathbf{Z}^{T}\mathbf{L}_{G}\mathbf{Z}]}{Tr[\mathbf{Z}^{T}\mathbf{D}_{G}\mathbf{Z}]} \longrightarrow \boxed{L_{CT} = \frac{Tr[\mathbf{Z}^{\mathbf{T}}\mathbf{L}\mathbf{Z}]}{Tr[\mathbf{Z}^{\mathbf{T}}\mathbf{D}\mathbf{Z}]} + \left\| \frac{\mathbf{Z}^{\mathbf{T}}\mathbf{Z}}{\|\mathbf{Z}^{\mathbf{T}}\mathbf{Z}\|_{F}} - \mathbf{I}_{N} \right\|_{F}}$$


X
A
MLP
tanh
$\mathbf{Z} \in \mathbb{R}^{n \times O(n)}$
$\mathbf{T}^{\mathbf{CT}} \in \mathbb{R}^{n \times n} = \frac{\mathrm{cdist}(\mathbf{Z})}{vol(G)} \odot \mathbf{A}$
$\mathbf{T}^{\mathbf{CT}}$

Use Effective Resistances matrix (commute times) to modify the input adjacency matrix for new layers

$$L_{CT} = \frac{Tr[\mathbf{Z}^{\mathbf{T}}\mathbf{L}\mathbf{Z}]}{Tr[\mathbf{Z}^{\mathbf{T}}\mathbf{D}\mathbf{Z}]} + \left\| \frac{\mathbf{Z}^{\mathbf{T}}\mathbf{Z}}{\|\mathbf{Z}^{\mathbf{T}}\mathbf{Z}\|_{F}} - \mathbf{I}_{N} \right\|_{F}$$

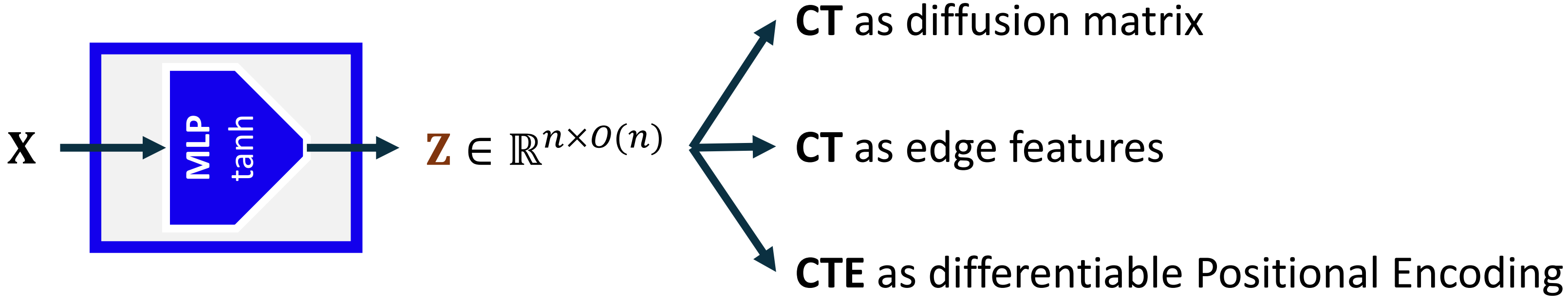
CT-layer can be added as the first layer or as the # desired layer

Adrián Arnaiz-Rodríguez, Ahmed Begga, Francisco Escolano, and Nuria Oliver.
DiffWire: Inductive Graph Rewiring via the Lovász Bound. *In The First Learning on Graphs Conference, 2022.*

Spectral Rewiring – Differentiable

DiffWire

$$L_{CT} = \frac{Tr[\mathbf{Z}^{\mathrm{T}}\mathbf{L}\mathbf{Z}]}{Tr[\mathbf{Z}^{\mathrm{T}}\mathbf{D}\mathbf{Z}]} + \left\| \frac{\mathbf{Z}^{\mathrm{T}}\mathbf{Z}}{\|\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\|_F} - \mathbf{I}_N \right\|_F$$



$\mathbf{X}$

$\mathbf{Z} \in \mathbb{R}^{n \times O(n)}$

CT as diffusion matrix

CT as edge features

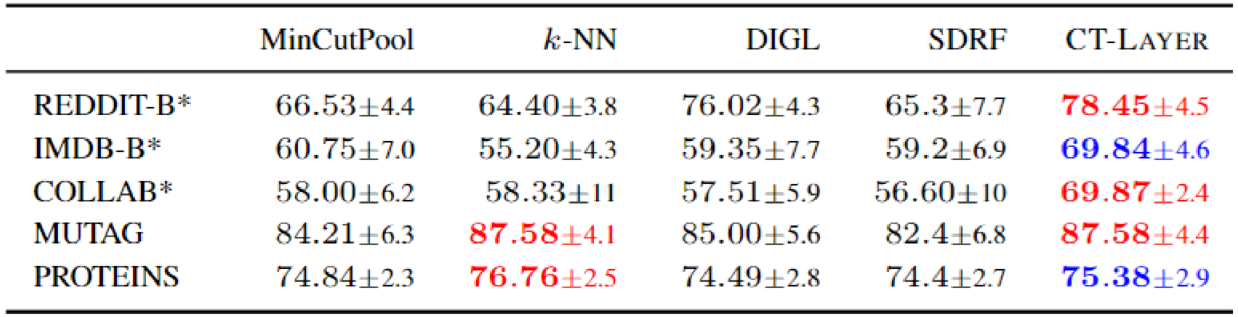
CTE as differentiable Positional Encoding

Adrián Arnaiz-Rodríguez, Ahmed Begga, Francisco Escolano, and Nuria Oliver.
DiffWire: Inductive Graph Rewiring via the Lovász Bound. *In The First Learning on Graphs Conference, 2022.*

Spectral Rewiring – Differentiable

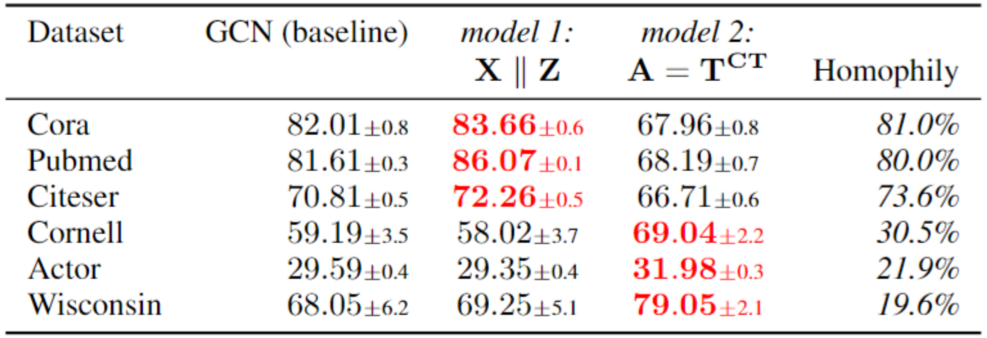
DiffWire

CT learned by **CT-Layer** as diffusion matrix



	MinCutPool	k-NN	DIGL	SDRF	CT-LAYER
REDDIT-B*	66.53±4.4	64.40±3.8	76.02±4.3	65.3±7.7	**78.45**±4.5
IMDB-B*	60.75±7.0	55.20±4.3	59.35±7.7	59.2±6.9	**69.84**±4.6
COLLAB*	58.00±6.2	58.33±11	57.51±5.9	56.60±10	**69.87**±2.4
MUTAG	84.21±6.3	**87.58**±4.1	85.00±5.6	82.4±6.8	**87.58**±4.4
PROTEINS	74.84±2.3	**76.76**±2.5	74.49±2.8	74.4±2.7	**75.38**±2.9

CTE learned by **CT-Layer** as differentiable Positional Encoding



Dataset	GCN (baseline)	*model 1:*	*model 2:*	
		$\mathbf{X} \parallel \mathbf{Z}$	$\mathbf{A} = \mathbf{T}^{\mathbf{CT}}$	Homophily
Cora	82.01±0.8	**83.66**±0.6	67.96±0.8	*81.0%*
Pubmed	81.61±0.3	**86.07**±0.1	68.19±0.7	*80.0%*
Citeser	70.81±0.5	**72.26**±0.5	66.71±0.6	*73.6%*
Cornell	59.19±3.5	58.02±3.7	**69.04**±2.2	*30.5%*
Actor	29.59±0.4	29.35±0.4	**31.98**±0.3	*21.9%*
Wisconsin	68.05±6.2	69.25±5.1	**79.05**±2.1	*19.6%*

PE for homophily
Diffusion for heterophily

Adrián Arnaiz-Rodríguez, Ahmed Begga, Francisco Escolano, and Nuria Oliver.
DiffWire: Inductive Graph Rewiring via the Lovász Bound. *In The First Learning on Graphs Conference, 2022.*

Curvature and Effective Resistance

Connection of curvature and Effective Resistance [Devriendt and Lambiotte, 2022]

- Direct connection for node and edge curvature [Devriendt and Lambiotte, 2022]

$$p_u = 1 - \frac{1}{2} \sum_{v \in \mathcal{N}(u)} R_{uv}$$

Node Curvature

$$\kappa_{uv} = \frac{2(p_u + p_v)}{R_{uv}}$$

Edge Curvature

- Direct connection with node bottleneckedness [Arnaiz-Rodriguez et al 2024]

$$B_R(u) = \sum_{v \in \mathcal{N}(u)} R_{uv}$$

Node bottleneckedness

$$B_R(u) = -2(p_u - 1)$$

Arnaiz-Rodriguez, A., Curto, G., & Oliver, N. (2024).
Structural Group Unfairness: Measurement and Mitigation by means of the Effective Resistance. *In TrustLOG Workshop at WWW 2024.*

Spectral Rewiring

$$\hat{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$$

FOSR [Karhadkar et al 2022]

- Estimate the change of $\boldsymbol{\lambda_2}$ after edge (u, v) addition

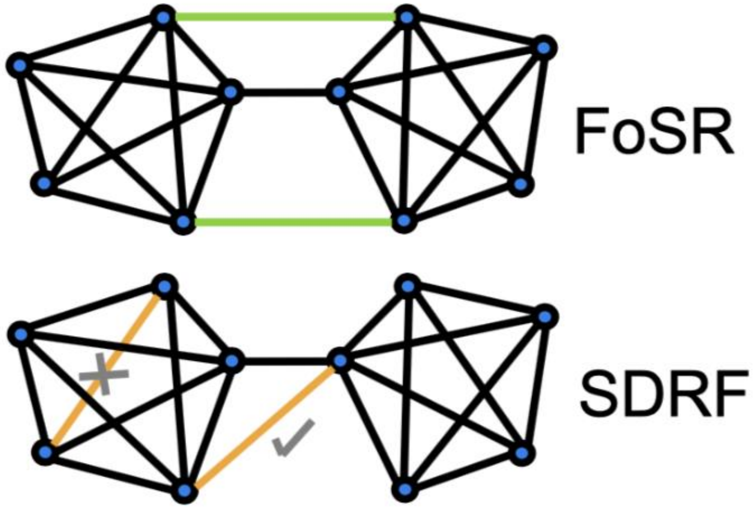
$$\boxed{\frac{2\phi_2(u)\phi_2(v)}{\sqrt{1+d_u}\sqrt{1+d_v}}} + 2\lambda_2\phi_2(u)^2\left(\frac{\sqrt{d_u}}{\sqrt{1+d_u}} - 1\right) + 2\lambda_2\phi_2(v)^2\left(\frac{\sqrt{d_v}}{\sqrt{1+d_v}} - 1\right)$$

- **Goal: minimize the dominant term**

1. Approximate λ_2 via power iteration

$$\phi_2^{t+1} \approx \hat{A}\phi_2^t - \frac{\langle \phi_2^t, \sqrt{d} \rangle}{2m}\sqrt{d}$$

2. Choose edge that minimizes the dominant term



$$R_{u,v} = (\mathbf{e_i} - \mathbf{e_j})^T L^+ (\mathbf{e_i} - \mathbf{e_i})$$

Spectral Rewiring

GTR [Black et al 2023]

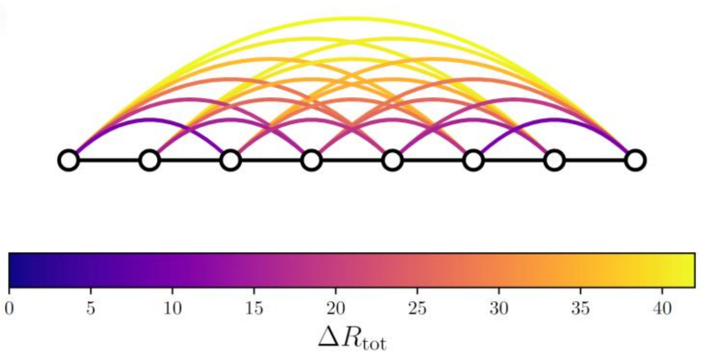
- How much adding a specific decreases R_{tot}?
- Biharmonic Distance [Lipman et al., 2010]

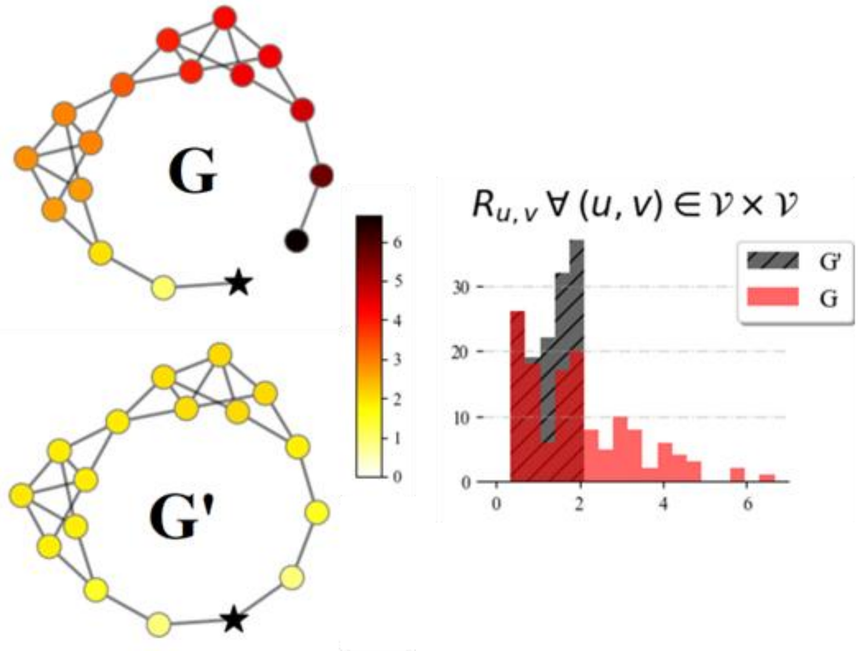
$$B_{u,v} = \sqrt{(\mathbf{e_i} - \mathbf{e_j})^T (L^+)^2 (\mathbf{e_i} - \mathbf{e_i})}$$

- **Proportional to the partial derivative of the total resistance with respect to the weight of the edge** [Gosh et al., 2008]

$$\frac{\partial R_{\mathrm{tot}}}{\partial w_{uv}} = -nB_{uv}^2$$

$$R_{\mathrm{tot}}(G) - R_{\mathrm{tot}}(G \cup (u,v)) = n \cdot \frac{B_{uv}^2}{1 + R_{uv}}$$

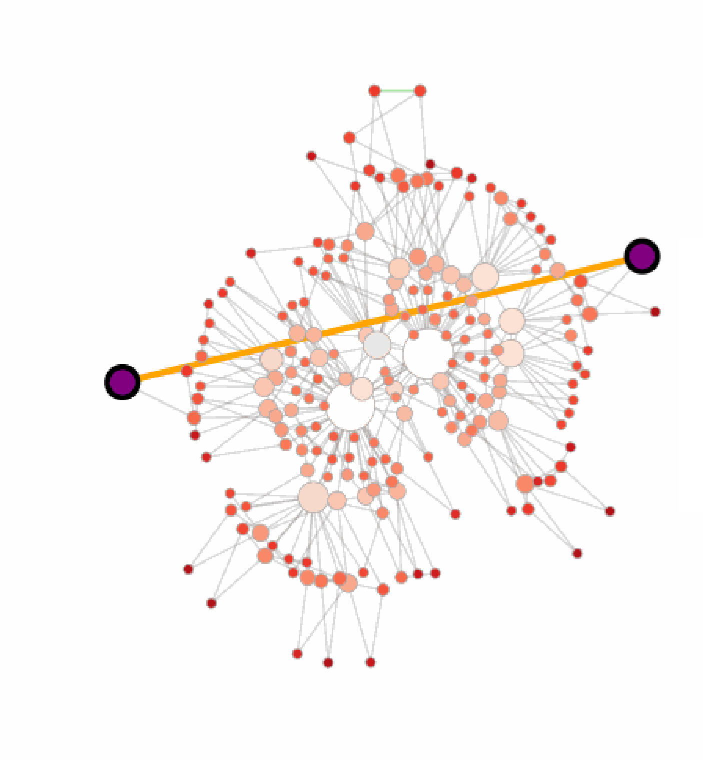


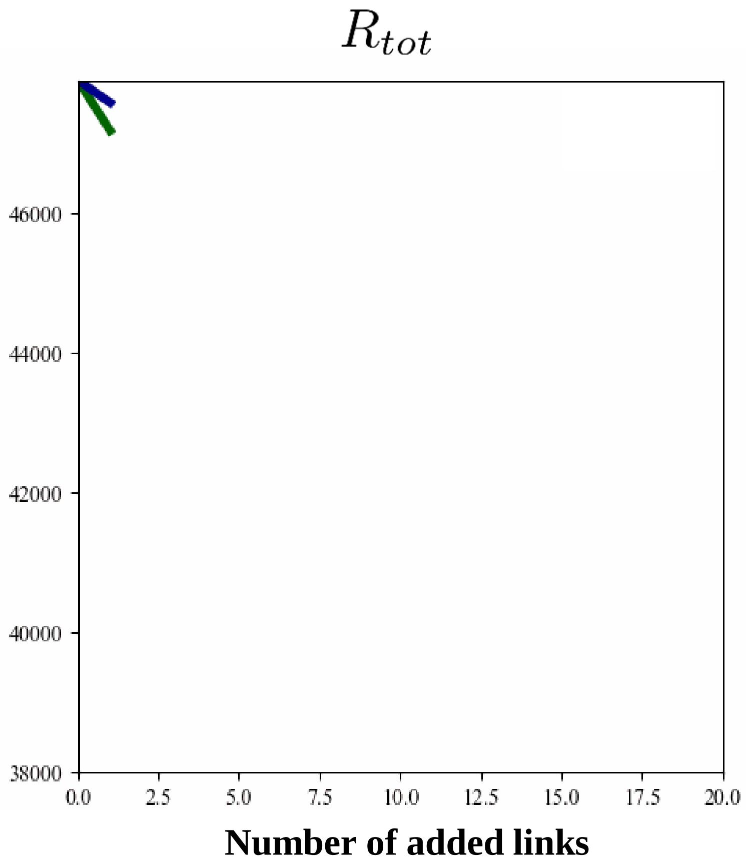


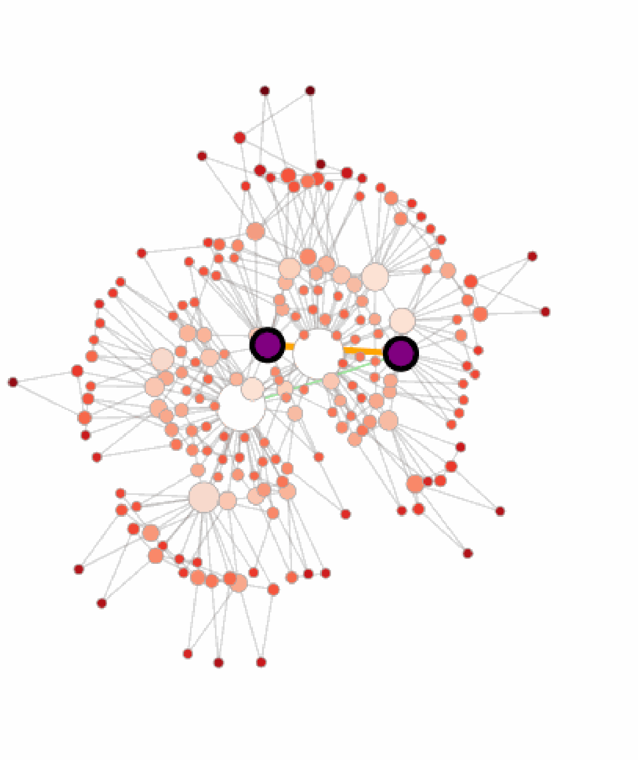
GTR and FOSR

GTR

SDRF



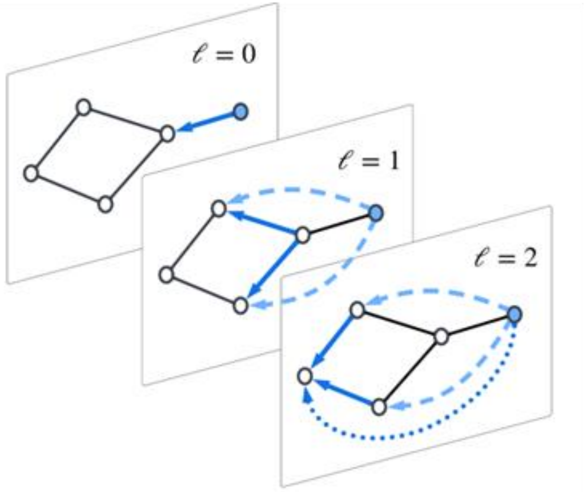




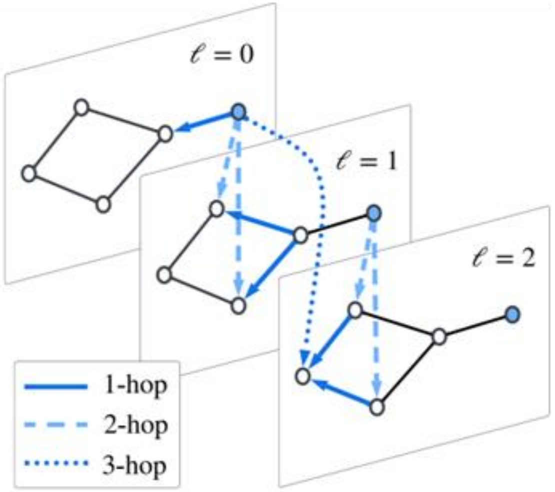
Spatial Rewiring - Dynamic

Drew [Gutteridge et al 2023]

- Closer nodes should interact earlier in the architecture
- Rewire to modulate **not only *if*** nodes interact, **but also *when***.



(b) DRew



(c) νDRew

Spatial Rewiring - Dynamic

Drew [Gutteridge et al 2023]

- Closer nodes should interact earlier in the architecture
- Rewire to modulate **not only *if*** nodes interact, **but also *when***.
- Multi-hop rewiring that evolves during the layers.
  - Nodes interact from a certain depth
    (Hop $\boldsymbol{l}$+$\mathbf{1}$ is only aggregated in layer $\boldsymbol{l}$)
  - Nodes interact with Delay
    (nodes interact with previous states)

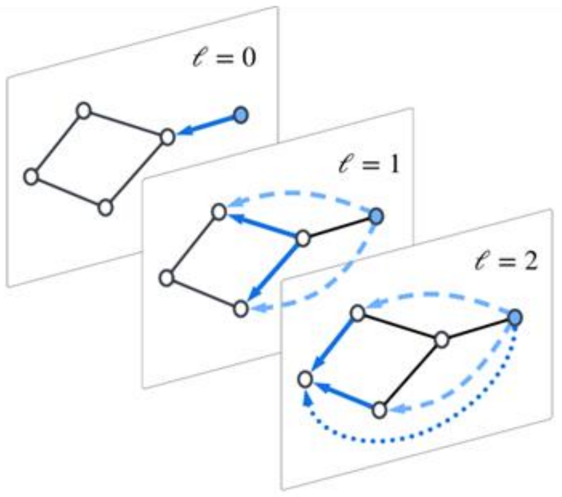
Delay for each hop to interact with the previous states of the nodes

Separate aggregation for each *k*-hop neighborhood

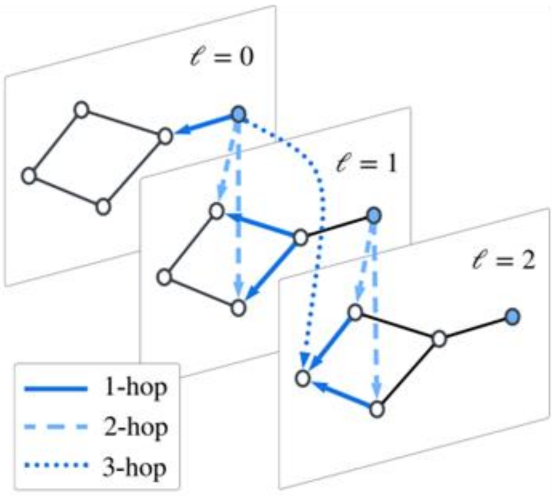
$$a_{i,k}^{(\ell)} = \mathrm{AGG}_k^{(\ell)}\left(\{h_j^{(\ell-\tau_\nu(k))} : j \in \mathcal{N}_k(i)\}\right), 1 \le k \le \ell + 1$$

$$h_i^{(\ell+1)} = \mathrm{UP}_k^{(\ell)}\left(h_i^{(\ell)}, a_{i,1}^{(\ell)}, \ldots, a_{i,\ell+1}^{(\ell)}\right)$$

Hop $l+1$ is only aggregated in layer l



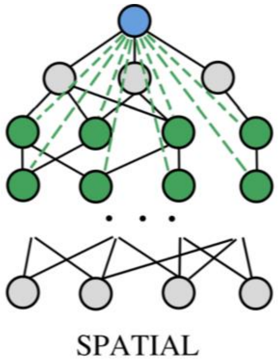
(b) DRew

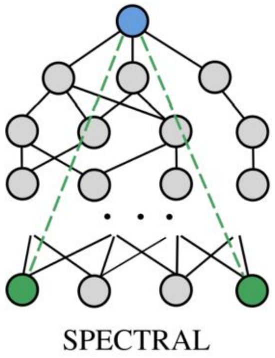


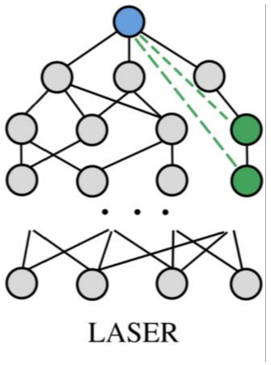
(c) νDRew

Spatio-Spectral Rewiring

LASER [Barbero et al 2024]







Spatial

Add edges within a certain ***k-hop***

❌ Dense nature

✅ Preserve locality

Spectral

Add edges based on a **spectral measure**

✅ Preserve Sparsity

❌ Override locality information

LASER

✅ Preserve Sparsity

✅ Preserve locality

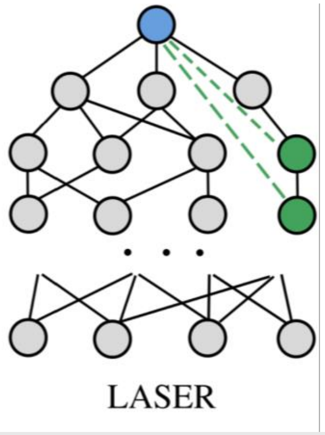
Spatio-Spectral Rewiring

LASER [Barbero et al 2024]

- Use a sequence of rewired graphs (ℓ snapshots)
- Successive "local" modifications

$$\mathsf{G} = \mathsf{G}_0 \xhookrightarrow{\mathcal{R}_1} \mathsf{G}_1 \xhookrightarrow{\mathcal{R}_2} \cdots \xhookrightarrow{\mathcal{R}_L} \mathsf{G}_L$$

$$(v,u) \in \mathsf{E}_\ell \text{ if } \Big(\underbrace{\mu_{\mathsf{G}_0}(v,u) < \epsilon}_{\substack{\text{Connectivity threshold}\\ \text{Connectivity measure}\\ \mu : V \times V \to R}} \text{ and } \underbrace{\nu_{\mathsf{G}_0}(v,u) \in \mathcal{I}_\ell}_{\substack{\text{Locality restriction}\\ \text{Locality measure}\\ \nu : V \times V \to R}}\Big) \text{ or } (v,u) \in \mathsf{E}_{\ell-1}.$$



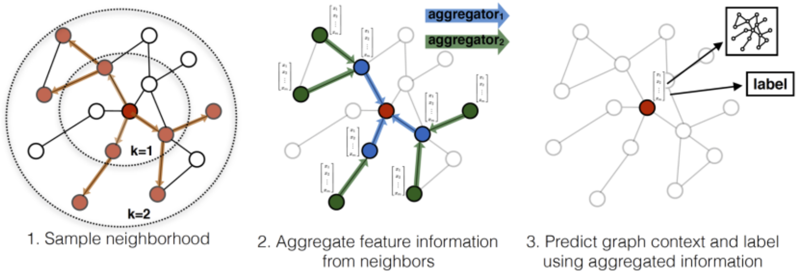
☑ Preserves Sparsity

☑ Preserves locality

Other Rewiring Flavors

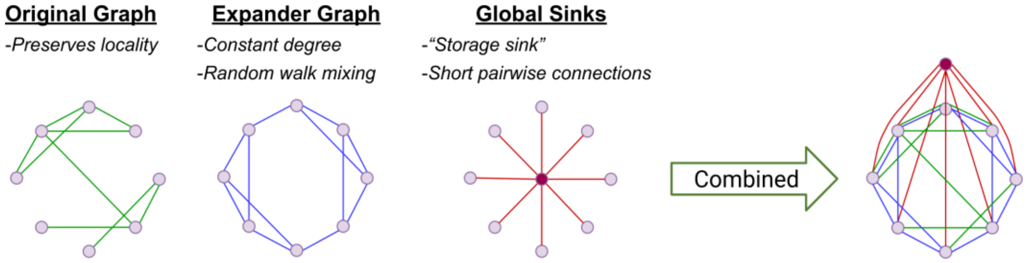
Beyond *pure* Spatial or/and Spectral

- **Sampling**: GraphSAGE [Hamilton et al., '17], GAT [Veličković et al '17]



1. Sample neighborhood
2. Aggregate feature information from neighbors
3. Predict graph context and label using aggregated information

- **Graph transformers with PE**: Graphormer [Ying et al., '21], SAN [Kreuzer et al., '21], GraphGPS [Rampášek et al., '22], Exphormer [Shirzad, Velingker, Venkatachalam et al., '23]



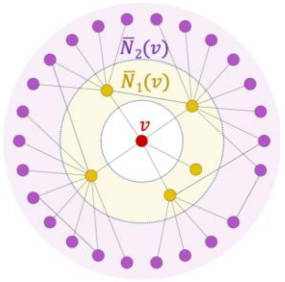
Original Graph
-Preserves locality

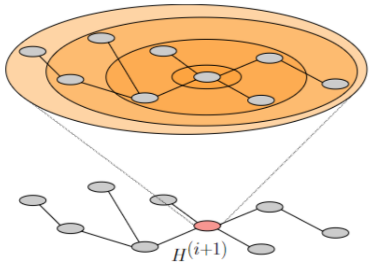
Expander Graph
-Constant degree
-Random walk mixing

Global Sinks
-"Storage sink"
-Short pairwise connections

Combined

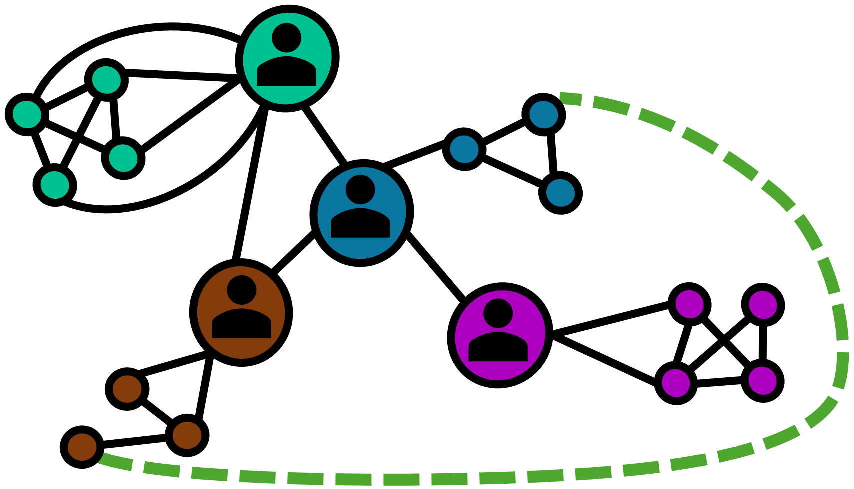
- **High-Order and Hierarchichal GNNS:** Mix-Hop [Abu-El-Haija et al '19], H2GNN [Zhue et al '20], DHGR [Bi et al '22], …





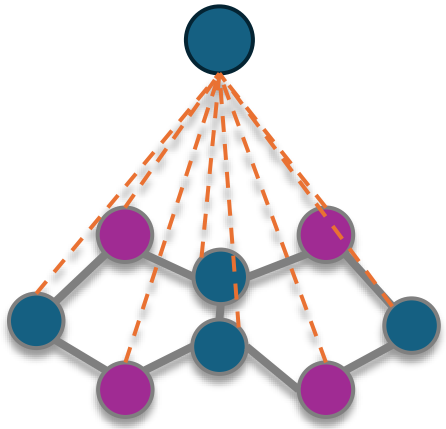
Open problems for Graph Rewiring

- **Blind to the downstream task**
- Focus on addressing OSQ while potentially introducing OSM
- Fail to answer **how much rewiring** is necessary to do
- Most of them pre-processing approaches → **task-agnostic and non-learnable**



Virtual Nodes

$$A_{\text{vn}} = \begin{bmatrix} A & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{bmatrix}$$



- Add new nodes that serve as a global attention shortcuts [Scarselli et al 08; Pham et al 17]
- How many nodes do we add?
  - One global node
  - Several virtual nodes
- How do we connect existing nodes to virtual ones?
  - All-to-one
  - Many-to-many
- Application to transformers → memory sinks [Cai et al 23; Shirzad, Velingker, Venkatachalam et al., 23]

VN embedding

$$\boxed{h_{\text{vn}}^{(\ell+1)}} = \sigma\left(\Omega_{\text{vn}}^{(\ell)} h_{\text{vn}}^{(\ell)} + \boxed{\frac{1}{\tilde{n}} \sum_{j=1}^{n} \phi_{\text{vn}}^{(\ell)}\left(h_{\text{vn}}^{(\ell)}, h_j^{(\ell)}\right)\Big)}\right.,$$

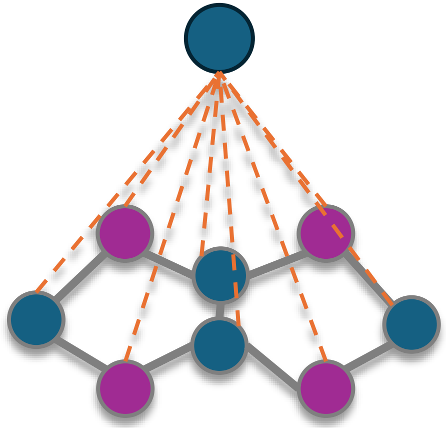
Mean node aggregation

$$h_i^{(\ell+1)} = \sigma\left(\Omega^{(\ell)} h_i^{(\ell)} + \sum_{j=1}^{n} \mathsf{A}_{ij} \psi^{(\ell)}\left(h_i^{(\ell)}, h_j^{(\ell)}\right) + \boxed{\psi_{\text{vn}}^{(\ell)}\left(h_i^{(\ell)}, h_{\text{vn}}^{(\ell)}\right)}\right)$$

Update for node embedding

Virtual Nodes

$$A_{\text{vn}} = \begin{bmatrix} A & \mathbf{1} \\ \mathbf{1}^\top & 0 \end{bmatrix}$$



- Add new nodes that serve as a global attention shortcuts [Scarselli et al 08; Pham et al 17]
- How many nodes do we add?
  - One global node
  - Several virtual nodes
- How do we connect existing nodes to virtual ones?
  - All-to-one
  - Many-to-many
- Application to transformers → memory sinks [Cai et al 23; Shirzad, Velingker, Venkatachalam et al., 23]

Why virtual nodes are beneficial?

[Southern et al 2024]

Average change in commute time

$$\frac{1}{n^2}\sum_{u,v\in V}^{n} \mathrm{CT}_{\text{vn}}(u,v) - \mathrm{CT}(u,v)$$

$$= \frac{4|E|}{n}\sum_{l}^{n-1} \frac{1}{\lambda_l(\lambda_l+1)}\left(\frac{n}{|E|}\lambda_l - 1\right)$$

- For many real-world graphs, **the change is negative** (exception e.g. in complete graphs)
- On these, the **# layers** required by **MPNN + VN** to **learn graph functions with strong mixing** is **smaller than that of MPNN**

Sensitivity

- For MPNN + VN **with mean node aggregation** for the embedding of the virtual node **VN**

$$\left\|\frac{\partial \mathbf{h}_u^{(l+1)}}{\partial \mathbf{h}_v^{(l)}}\right\| \downarrow$$

- **Independent on of *v* whenever *u* and *v* are separated by more than 2 hops**
- Any message is first received by node at layer ℓ + 1 through the VN

Advanced Architectures

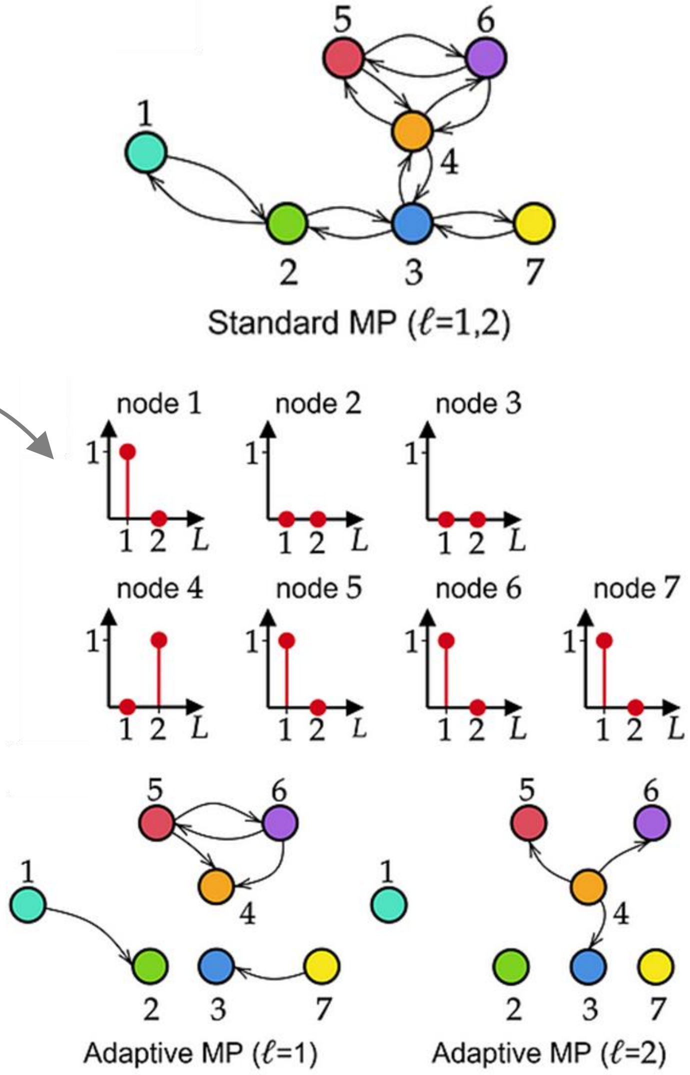
Adaptative Architectures

What messages to share and how many times?

Adaptative Message Passing [Errica et al 24]
learn the optimal depth and filter messages dynamically

- **Differentiable message filtering** mechanism decides what to propagate at each layer
  - Decides what to propagate at each layer

$$h_v^\ell = \phi^\ell\left(h_v^{\ell-1}, \Psi\left(\{F_i(u,\ell-1) \odot \psi^\ell(h_u^{\ell-1}, a_{uv}^\ell) | u \in \mathcal{N}_v\}\right)\right)$$



Advanced Architectures

Adaptative Architectures

What messages to share and how many times?

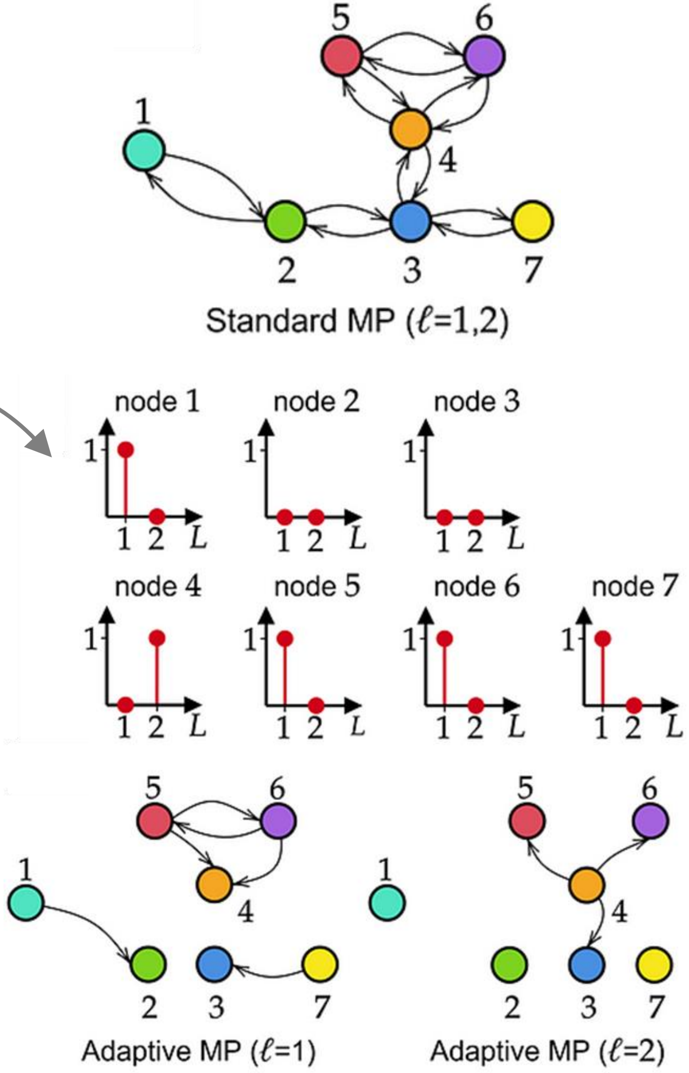
Adaptative Message Passing [Errica et al 24]
learn the optimal depth and filter messages dynamically

- **Differentiable message filtering** mechanism decides what to propagate at each layer
  - Decides what to propagate at each layer

$$h_v^{\ell} = \phi^{\ell}\left(h_v^{\ell-1}, \Psi\left(\{F_i(u,\ell-1) \odot \psi^{\ell}(h_u^{\ell-1}, a_{uv}^{\ell}) | u \in \mathcal{N}_v\}\right)\right)$$

- **Dynamic adjustment of network depth**
  - Learn depth using variational Inference
  - how many message-passing layers are required for a specific task?

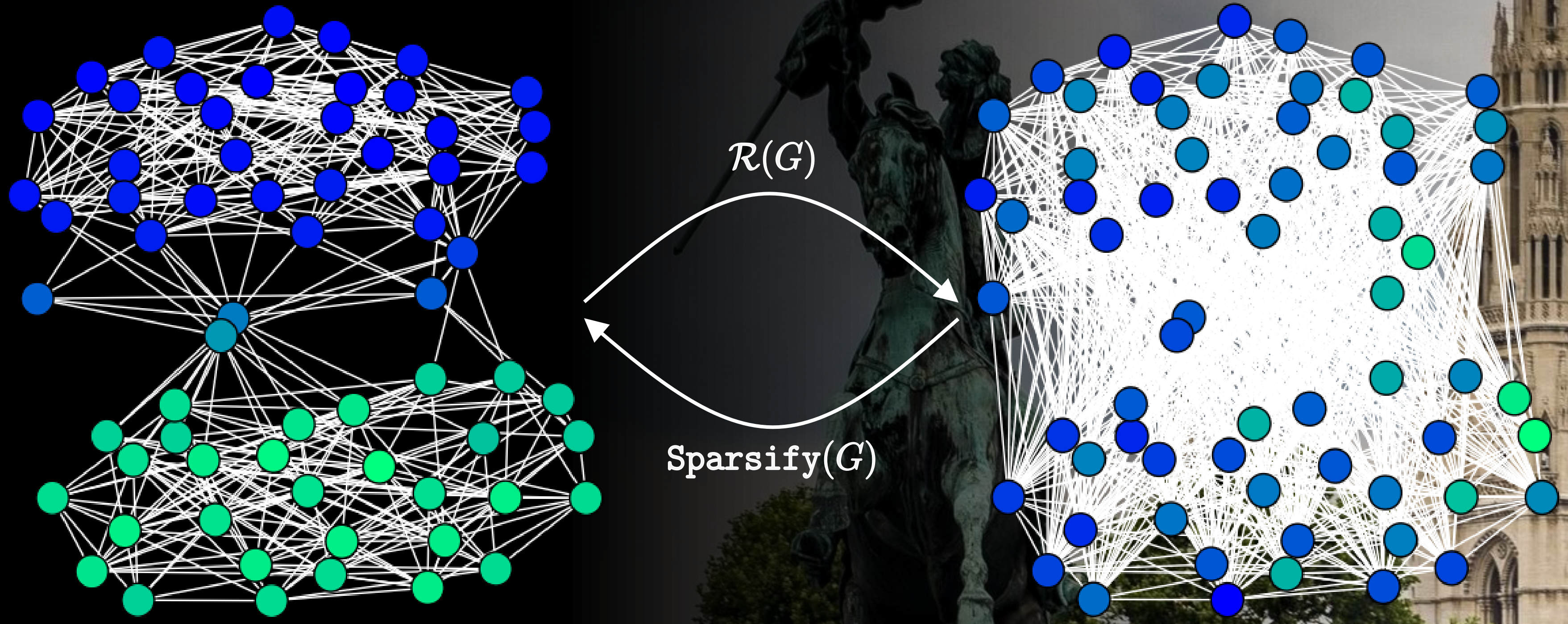
$$\ln p(g_i, Y_i)$$
$$\geq \mathbb{E}_{q(\theta,L,F_i|g_i,Y_i)}\left[\ln p(Y_i, L, F_i, \theta|g_i) - \ln q(L, F_i, \theta|g_i)\right]$$



OSQ Take-away

- Over-squashing is caused by bottlenecks in the graph
- Measured by spectral quantities and by the Jacobian obstruction
- Obstruction is bounded by the topology and it’s independent of the GNN

- Solutions
  - Rewiring (spatial vs spectral, static vs dynamic, pre-processing vs learnable)
  - Virtual nodes
  - Different architectures for diffusion in graphs
    - Multi-hops, Adaptative, Physics-informed, Spectral, Transformers…
  - *Note*: Some approaches are combined (e.g. global nodes+ rewiring, multi-hop + rewiring…)



Trade-off between OSM and OSQ

Trade-off between OSM and OSQ

A comparison of the bounds

$$\frac{\lambda_2}{2} \leq h_G \leq \sqrt{2\lambda_2}$$

The lower λ_2 → the higher the bottleneck → the more OSQ

$$\left\| f^T P^k - \pi \right\|_2 \leq e^{-k\boldsymbol{\lambda}'} \log \left(\frac{\max_v \sqrt{d_x}}{\min_u \sqrt{d_u}} \right)$$

$$\mathcal{E}(PX) \leq (1 - \lambda_2)^2 \mathcal{E}(X)$$

The higher λ_2 → the faster convergence→ the more OSM

Trade-off between OSM and OSQ

A comparison of the bounds

$$\frac{\lambda_2}{2} \leq h_G \leq \sqrt{2\lambda_2}$$

The lower λ_2 → the higher the bottleneck → the more OSQ

$$\sum_{u,v \in V \times V} \left\| \frac{\partial h_v^{(r)}}{\partial h_u^{(0)}} \right\| \leq c(b - R_{tot})$$

The higher R_{tot}→ the lower the bound of sum pairwise sensitivities → the more OSQ

$$\tilde{\mathrm{O}}^m(u, v) = \sum_{k=0}^{m} \left\| \tilde{\mathbf{J}}_k^{(m)}(v, u) \right\| \leq c\ R_{u,v}$$

The higher R_{uv}→ the larger the pairwise obstruction → the more OSQ

$$\left\| f^T P^k - \pi \right\|_2 \leq e^{-k\boldsymbol{\lambda}'} \log\left(\frac{\max_v \sqrt{d_x}}{\min_u \sqrt{d_u}} \right)$$

$$\mathcal{E}(PX) \leq (1 - \lambda_2)^2 \mathcal{E}(X)$$

The higher λ_2 → the faster convergence→ the more OSM

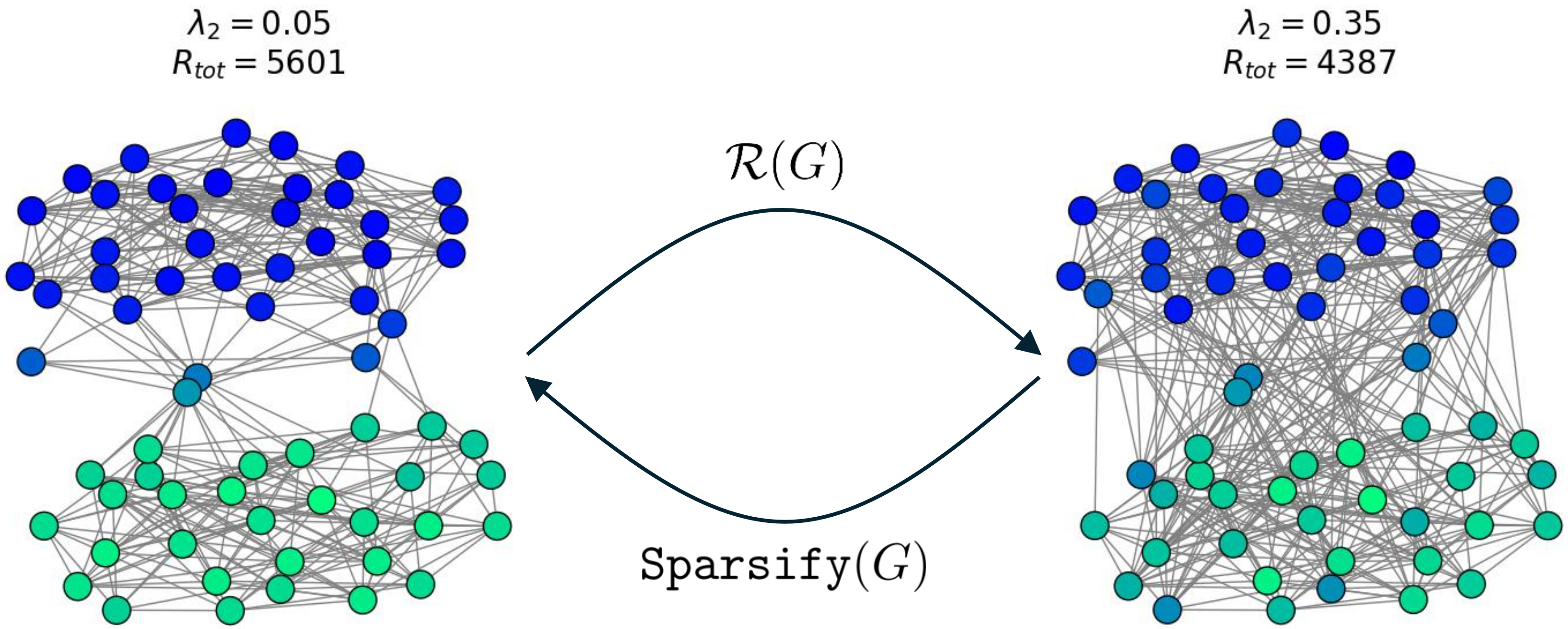
$$\lim\nolimits_{t \to \infty} x(t) = \lim\nolimits_{t \to \infty} e^{-tL} x(0) = \frac{\mathbf{1}\mathbf{1}^T x(0)}{n}$$

$$T_k = \frac{1}{\lambda_k}$$

$$R_{tot} = \sum_{u \sim v} R_{u,v} = n \sum_{k=2}^{n} \frac{1}{\lambda_k} = n \sum_{k=2}^{n} T_k$$

The lower R_{tot} → the faster convergence→ the more OSM

Trade-off between OSM and OSQ



Good to mitigate over-smoothing

Risk of over-smoothing

Risk of over-squashing

Good to mitigate over-squashing

Trade-off between OSM and OSQ

Layers needed for feature collapse vs Cheeger constant

- The **bottleneck** of the graph is **upper bounded** by the inverse of the **number of steps** needed to reach at most ***ϵ*-feature collapse** [Giraldo et al 2023]

$$\left\|f^T P^k - \pi\right\|_2 \leq \epsilon \quad\longrightarrow\quad s \geq \frac{1}{\lambda' \log\left(\frac{\max_v \sqrt{d_x}}{\epsilon \min_u \sqrt{d_u}}\right)}$$

***ϵ*-feature collapse**

Difference of signal ***f*** and ***π*** at most ***ϵ***

s is the **number of steps** to reach ***ϵ*-feature collapse**

Trade-off between OSM and OSQ

Layers needed for feature collapse vs Cheeger constant

- The **bottleneck** of the graph is **upper bounded** by the inverse of the **number of steps** needed to reach at most **ϵ-feature collapse** [Giraldo et al 2023]

$$\left\| f^T P^k - \pi \right\|_2 \leq \epsilon \longrightarrow s \geq \frac{1}{\lambda' \log\left(\frac{\max_v \sqrt{d_x}}{\epsilon \min_u \sqrt{d_u}}\right)}$$

ϵ-feature collapse
Difference of signal $\boldsymbol{f}$ and $\boldsymbol{\pi}$ at most $\boldsymbol{\epsilon}$

s is the **number of steps** to reach **ϵ-feature collapse**

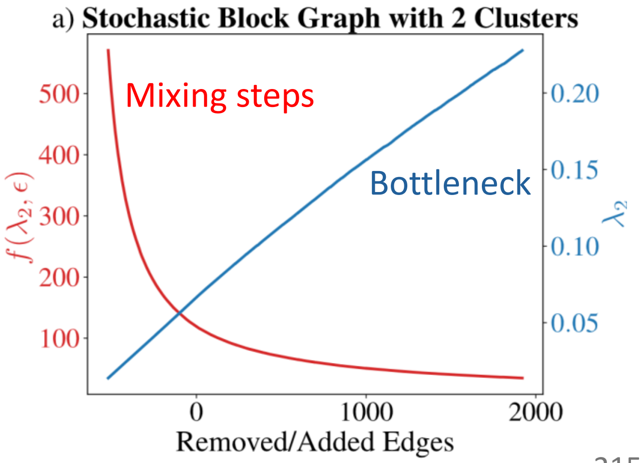
$$2h_G \leq \frac{1}{s} \log\left(\frac{\max_v \sqrt{d_x}}{\epsilon \min_u \sqrt{d_u}}\right)$$

$$s \to \infty \iff h_G \to 0$$

Avoid OSM making the signal to never converge
→ Cheeger constant is 0

$$h_G \to \infty \iff s \to 0$$

Avoid OSQ making the bottleneck large
→ small steps for ϵ-feature collapse



a) **Stochastic Block Graph with 2 Clusters**

Trade-off between OSM and OSQ

Solutions and Analysis

- **Relational GNNs**
  - ***FOSR*** [Karhadkar et al 22] and ***LASER*** [Barbero et al 24].

- **Curvature Methods**
  - ***SJLR*** [Giraldo et al 23] **adds and remove edges** and analysis of trade-off in the spectral domain.
  - ***BORF*** [Nguyen et al 23] adds and remove edges with low and high curvatures (solving OSQ and OSM) resembling an expander. They connect of curvature with the DE and with Jacobian.
  - ***AFR-3*** [Fesser and Weber, 24] propose a heuristic to choosing how many edges to add in curvature methods. They connect OSM and OSQ with augmented Forman curvature.
  - ***CurvDrop*** [Liu et al 23] sample edges based on curvature to mitigate OSM and OSQ.

- **Spectral**
  - ***ProxyDelete*** [Jamadandi et al 24] analyze that **deleting edges** can address OSM and OSQ simultaneously.
  - ***UniFilter*** [Huang et al 24] propose a general graph filter based on a universal polynomial basis tailored for different heterophily degrees

- **More**
  - ***Adaptative Message Passing*** [Errica et al 24] propose a probabilistic framework to learn **how many messages to exchange between nodes (GNN depth) and which messages to filter out** to prevent feature convergence and increase feature sensitivity.
  - [Southern et al 24] compares **virtual nodes** with smoothing techniques and over-squashing measures.

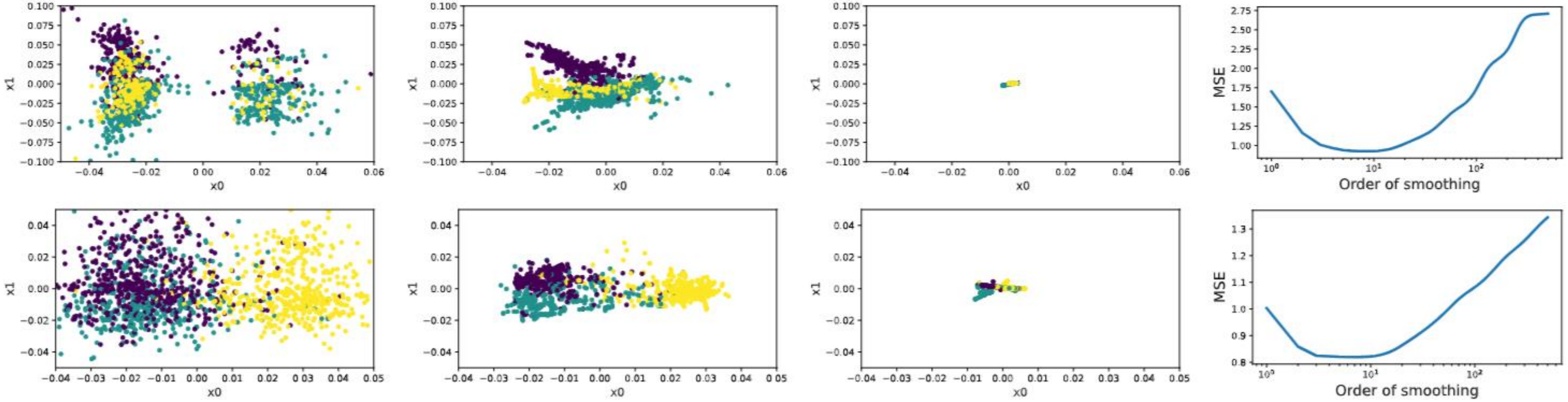
Open Questions

Open Questions on OSM

- It is not always correlated with the accuracy… Is it really a problem of GNNs? Not the only one
  - Role of real world GNNs and training process? Mitigated even with gated GCNs or relational GCNs
- **Is OSM always a problem? *Not too little, not too much*** (focus before $k \to \infty$)
  - Graph CLF → beneficial smoothing is desired if it is aligned with the task
  - **Node CLF → For homophily some OSM is desired!** [Keriven 2022]

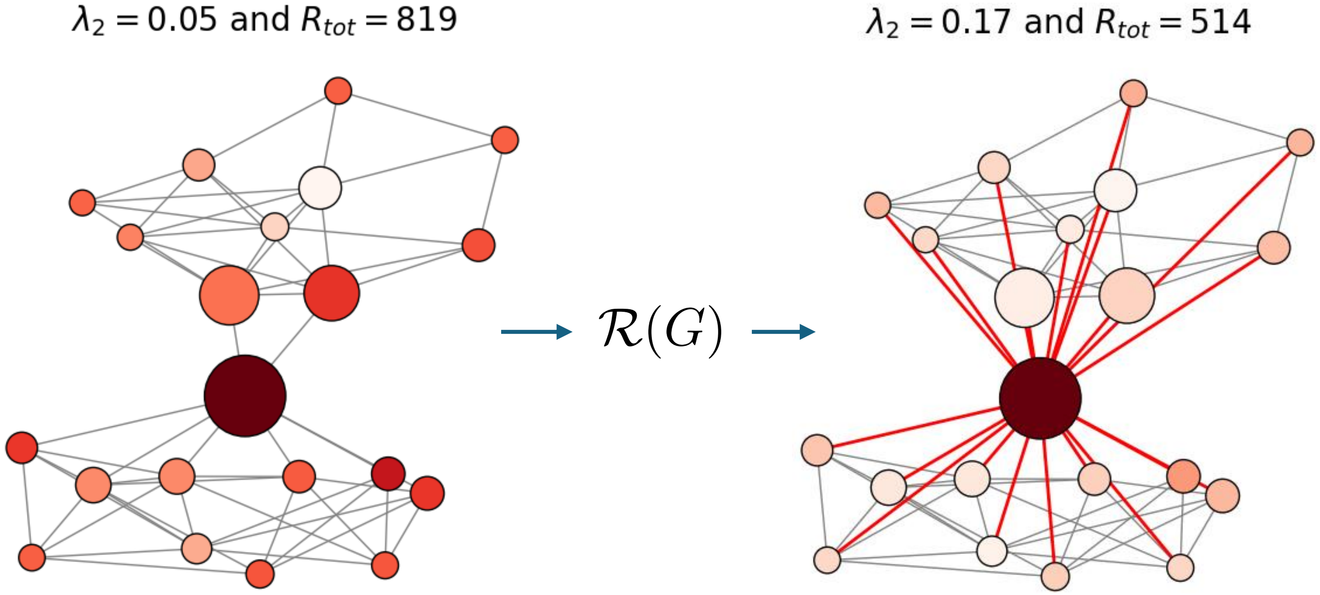
OSM happens faster in some subspaces than in others

Useful if labels are correlated with those subspaces



Open Questions on OSQ

- **Bottleneck vs Sensibility** – How do they measure OSQ? Are we identifying information bottlenecks?



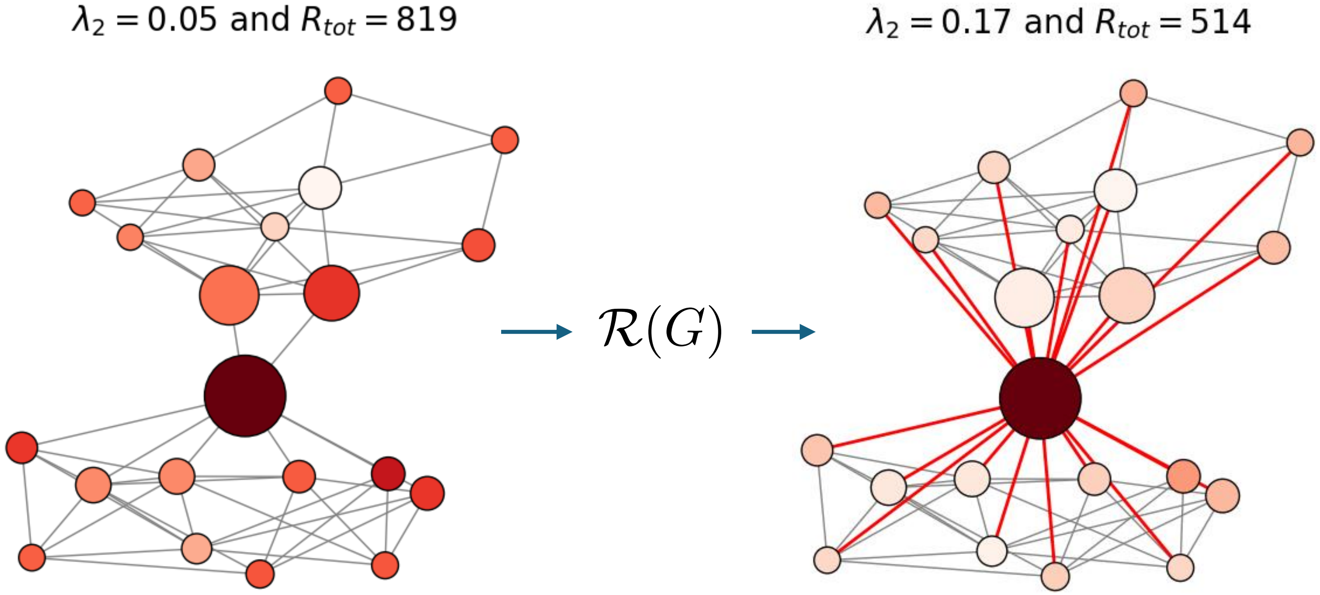
$\lambda_2 = 0.05$ and $R_{tot} = 819$

$\lambda_2 = 0.17$ and $R_{tot} = 514$

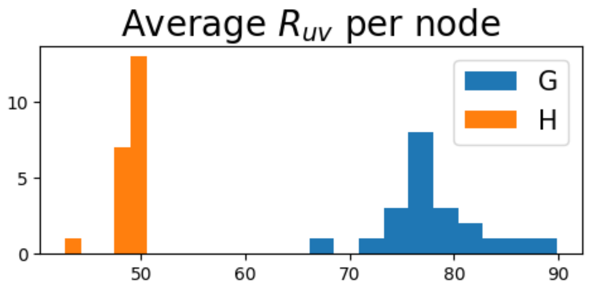
λ_2 increases and R_{tot} decreases, cool! But… **Is OSQ Solved?**

Open Questions on OSQ

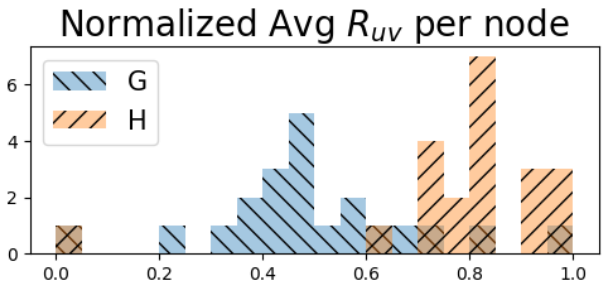
- **Bottleneck vs Sensibility** – How do they measure OSQ? Are we identifying information bottlenecks?


$\lambda_2 = 0.05$ and $R_{tot} = 819$
$\lambda_2 = 0.17$ and $R_{tot} = 514$
$\rightarrow \mathcal{R}(G) \rightarrow$

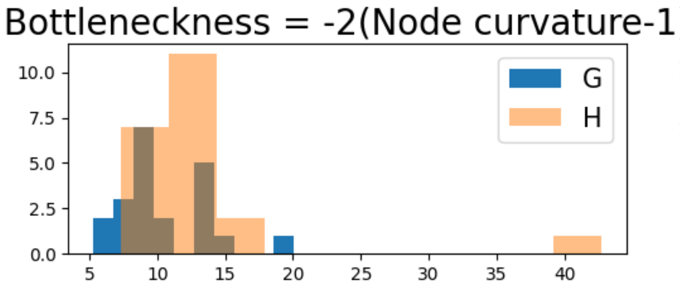
λ_2 increases and R_{tot} decreases, cool! But… **Is OSQ Solved?**



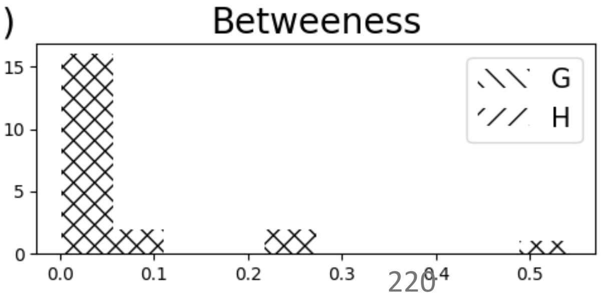
Seems to work very well but…



Becomes extreme



Becomes extreme

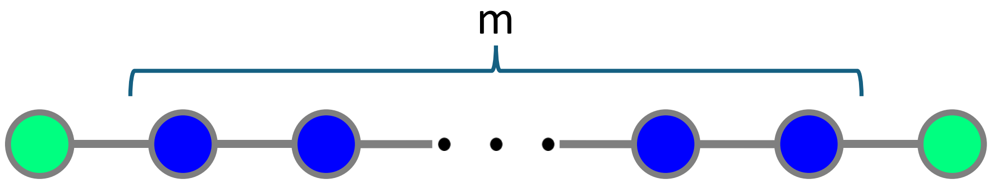


Does not change

Open Questions on OSQ

- Are we properly measuring which graphs suffer from **task-relevant over-squashing**?
  - Bottleneck vs Sensibility measures are independent of the label of the nodes – **Measures blind to task (labels)**
    - Prior work connection of OSQ with the function that MPNN seek to learn [Di Giovanni et al 2024]
    - Homophilic bottlenecking: analyze combined effect of heterophily and over-squashing [Rubin et al 2023]
  - OSQ datasets are currently measured with heterophily metrics – **Heterophily is not the same as long range!**

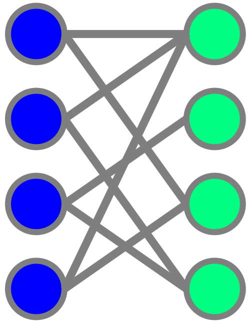
1-hop



$$h_{edge} = \frac{m-1}{n-1} = \frac{n-3}{n-1}$$

$$h_{edge}^{m\to\infty} = 1$$

Homophilic but long-range



$$h_{edge} = 0$$

$$\max(d(u,v) : y_u = y_v) = 2$$

Heterophilic but short-range

- **Most OSQ mitigation strategies are task-agnostic and non-learnable**

Open Questions on OSQ

Beyond Heterophily - How does the structure align with the labels?

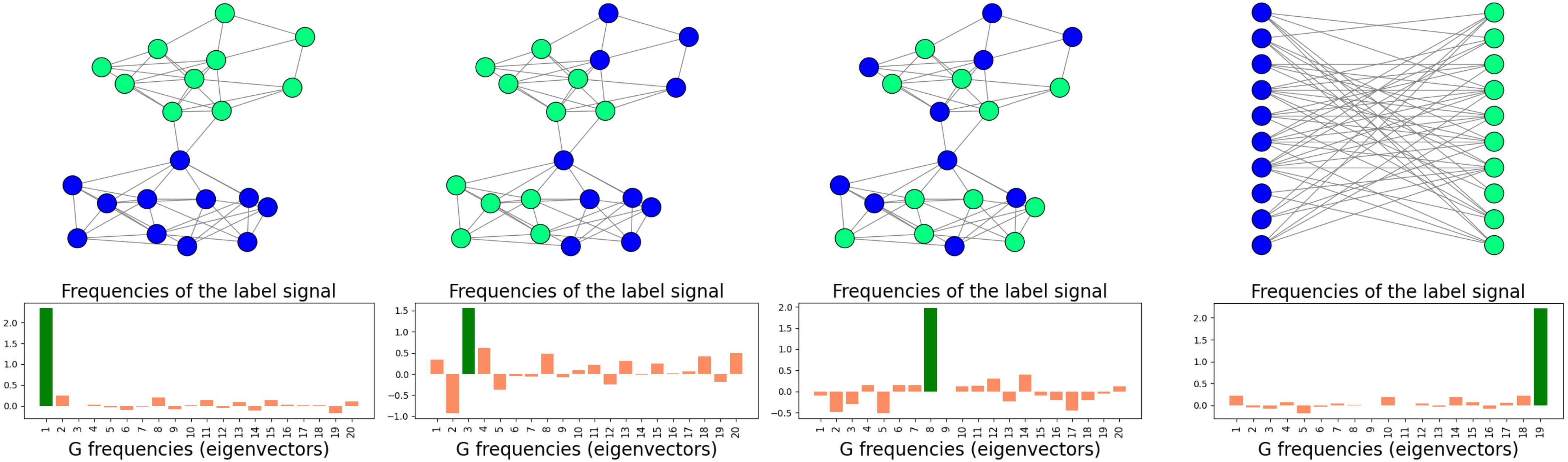
- K-hop homophily metrics. High-order homophily.
- **Spectral metrics**
  - Graph Fourier operator. Project Y signal in the spectral domain. $\mathcal{F}(\boldsymbol{x}) = \boldsymbol{\phi}^\top \boldsymbol{x}$

Homophilic signals have **higher energy in low frequency** components

[Zhu et al 2020; Luan et al 2024]

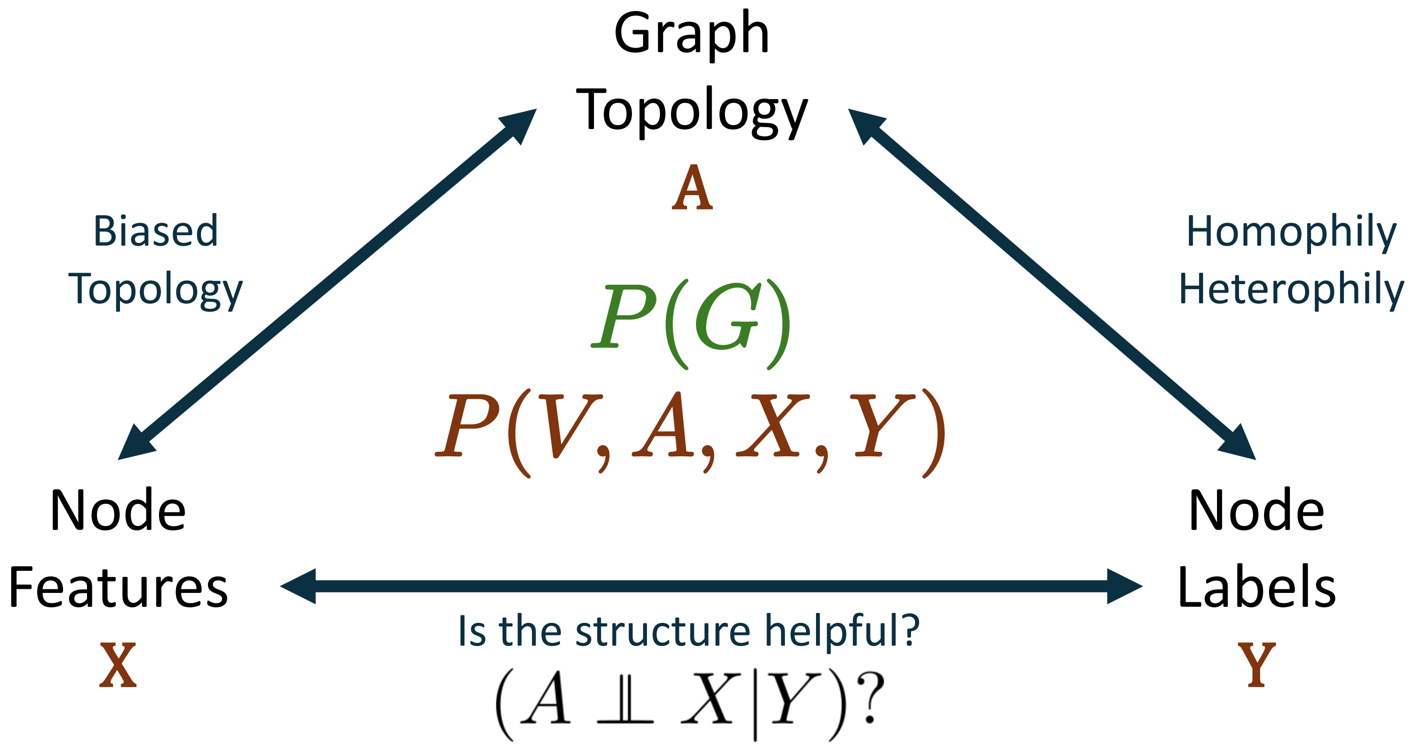
Heterophilic signals have **higher energy in high frequency** components

$$S_i = Y^\top \phi_i, \;\; S_i \in \mathbb{R}^n$$



- More discussion about homophily in [Zhu et al 20; Qian et al 21; Luan et al 22; Ma et al 22; Huang et al 24; Luan et al 24; Zheng et al 24]

Underlying problem: Probability distribution of G



Graph Topology

$\mathbf{A}$

Biased Topology

Homophily Heterophily

$P(G)$

$P(V, A, X, Y)$

Node Features

$\mathbf{X}$

Node Labels

$\mathbf{Y}$

Is the structure helpful?

$(A \perp\!\!\!\perp X | Y)?$

Survey on Deep Graph Generation [Zhu et al 2022]
Survey on Graph Structure Learning [Zhu et al 2021; Luan et al 2024]
Survey on Causality on GNNs [Jiang et al 2023]

Conclusions OSM and OSQ

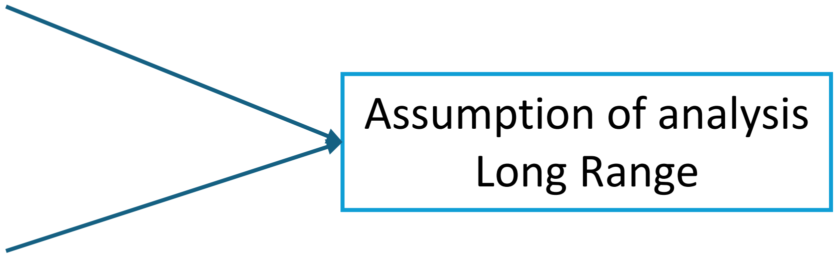
- **OSM → Feature convergence to not expressive**
  - Dirichlet Energy-based measures
  - Due to network depth and graph density
  - Solutions based on feature normalization, graph sparsification, W normalization and time-continuous GNNs

- **OSQ → Exponential compression of nodes' features into fixed-length feature vectors**
  - Measured by
    - Feature compression → Existence of bottlenecks in the graph
    - Feature Sensitivity → Interaction between nodes' features
  - Solutions based on graph rewiring, virtual nodes or adaptative architectures

- **Trade-off →** Both problems are connected

Recap on open questions

- **OSM**
  - Extend OSM analysis to real world GNNs… Is it really a problem on GNNs?
    - Accuracy drops even with high DE’s
  - Is over-smoothing always bad?
  - **Task-oriented over-smoothing**



Assumption of analysis
Long Range

- **OSQ**
  - Identify differences between bottleneck analysis and sensitivity
    - Do both happen at the same time?
  - **Task-oriented Over-squashing**
    - might not be always bad
    - Alignment of structure, features and labels
      - Currently: homophily. But homophily not is long range

- **What is the probability distribution of a graph/s?**

More recent work

- **ICML24**
  - https://github.com/azminewasi/Awesome-Graph-Research-ICML2024
- **ICLR24**
  - https://github.com/azminewasi/Awesome-Graph-Research-ICLR2024

azminewasi
Azmine Toushik Wasi

- **LoG Conference**
  - Sept 4th Abstract Deadline
  - Sept 11th Submission Deadline

References

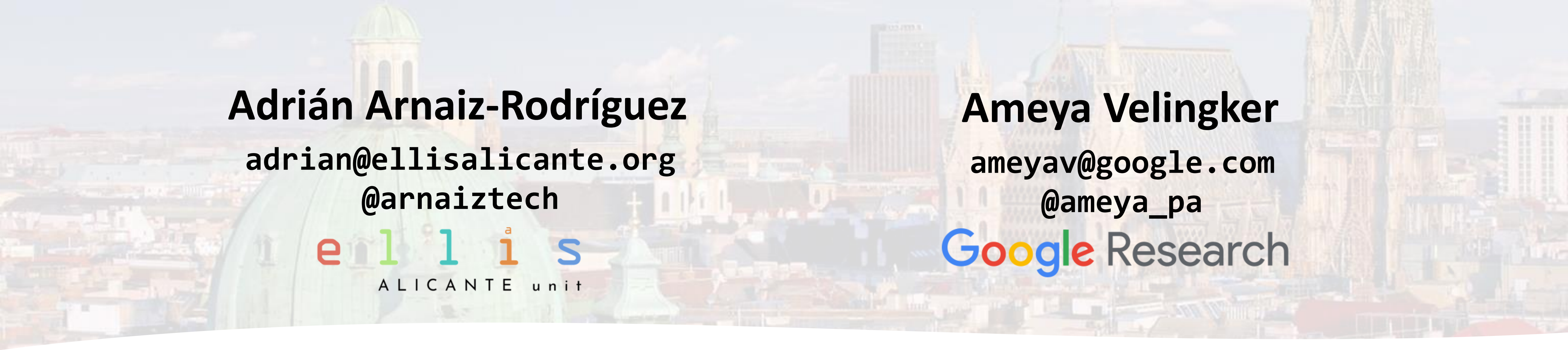
- Abboud, R., Dimitrov, R., & Ceylan, I. I. (2022, December). Shortest path networks for graph property prediction. In Learning on Graphs Conference (pp. 5-1). PMLR.
- Abu-El-Haija, Sami, et al. "Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing." international conference on machine learning. PMLR, 2019.
- Alev, V. L., Anari, N., Lau, L. C., & Gharan, S. O. (2018). Graph clustering using effective resistance http://drops.dagstuhl.de/opus/volltexte/2018/8369.
- Alon, U., & Yahav, E. (2020). On the bottleneck of graph neural networks and its practical implications. In ICLR 2021.
- Arnaiz-Rodríguez, A., Begga, A., Escolano, F., & Oliver, N. (2022). Diffwire: Inductive graph rewiring via the Lovász bound. In the First Learning on graphs (LoG) Conference 2022.
- Arnaiz-Rodriguez, A., Curto, G., & Oliver, N. (2024). Structural Group Unfairness: Measurement and Mitigation by means of the Effective Resistance. In TrustLOG Workshop at WWW 2024.
- Banerjee, P. K., Karhadkar, K., Wang, Y. G., Alon, U., & Montúfar, G. (2022, September). Oversquashing in gnns through the lens of information contraction and graph expansion. In 2022 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton) (pp. 1-8). IEEE.
- Chamberlain, B., Rowbottom, J., Eynard, D., Di Giovanni, F., Dong, X., & Bronstein, M. (2021). Beltrami flow and neural diffusion on graphs. Advances in Neural Information Processing Systems, 34, 1594-1609.
- Bi, W., Du, L., Fu, Q., Wang, Y., Han, S., & Zhang, D. (2022). Make heterophily graphs better fit gnn: A graph rewiring approach. arXiv preprint arXiv:2209.08264.
- Black, M., Wan, Z., Nayyeri, A., & Wang, Y. (2023, July). Understanding oversquashing in gnns through the lens of effective resistance. In International Conference on Machine Learning (pp. 2528-2547). PMLR.
- Bodnar, C., Di Giovanni, F., Chamberlain, B., Lio, P., & Bronstein, M. (2022). Neural sheaf diffusion: A topological perspective on heterophily and oversmoothing in gnns. Advances in Neural Information Processing Systems, 35, 18527-18541.
- Brüel-Gabrielsson, R., Yurochkin, M., & Solomon, J. (2022). Rewiring with positional encodings for graph neural networks. TMLR 2023
- Buchnik, E., & Cohen, E. (2018, June). Bootstrapped graph diffusions: Exposing the power of nonlinearity. In Abstracts of the 2018 ACM International Conference on Measurement and Modeling of Computer Systems (pp. 8-10).
- Cai, C., & Wang, Y. (2020). A note on over-smoothing for graph neural networks. In ICML 2020.
- Cai, C., Hy, T. S., Yu, R., & Wang, Y. (2023, July). On the connection between mpnn and graph transformer. In International Conference on Machine Learning (pp. 3408-3430). PMLR.
- Chamberlain, B., Rowbottom, J., Gorinova, M. I., Bronstein, M., Webb, S., & Rossi, E. (2021, July). Grand: Graph neural diffusion. In International conference on machine learning (pp. 1407-1418). PMLR.
- Chamberlain, B., Rowbottom, J., Eynard, D., Di Giovanni, F., Dong, X., & Bronstein, M. (2021b). Beltrami flow and neural diffusion on graphs. Advances in Neural Information Processing Systems, 34, 1594-1609.
- Chen, M., Wei, Z., Huang, Z., Ding, B., & Li, Y. (2020, November). Simple and deep graph convolutional networks. In International conference on machine learning (pp. 1725-1735). PMLR.
- Chen, T., Zhou, K., Duan, K., Zheng, W., Wang, P., Hu, X., & Wang, Z. (2022). Bag of tricks for training deeper graph neural networks: A comprehensive benchmark study. IEEE TPAMI.
- Chung, F. R. (1997). Spectral graph theory (Vol. 92). American Mathematical Soc..
- Devriendt, K., & Lambiotte, R. (2022). Discrete curvature on graphs from the effective resistance. Journal of Physics: Complexity, 3(2), 025008.
- Di Giovanni, F., Giusti, L., Barbero, F., Luise, G., Lio, P., & Bronstein, M. M. (2023, July). On over-squashing in message passing neural networks: The impact of width, depth, and topology. ICML
- Di Giovanni, F., Rowbottom, J., Chamberlain, B. P., Markovich, T., & Bronstein, M. M. (2023). Understanding convolution on graphs via energies. In TLMR.
- Di Giovanni, F., Rusch, T. K., Bronstein, M. M., Deac, A., Lackenby, M., Mishra, S., & Veličković, P. (2024). How does over-squashing affect the power of GNNs?. TMLR.
- Dwivedi, V. P., Rampášek, L., Galkin, M., Parviz, A., Wolf, G., Luu, A. T., & Beaini, D. (2022). Long range graph benchmark. Advances in Neural Information Processing Systems, 35, 22326-22340.
- Eliasof, Moshe, Eldad Haber, and Eran Treister. "Pde-gcn: Novel architectures for graph neural networks motivated by partial differential equations." Advances in neural information processing systems 34 (2021): 3836-3849.
- Errica, F., Christiansen, H., Zaverkin, V., Maruyama, T., Niepert, M., & Alesiani, F. (2024). Adaptive Message Passing: A General Framework to Mitigate Oversmoothing, Oversquashing, and Underreaching. In JMLR 2024
- Fesser, L., & Weber, M. (2024, April). Mitigating over-smoothing and over-squashing using augmentations of Forman-Ricci curvature. In Learning on Graphs Conference (pp. 19-1). PMLR.
- Gasteiger, J., Weißenberger, S., & Günnemann, S. (2019). Diffusion improves graph learning. Advances in neural information processing systems, 32.
- Giraldo, J. H., Skianis, K., Bouwmans, T., & Malliaros, F. D. (2023, October). On the trade-off between over-smoothing and over-squashing in deep graph neural networks. In ICKM.
- Gutteridge, B., Dong, X., Bronstein, M. M., & Di Giovanni, F. (2023, July). Drew: Dynamically rewired message passing with delay. In International Conference on Machine Learning (pp. 12252-12267). PMLR.

References

- Hamilton, W. L. (2020). Graph representation learning. Morgan & Claypool Publishers.
- Hasanzadeh, A., Hajiramezanali, E., Boluki, S., Zhou, M., Duffield, N., Narayanan, K., & Qian, X. (2020, November). Bayesian graph neural networks with adaptive connection sampling. In ICML 2022.
- Huang, K., Wang, Y. G., & Li, M. (2024). How Universal Polynomial Bases Enhance Spectral Graph Neural Networks: Heterophily, Over-smoothing, and Over-squashing. arXiv preprint arXiv:2405.12474.
- Jamadandi, Adarsh, Celia Rubio-Madrigal, and Rebekka Burkholz. "Spectral Graph Pruning Against Over-Squashing and Over-Smoothing." arXiv preprint arXiv:2404.04612 (2024).
- Karhadkar, K., Banerjee, P. K., & Montúfar, G. (2022). FoSR: First-order spectral rewiring for addressing oversquashing in GNNs. In ICLR 2023
- Jiang, W., Liu, H., & Xiong, H. (2023). Survey on Trustworthy Graph Neural Networks: From A Causal Perspective. arXiv preprint arXiv:2312.12477.
- Keriven, N. Not too little, not too much: a theoretical analysis of graph (over) smoothing. NeurIPS 2022.
- Kondor, R. I., & Lafferty, J. (2002, July). Diffusion kernels on graphs and other discrete structures. In Proceedings of the 19th international conference on machine learning (Vol. 2002, pp. 315-322).
- Li, Q., Han, Z., & Wu, X. M. (2018, April). Deeper insights into graph convolutional networks for semi-supervised learning. In Proceedings of the AAAI conference on artificial intelligence (Vol. 32, No. 1).
- Li, G., Muller, M., Thabet, A., & Ghanem, B. (2019). Deepgcns: Can gcns go as deep as cnns?. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 9267-9276).
- Liu, M., Gao, H., & Ji, S. (2020, August). Towards deeper graph neural networks. In Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining (pp. 338-348).
- Liu, Y., Zhou, C., Pan, S., Wu, J., Li, Z., Chen, H., & Zhang, P. (2023, April). Curvdrop: A ricci curvature based approach to prevent graph neural networks from over-smoothing and over-squashing. In WWW.
- Liu, Y., Zheng, Y., Zhang, D., Lee, V. C., & Pan, S. (2023b, June). Beyond smoothing: Unsupervised graph representation learning with edge heterophily discriminating. AAAI.
- Lim, D., et al. "New benchmarks for learning on non-homophilous graphs“. In WWW Workshop on GLB, 2021.
- Lovász, L. (1993). Random walks on graphs. Combinatorics, Paul erdos is eighty, 2(1-46), 4.
- Luan, S., Hua, C., Lu, Q., Zhu, J., Zhao, M., Zhang, S., ... & Precup, D. (2022). Revisiting heterophily for graph neural networks. Advances in neural information processing systems, 35, 1362-1375.
- Luan, S. et al (2024). The Heterophilic Graph Learning Handbook: Benchmarks, Models, Theoretical Analysis, Applications and Challenges. Arxiv 2407.09618
- Ma, Y., Liu, X., Shah, N., & Tang, J. (2022). Is homophily a necessity for graph neural networks?. ICLR 2022.
- Maskey, S., Paolino, R., Bacho, A., & Kutyniok, G. (2024). A fractional graph laplacian approach to oversmoothing. Advances in Neural Information Processing Systems, 36.
- Newman, M. “Assortative mixing in networks”. Phys. Rev. Lett., 89, 2002.
- Nguyen, K., Hieu, N. M., Nguyen, V. D., Ho, N., Osher, S., & Nguyen, T. M. (2023, July). Revisiting over-smoothing and over-squashing using ollivier-ricci curvature. In ICML 2023.
- Oono, K., & Suzuki, T. (2019). Graph neural networks exponentially lose expressive power for node classification. In ICLR 2020.
- Pei, H. et al. “Geom-GCN: Geometric GCNs”. In ICLR, 2019.
- Pham, T., Tran, T., Dam, H., & Venkatesh, S. (2017). Graph classification via deep learning with virtual nodes. arXiv preprint arXiv:1708.04357.
- Qiu, H., & Hancock, E. R. (2006). Graph embedding using commute time. In Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops, SSPR 2006 and SPR 2006,
- Qian, Y., Expert, P., Rieu, T., Panzarasa, P., & Barahona, M. (2021). Quantifying the alignment of graph and features in deep learning. IEEE TNNLS
- Rong, Y., Huang, W., Xu, T., & Huang, J. (2020). Dropedge: Towards deep graph convolutional networks on node classification. In ICLR 2020.
- Rusch, T. K., Bronstein, M. M., & Mishra, S. (2023). A survey on oversmoothing in graph neural networks. arXiv preprint arXiv:2303.10993.
- Rusch, T. K., Chamberlain, B., Rowbottom, J., Mishra, S., & Bronstein, M. (2022, June). Graph-coupled oscillator networks. In International Conference on Machine Learning (pp. 18888-18909). PMLR.
- Southern, J., Di Giovanni, F., Bronstein, M., & Lutzeyer, J. F. (2024). Understanding Virtual Nodes: Oversmoothing, Oversquashing, and Node Heterogeneity. arXiv preprint arXiv:2405.13526.
- Shao, Z., Shi, D., Han, A., Guo, Y., Zhao, Q., & Gao, J. (2023). Unifying over-smoothing and over-squashing in graph neural networks: A physics informed approach and beyond. arXiv preprint arXiv:2309.02769.

References

- Spielman D. (2018). Spectral Graph Theory, Lecture 10: Random Walks on Graphs. Lecture at Yale https://www.cs.yale.edu/homes/spielman/561/lect10-18.pdf
- Topping, J., Di Giovanni, F., Chamberlain, B. P., Dong, X., & Bronstein, M. M. (2021). Understanding over-squashing and bottlenecks on graphs via curvature. ICLR 2022.
- Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks.
- Xhonneux, L. P., Qu, M., & Tang, J. (2020, November). Continuous graph neural networks. In International conference on machine learning (pp. 10432-10441). PMLR.
- Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K. I., & Jegelka, S. (2018, July). Representation learning on graphs with jumping knowledge networks. In International conference on machine learning (pp. 5453-5462). PMLR.
- Zheng, C., Zong, B., Cheng, W., Song, D., Ni, J., Yu, W., ... & Wang, W. (2020, November). Robust graph representation learning via neural sparsification. In International Conference on Machine Learning (pp. 11458-11468). PMLR.
- Zheng, Y., Luan, S., & Chen, L. (2024). What Is Missing In Homophily? Disentangling Graph Homophily For Graph Neural Networks. arXiv preprint arXiv:2406.18854.
- Zhao, L., & Akoglu, L. (2020). Pairnorm: Tackling oversmoothing in gnns. In ICLR 2020.
- Zhao, J., Dong, Y., Tang, J., Ding, M., & Wang, K. (2021). Generalizing graph convolutional networks via heat kernel.
- Zhou, K., Huang, X., Li, Y., Zha, D., Chen, R., & Hu, X. (2020). Towards deeper graph neural networks with differentiable group normalization. Advances in neural information processing systems, 33, 4917-4928.
- Zhou, K., Huang, X., Zha, D., Chen, R., Li, L., Choi, S. H., & Hu, X. (2021). Dirichlet energy constrained learning for deep graph neural networks. Advances in Neural Information Processing Systems, 34, 21834-21846.
- Zhou, K., Dong, Y., Wang, K., Lee, W. S., Hooi, B., Xu, H., & Feng, J. (2021b, October). Understanding and resolving performance degradation in deep graph convolutional networks. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management (pp. 2728-2737).
- Zhu, J., et al. “Beyond homophily in graph neural networks: Current limitations and effective designs”. in NeurIPS, 2020
- Zhu, Y., Xu, W., Zhang, J., Du, Y., Zhang, J., Liu, Q., ... & Wu, S. (2021). A survey on graph structure learning: Progress and opportunities. arXiv preprint arXiv:2103.03036.
- Zhu, Y., Du, Y., Wang, Y., Xu, Y., Zhang, J., Liu, Q., & Wu, S. (2022, December). A survey on deep graph generation: Methods and applications. In Learning on Graphs Conference (pp. 47-1). PMLR.



Adrián Arnaiz-Rodríguez

adrian@ellisalicante.org
@arnaiztech

ellis ALICANTE unit

Ameya Velingker

ameyav@google.com
@ameya_pa

Google Research

Thanks to all collaborators, advisors and colleagues

Thanks for the discussions with Petar Veličković, Chistopher Morris, Karel Devriendt, Chaitanya Joshi, Soledad Villar, Nuria Oliver, Rishabh Anand, Christian Koke, Francesco Di Giovanni, Federico Barbero, Lorenzo Giusti, Federico Errica and Josh Southern

https://icml.cc/virtual/2024/tutorial/35233
https://icml2024graphs.ameyavelingker.com/

Google Research GENERALITAT VALENCIANA Conselleria de Innovación, Universidades, Ciencia y Sociedad Digital intel ELIAS Sabadell Fundación

Graph Learning: Principles, Challenges, and Open Directions

ICML 2024 - 22/07/2024

Michael Bronstein
DeepMind Professor of
Artificial Intelligence

Michael Galkin
AI Research Scientist
at Intel Labs

Christopher Morris
Assistant Professor
at RWTH Aachen
University

Bryan Perozzi
Research Scientist at
Google Research

Panel Discussion

Open questions and challenges on GNNs
Graphs + LLMs
Graph Foundation Models