CISPA
HELMHOLTZ CENTER FOR INFORMATION SECURITY

Learning on Graphs: Strategies for Improving and Understanding Generalization in Graph Neural Networks

Adarsh Jamadandi | 06/June/2025 |
CEVI, KLE Technological University, Hubli, India.

Will be based on

Spectral Graph Pruning Against Over-squashing and Over-smoothing.

Adarsh Jamadandi*, Celia Rubio-Madrigal* and Rebekka Burkholz, **NeurIPS'2024.**

Celia Rubio-Madrigal

Rebekka Burkholz

Relational-ML Group

Will be based on

Finding Memo(rization) in Graph Neural Networks.

Adarsh Jamadandi*, Jing Xu, Adam Dziedzic and Franziska Boenisch.

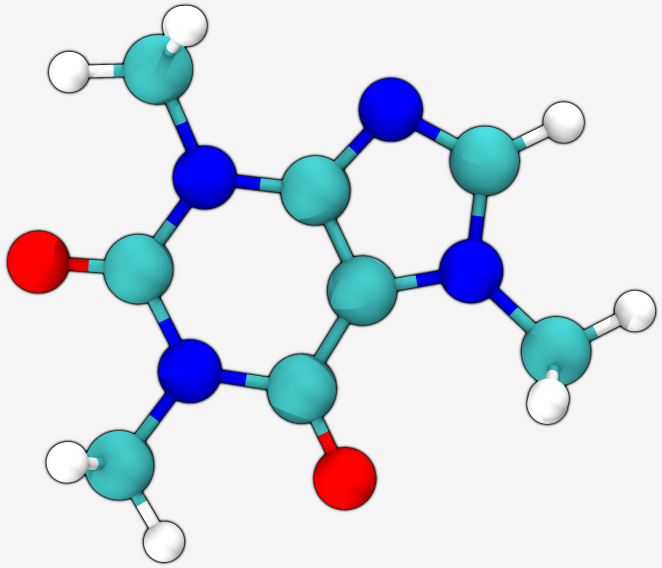
Pre-print 2025 (Under-review).

Jing Xu

Adam Dziedzic

Franziska Boenisch

SprintML Group

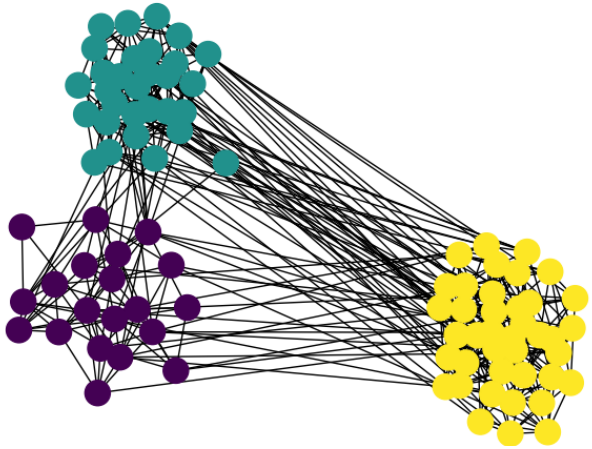


A Brief Primer on Graph Neural Networks

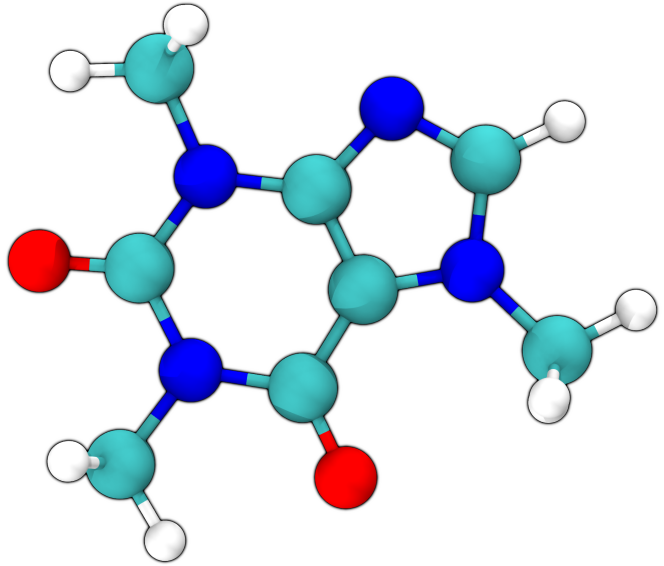
Why Graphs? Deconstructing Graphs and Deep Learning on Graphs.

Why Graphs?

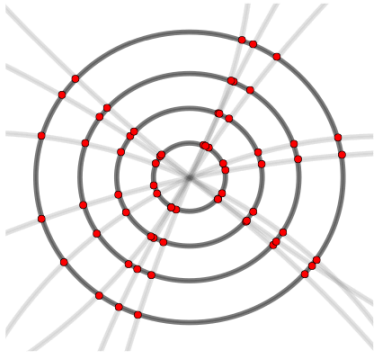
- Graphs are everywhere!
- Social Networks, Biological Networks, Molecules and even High-energy particle interactions can be modelled as graphs.

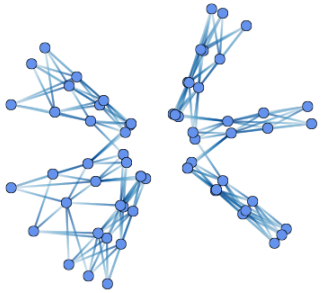


Social Networks



Chemical Molecules

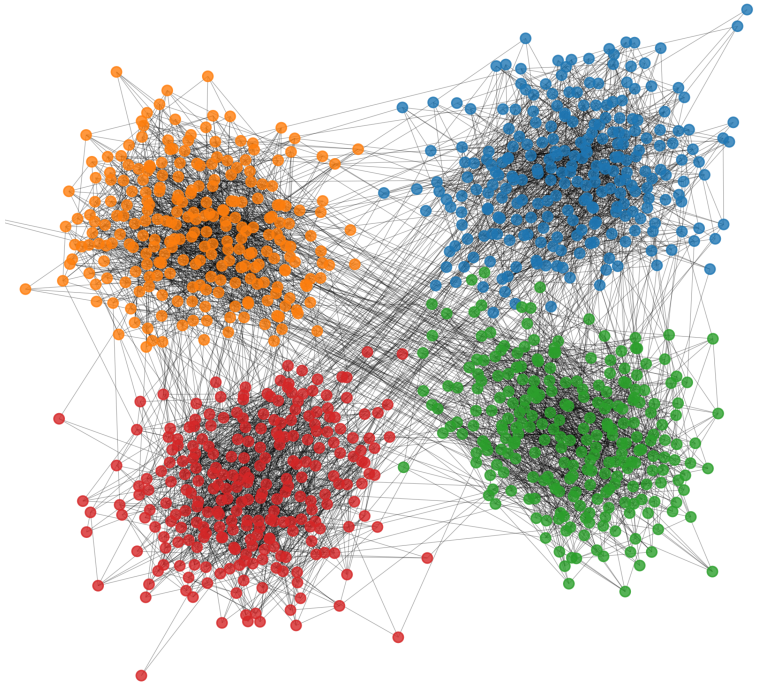




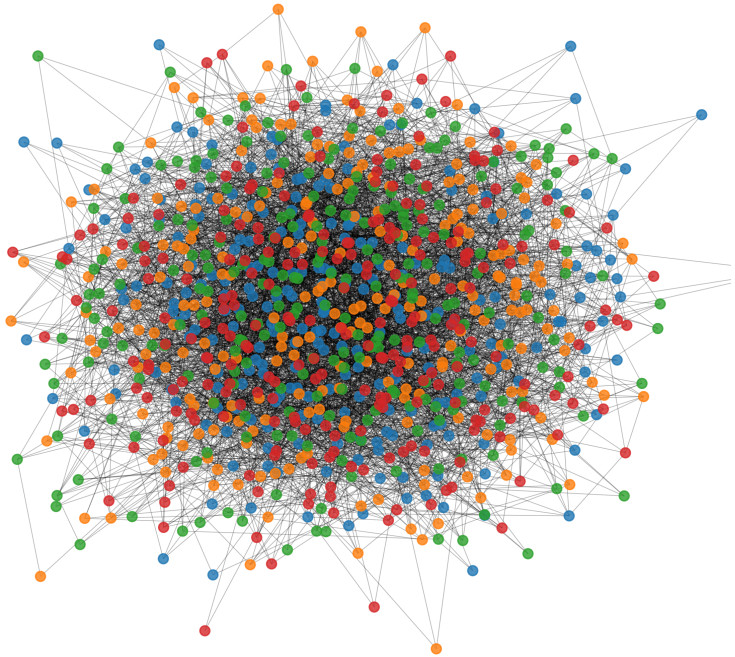
High Energy Physics

Deconstructing Graphs

- Graphs can be of two types - homophilic and heterophilic.
- Depends on whether the neighborhood nodes have similar features/labels.



Homophilic Graphs

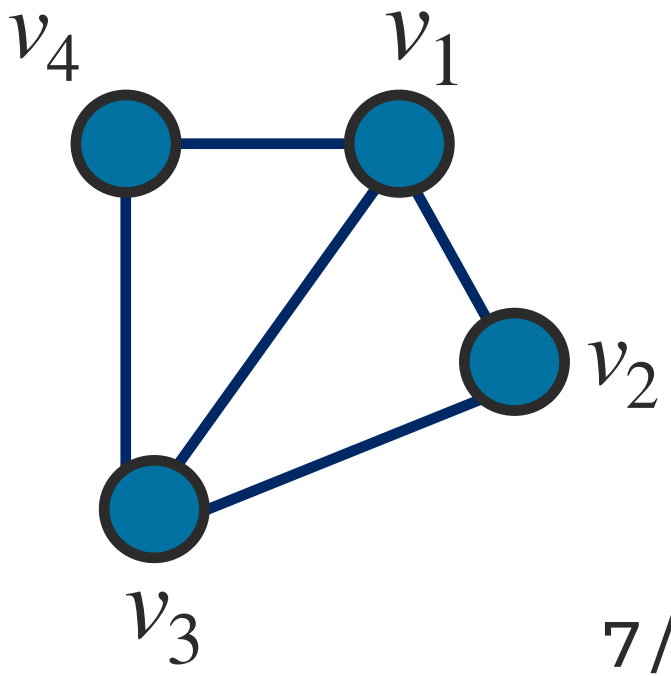


Heterophilic Graphs

Deconstructing Graphs

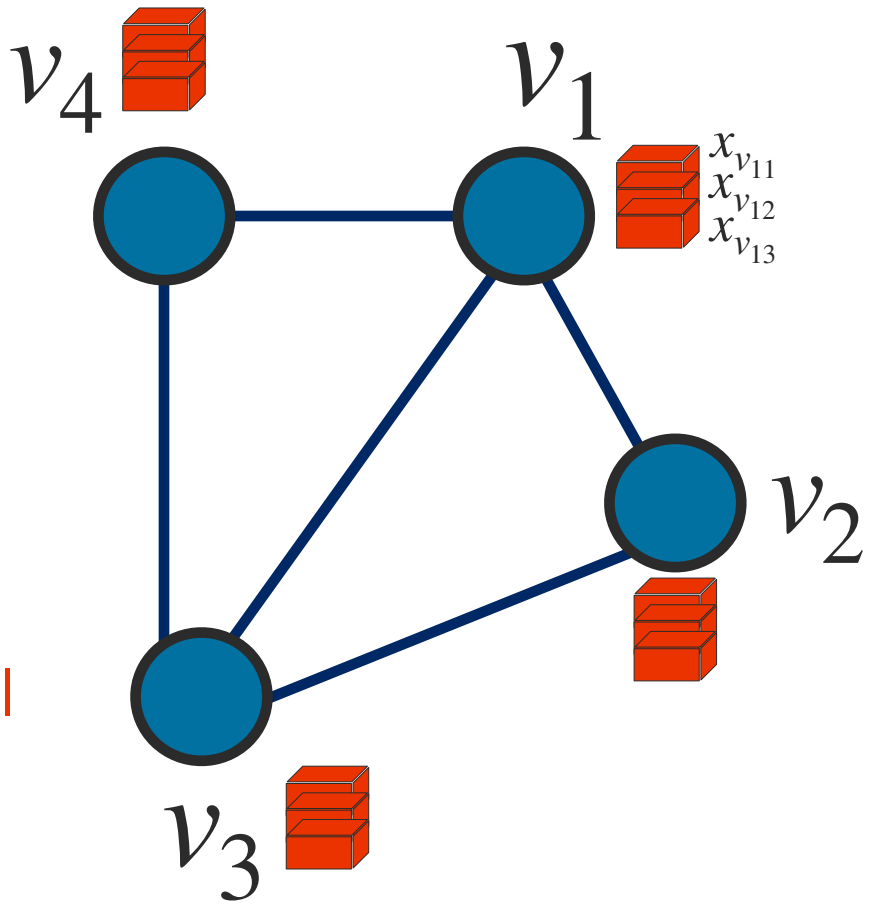
- Connectivity information - Adjacency Matrix ($\mathbf{A}$).
- Degree Matrix ($\mathbf{D}$).
- Laplacian Matrix ($\mathscr{L} = \mathbf{D} - \mathbf{A}$)
- Symmetric Normalized Laplacian ($\mathscr{L}_G = D^{-1/2}\mathscr{L}D^{-1/2}$)

$$\mathbf{A} = \begin{pmatrix} & v_1 & v_2 & v_3 & v_4 \\ v_1 & 0 & 1 & 1 & 1 \\ v_2 & 1 & 0 & 1 & 0 \\ v_3 & 1 & 1 & 0 & 1 \\ v_4 & 1 & 0 & 1 & 0 \end{pmatrix} \quad \mathbf{D} = \begin{pmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}$$



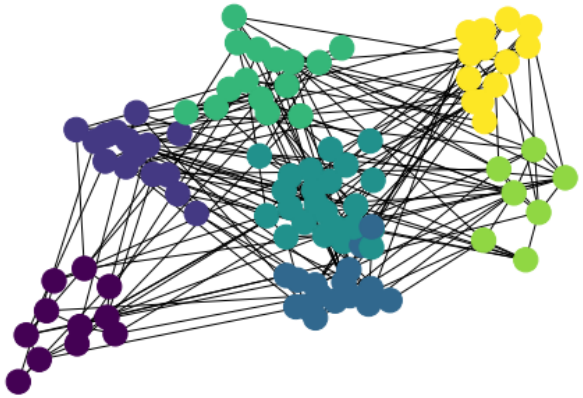
Deep Learning on Graphs

- Given a graph $G = (V, E)$, with $|V|$ vertices and $|E|$ edges and associated node features, $x_{v_i} \in \mathbf{X}$.
- Node prediction - predict labels for the nodes Y_{v_i}.
- Graph prediction - predict a global label Y_G for the entire graph.

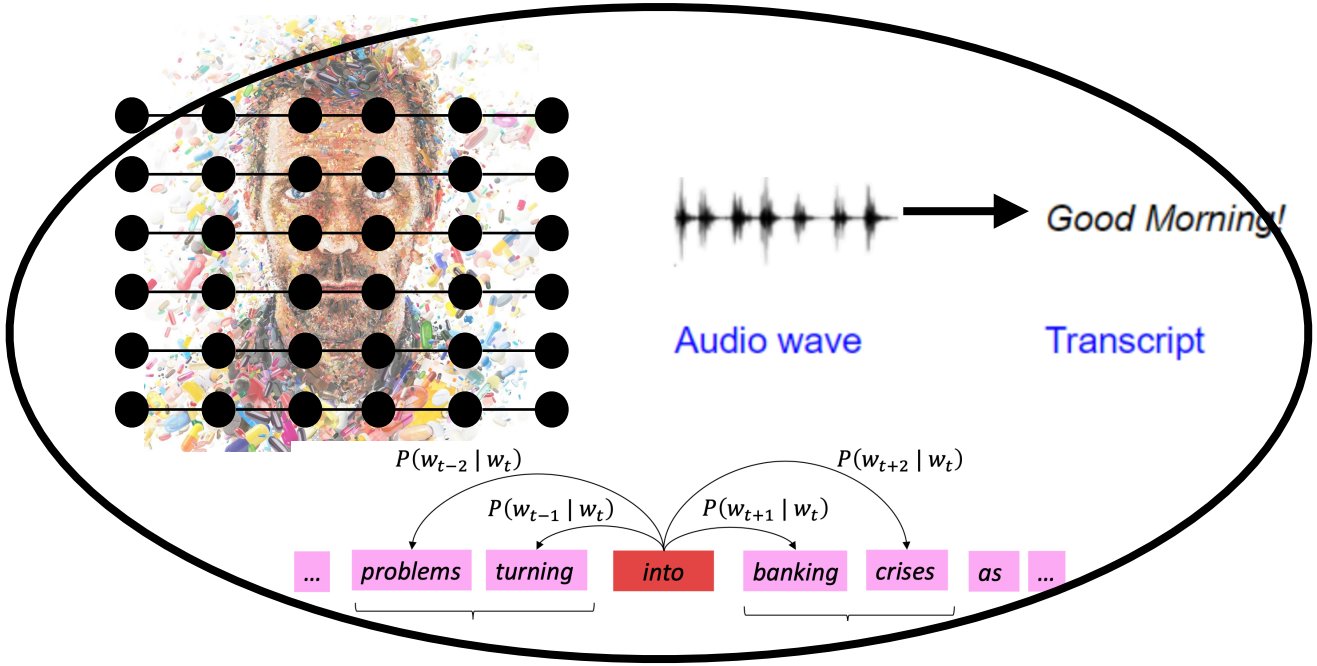


Deep Learning on Graphs

- Regular deep learning algorithms don't work on GNNs. Why?
- No fixed ordering. Complex topological structure.



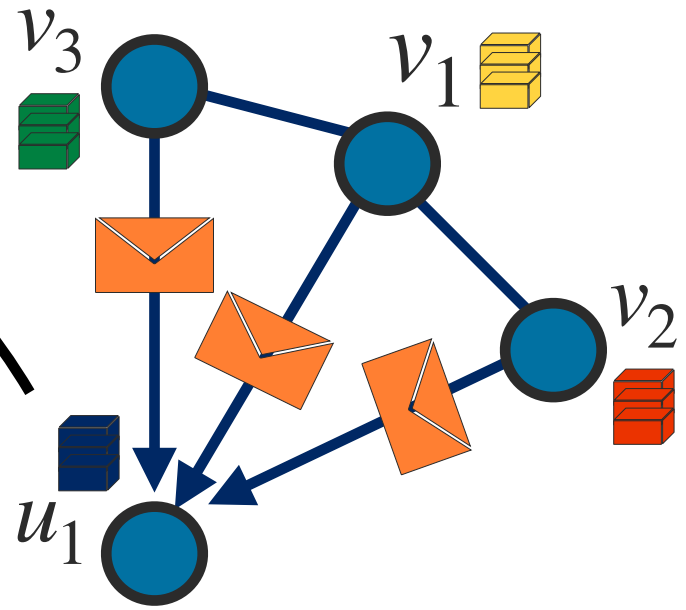
VS.



Message Passing

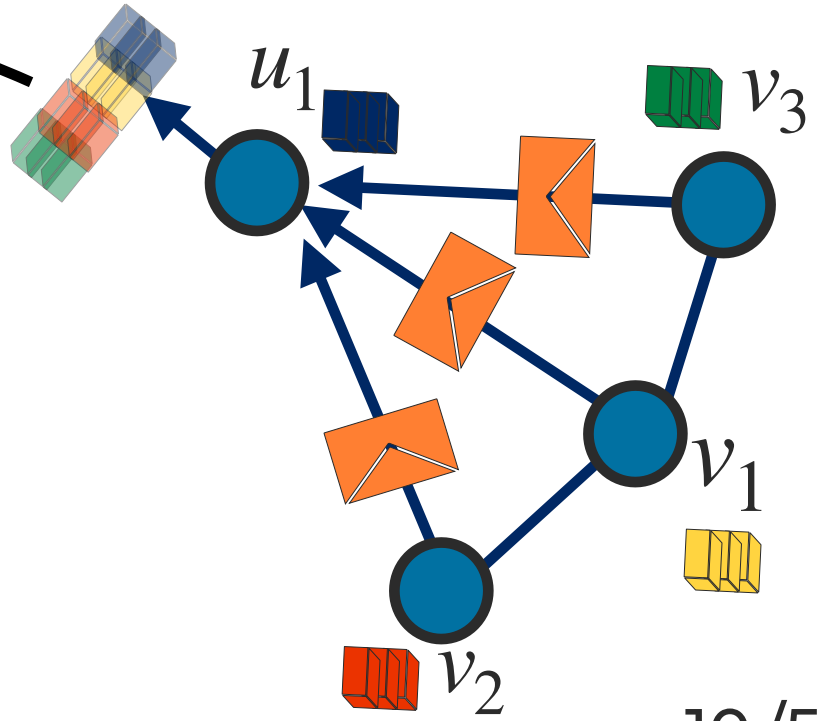
Compute Messages

$$\mathbf{m}_u^{(l)} = \text{MSG}^{(l)}(\mathbf{h_u}^{(l-1)}) \qquad u \in \{\mathcal{N}(v) \cup v\}$$



Aggregate Messages

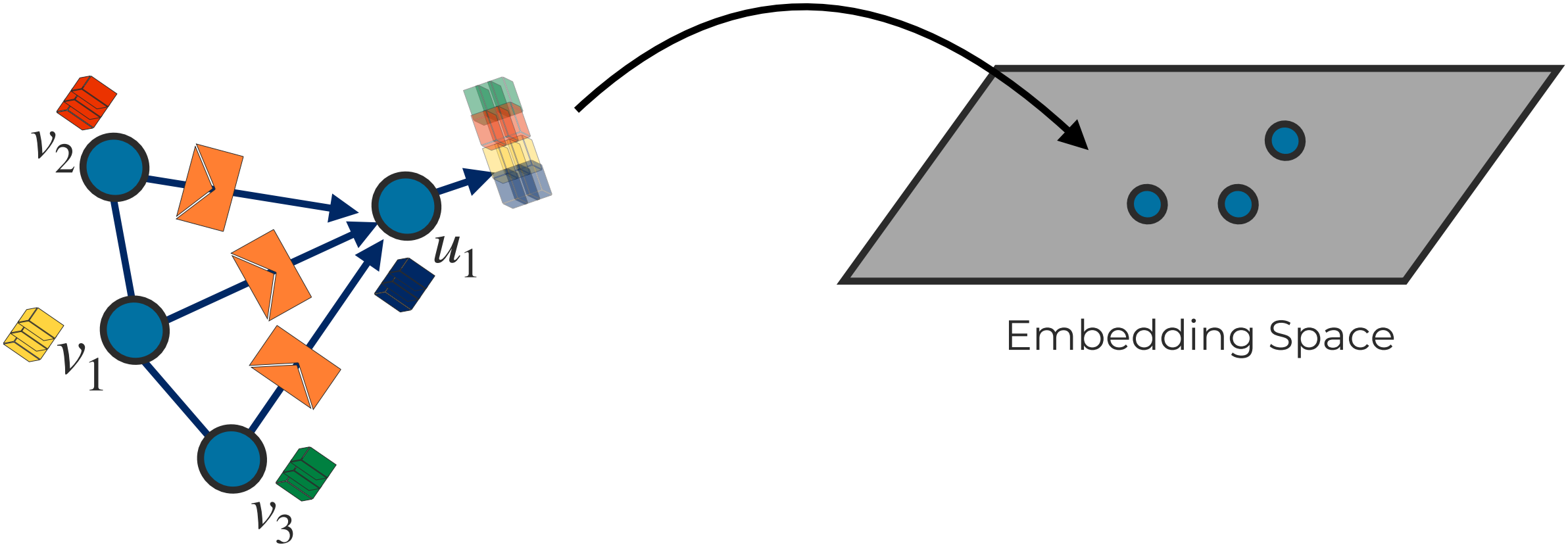
$$\mathbf{h}_v^{(l)} = \text{AGG}^{(l)}\left(\{\mathbf{m}_u^l, u \in \mathcal{N}(v)\}, \mathbf{m}_v^{(l)}\right)$$



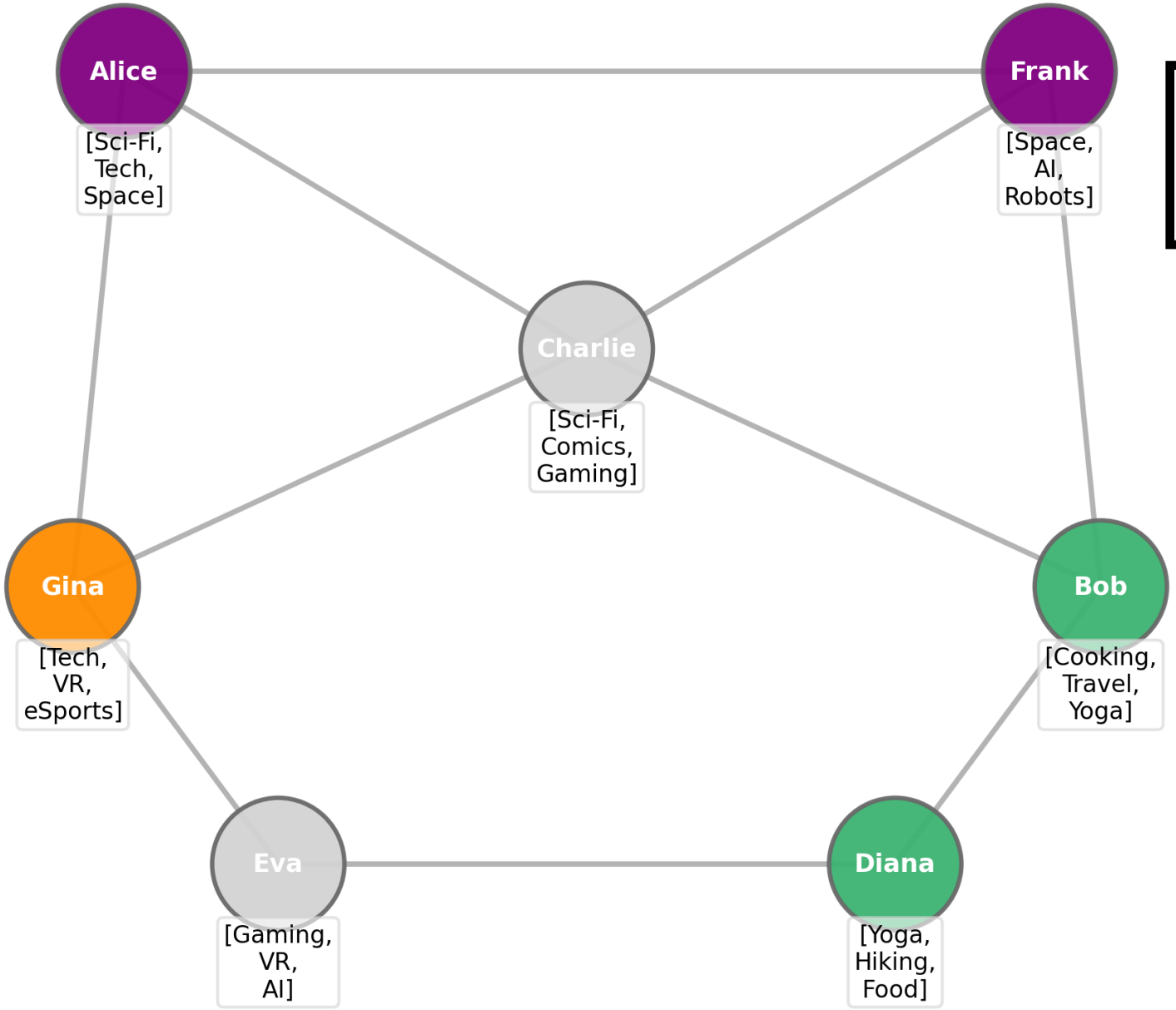
Credits: Stanford CS224W.

Graph Representation Learning

Final Graph level representation



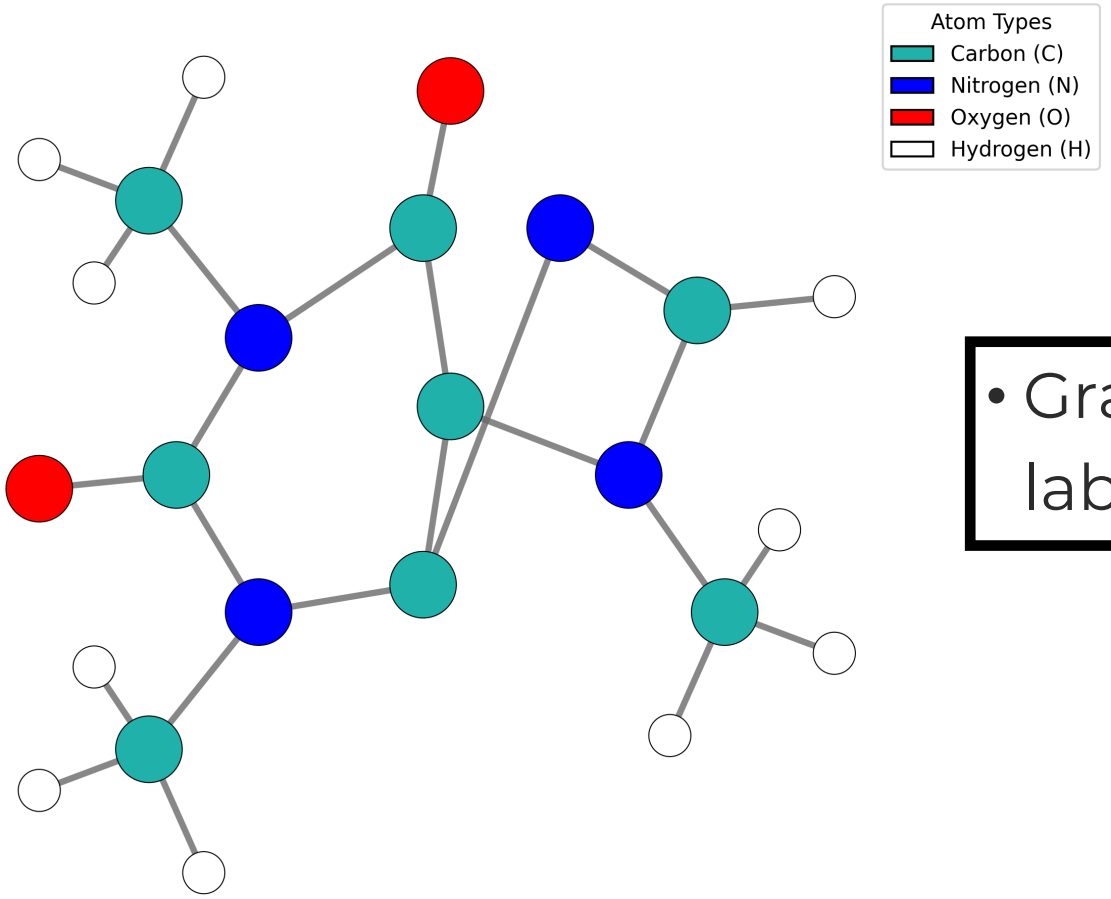
Node Classification Real-World Example



- Node prediction - predict labels for the nodes Y_{v_i}.

User Categories
- Sci-Fi Fan
- Lifestyle Buff
- Tech & Gaming
- Label to Predict

Graph Classification Real-World Example

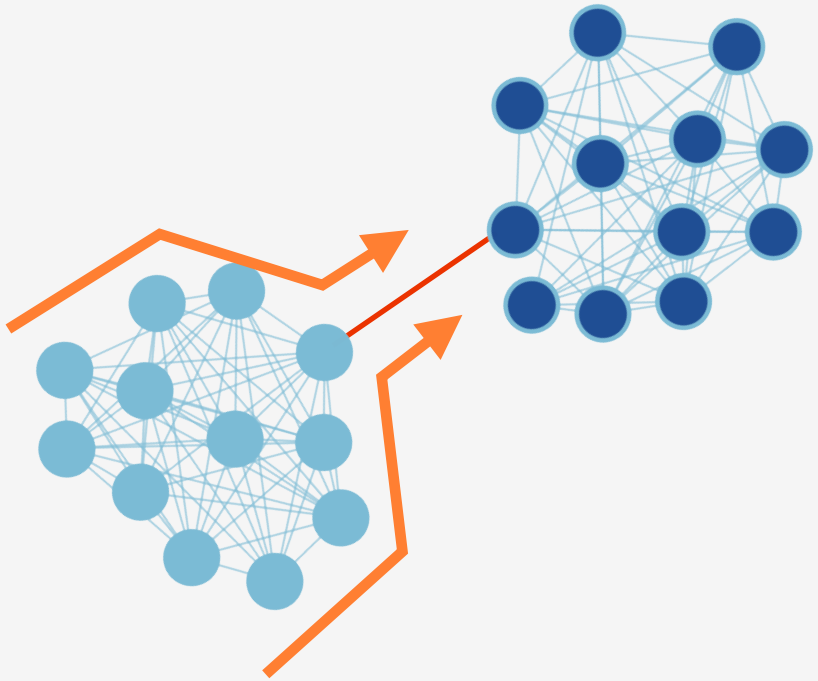


Predicted: Caffeine

- Graph prediction - predict a global label Y_G for the entire graph.

TL;DR

- **Graphs are everywhere!**
- **Deep Learning on Graphs → Message Passing Paradigm.**

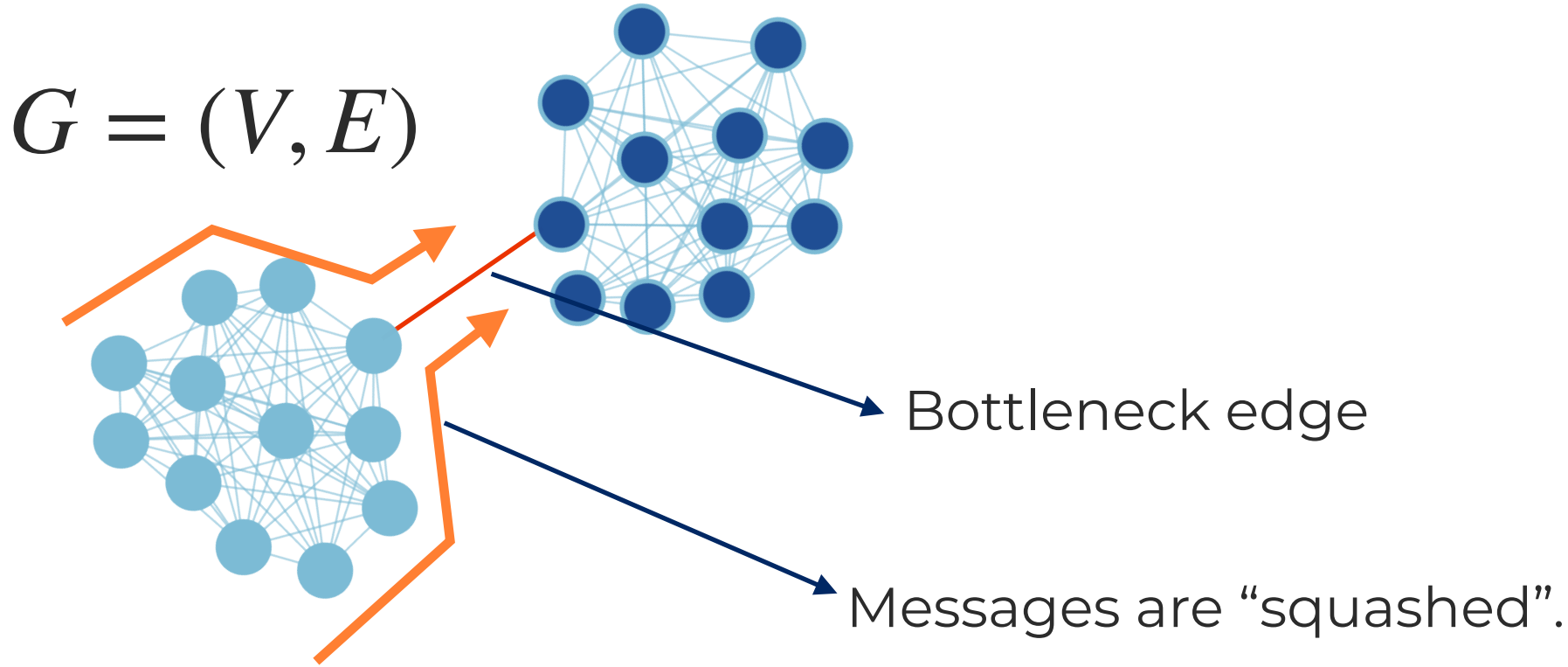


Factors Affecting Message Passing

Over-squashing and Over-smoothing.

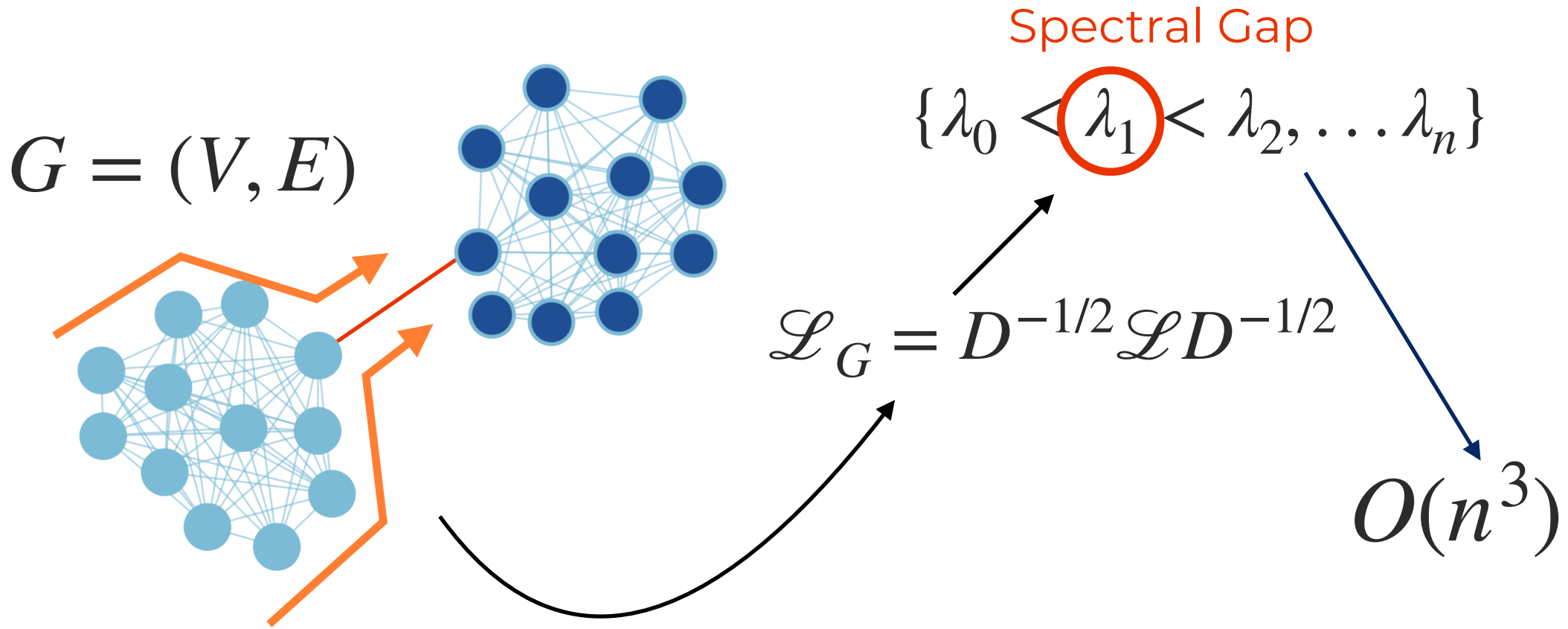
Over-squashing

- Presence of *topological bottlenecks* in the input graph can cause information congestion.
- GNN fails to model long-range interactions.



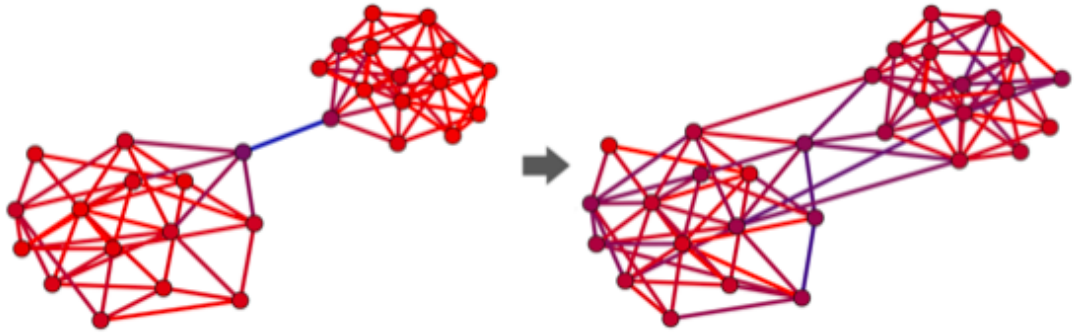
Characterizing Over-squashing

- Lets look at the graph spectrum.
- Smaller spectral gap indicates presence of bottlenecks.



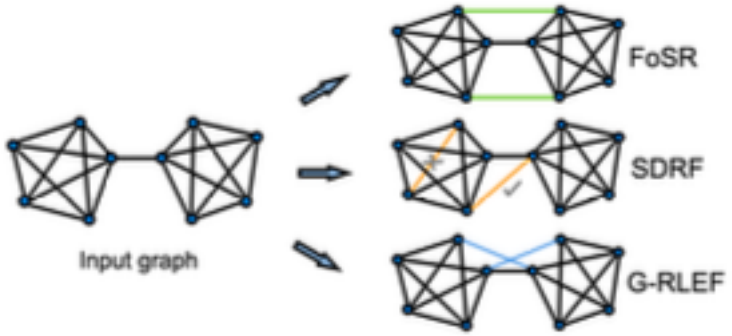
Resolving Over-squashing

- Rewire the graph based on *some criterion*.
- What kind of criterion?



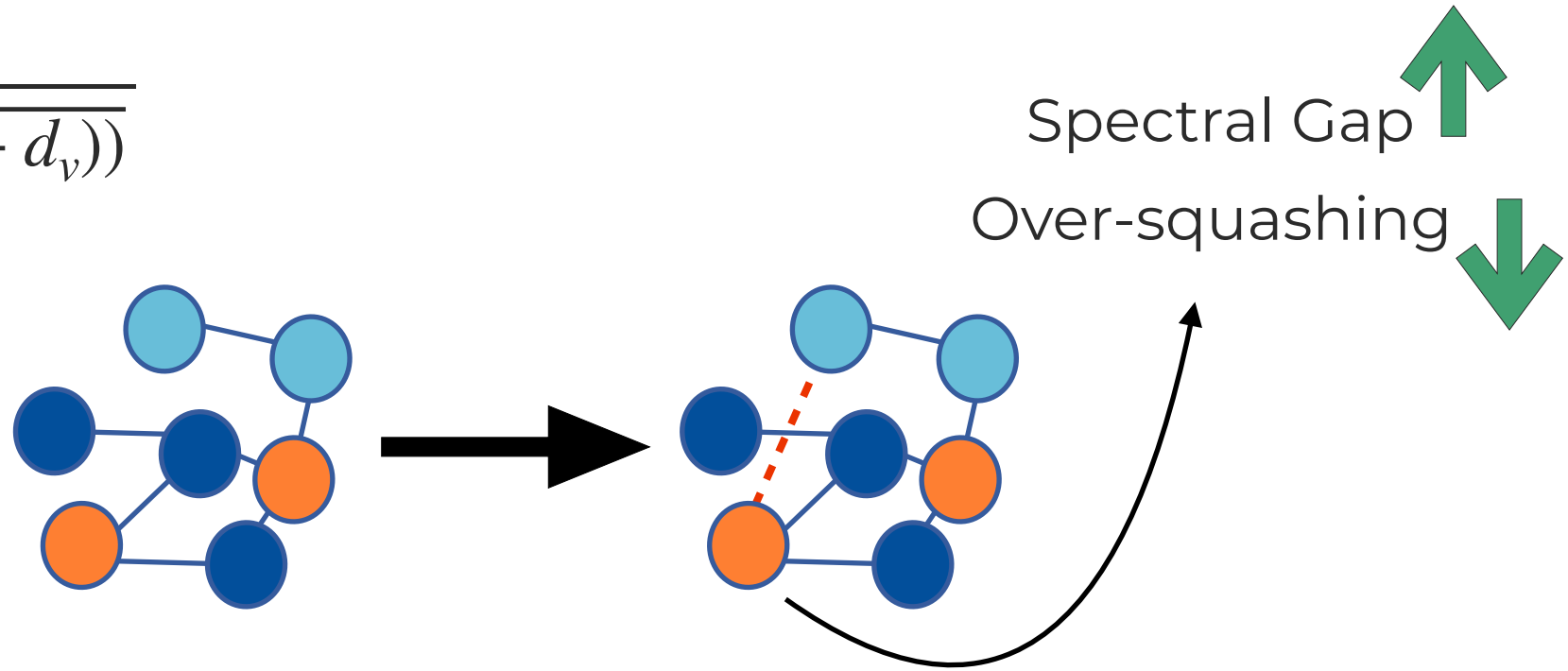
- Topping et al. ICLR, 2022 propose Stochastic Discrete Ricci Flow (SDRF) to rewire the graph.
- Computationally very expensive!

- Karhadkar et al. ICLR, 2023 propose a first order proxy of spectral gap (FoSR) to add edges to the graph to address over-squashing.



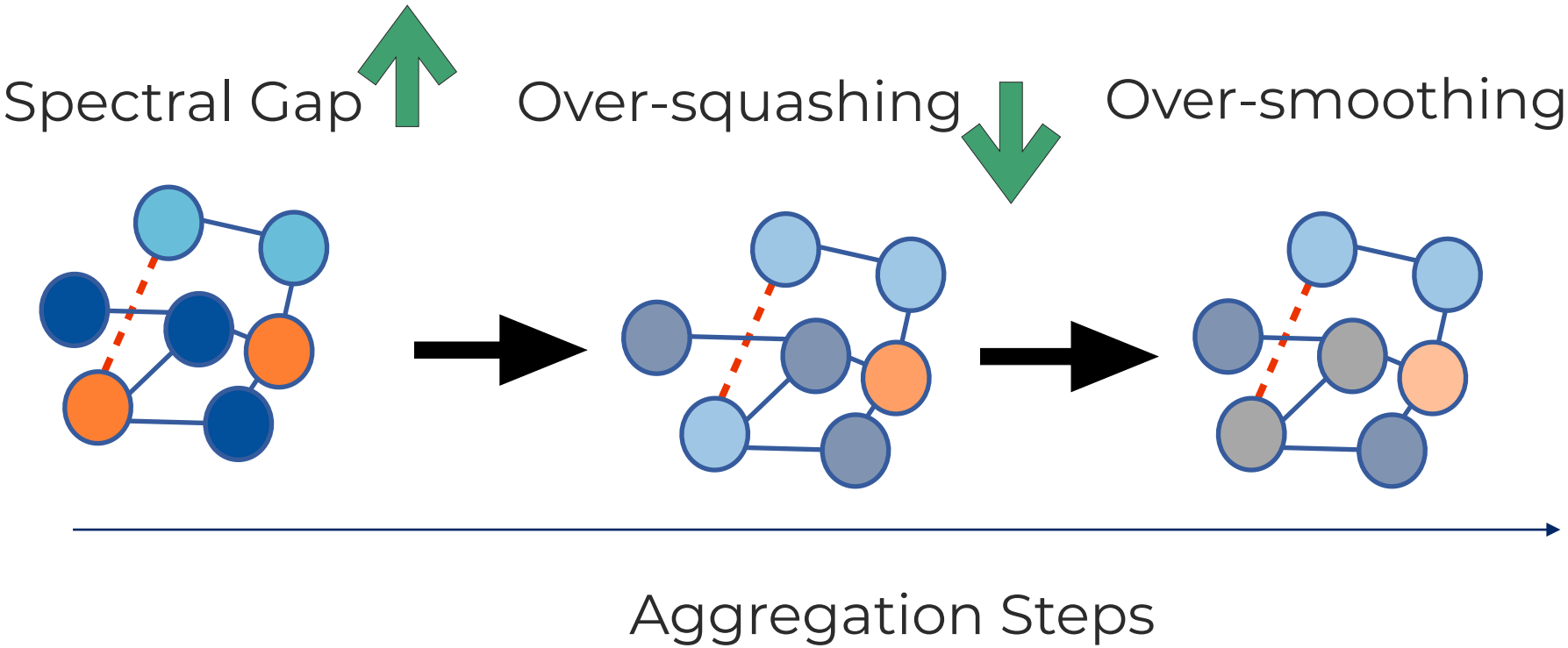
Adding Edges to Mitigate Over-squashing

- Add edges which maximize the spectral gap.
- Karhadkar et al. ICLR, 2023 propose a first order proxy of spectral gap (FoSR)
- $$\frac{2x_u x_v}{\sqrt{(1+d_u)(1+d_v)}}$$



But causes Over-smoothing

- Add edges which maximize the spectral gap.
- But will lead to over-smoothing!
- Node representations will become increasingly similar.



TL;DR

- **Information congestion → Low spectral gap →Over-squashing.**
- **Add edges → Increase spectral gap → Mitigate over-squashing.**
- **But adding edges → makes node representations same → Over-smoothing!**

Braess Paradox

How a counter-intuitive phenomenon allows to alleviate over-squashing and over-smoothing!

Braess Paradox to the Rescue!

Adding an extra road increases commute time!

Braess Paradox to the Rescue!

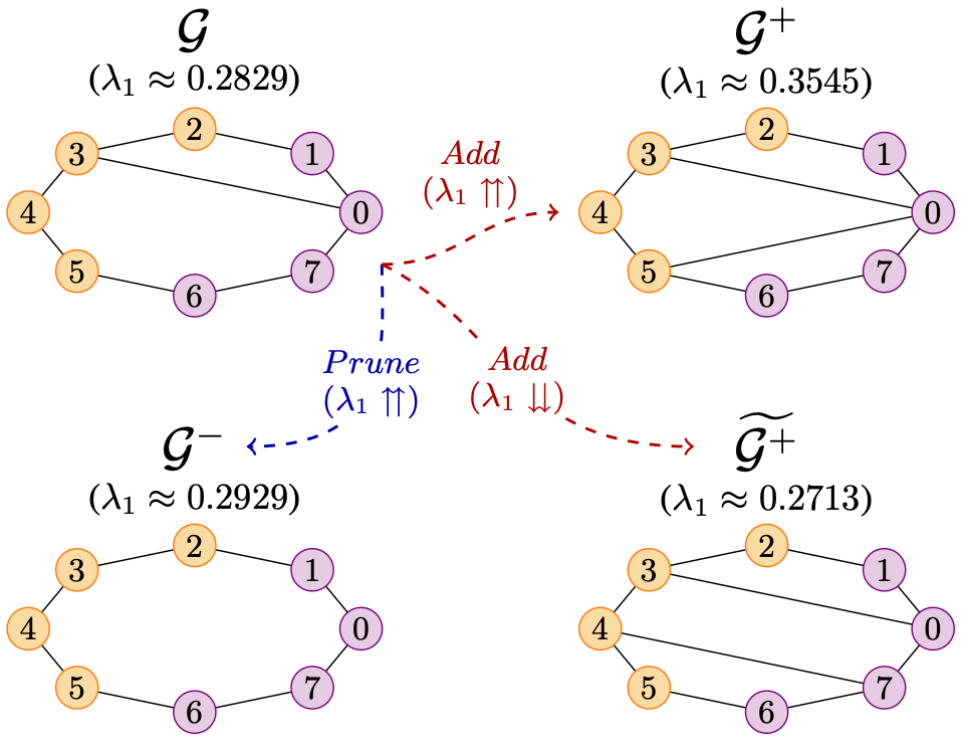
- Our first contribution - Introduce Braess Paradox in the context of Graph Neural Networks.
- Show edge deletions can also increase spectral gap!

Lemma 2.1. *Eldan et al. (2017): Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a finite graph, with f denoting the eigenvector and $\lambda_1(\mathcal{L}_\mathcal{G})$ the eigenvalue corresponding to the spectral gap. Let $\{u, v\} \notin \mathcal{V}$ be two vertices that are not connected by an edge. Denote $\hat{\mathcal{G}} = (\mathcal{V}, \hat{\mathcal{E}})$, the new graph obtained after adding an edge between $\{u, v\}$, i.e., $\hat{\mathcal{E}} := \mathcal{E} \cup \{u, v\}$. Denote with $\mathcal{P}_f := \langle f, \hat{f}_0 \rangle$ the projection of f onto the top eigenvector of $\hat{\mathcal{G}}$. Define $g(u, v, \mathcal{L}_\mathcal{G}) :=$*

$$-\mathcal{P}_f^2 \lambda_1(\mathcal{L}_\mathcal{G}) - 2(1 - \lambda_1(\mathcal{L}_\mathcal{G})) \left(\frac{\sqrt{d_u + 1} - \sqrt{d_u}}{\sqrt{d_u + 1}} f_u^2 + \frac{\sqrt{d_v + 1} - \sqrt{d_v}}{\sqrt{d_v + 1}} f_v^2 \right) + \frac{2 f_u f_v}{\sqrt{d_u + 1}\sqrt{d_v + 1}}.$$

If $g(u, v, \mathcal{L}_\mathcal{G}) > 0$, then $\lambda_1(\mathcal{L}_\mathcal{G}) > \lambda_1(\mathcal{L}_{\hat{\mathcal{G}}})$.

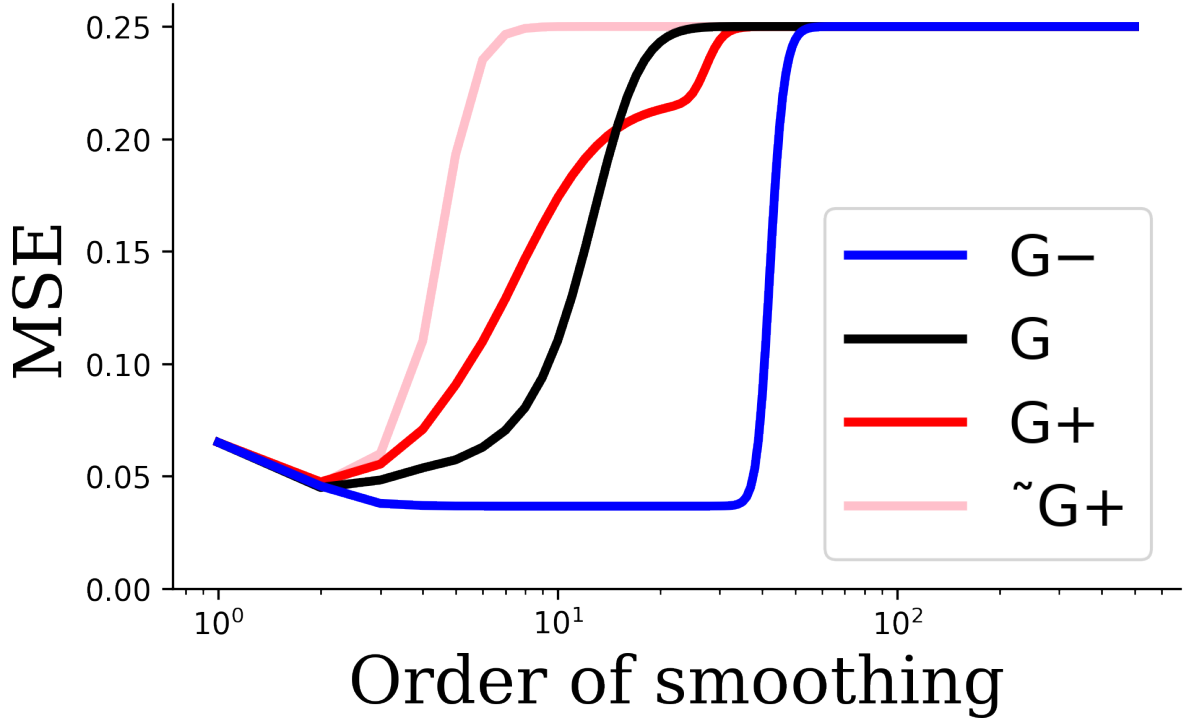
Spectral Gap Maximization via Edge Deletions



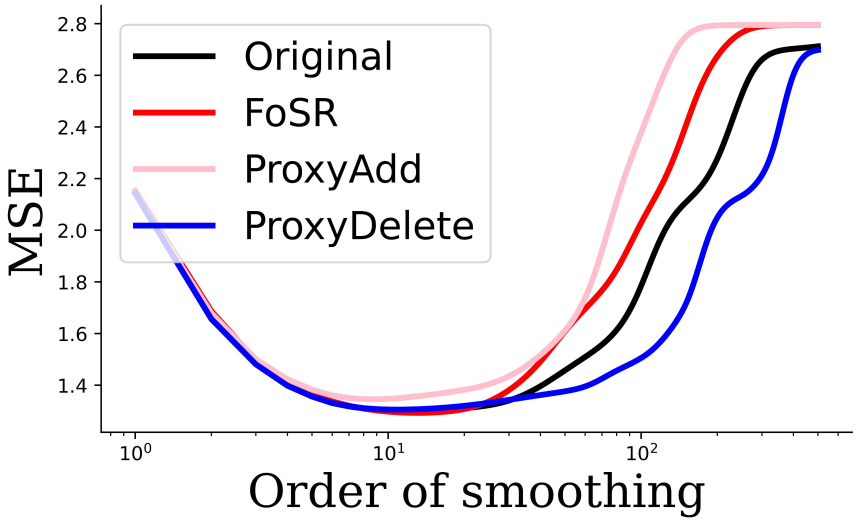
Spectral gap ↑

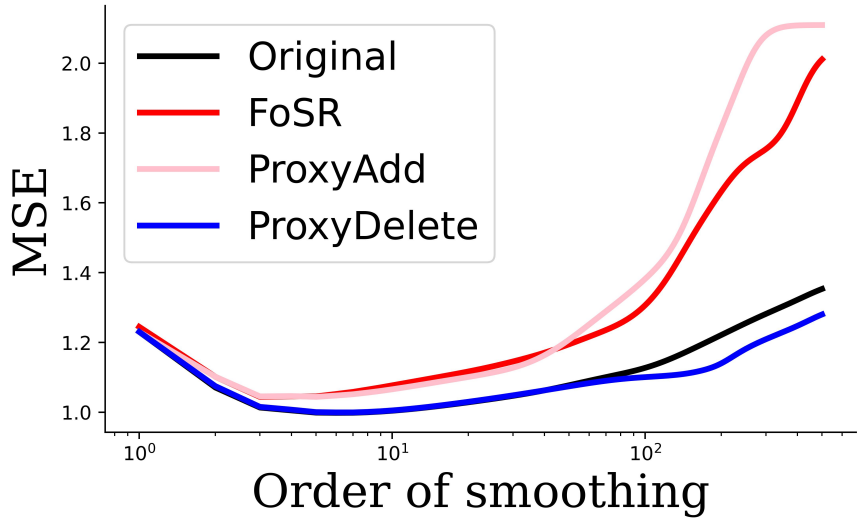
Over-squashing ↓

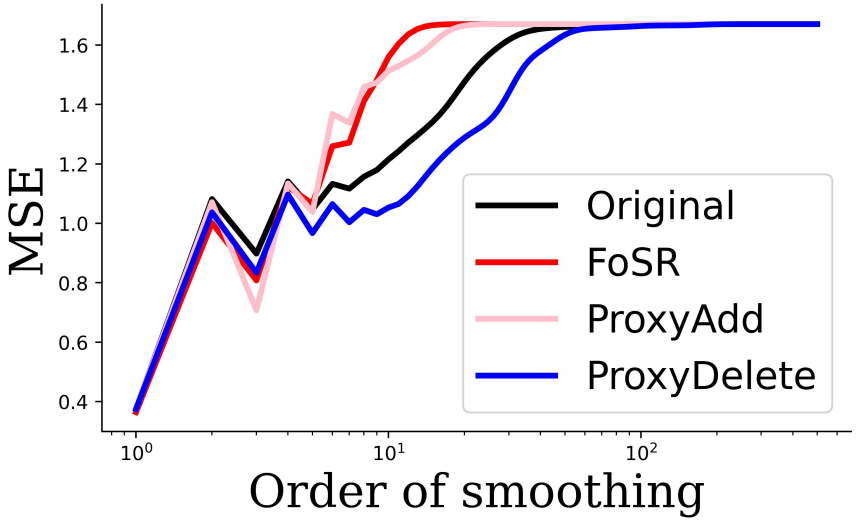
Over-smoothing ↓

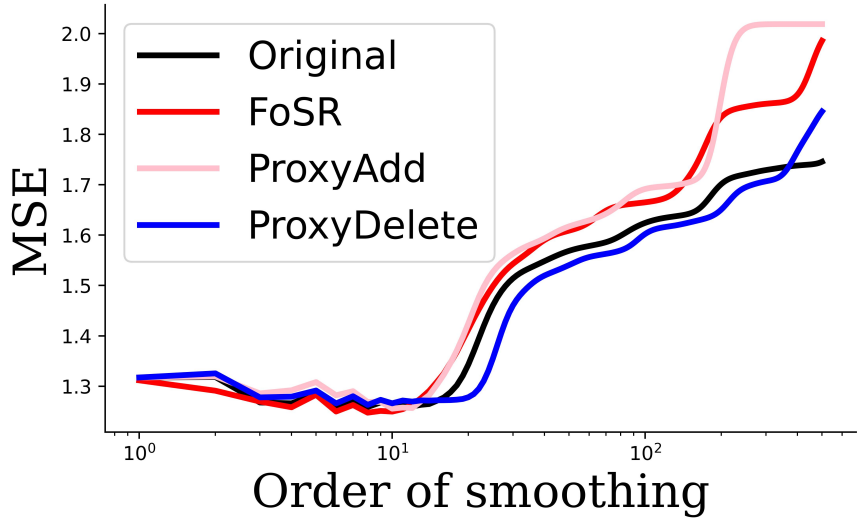


Real-world Graphs



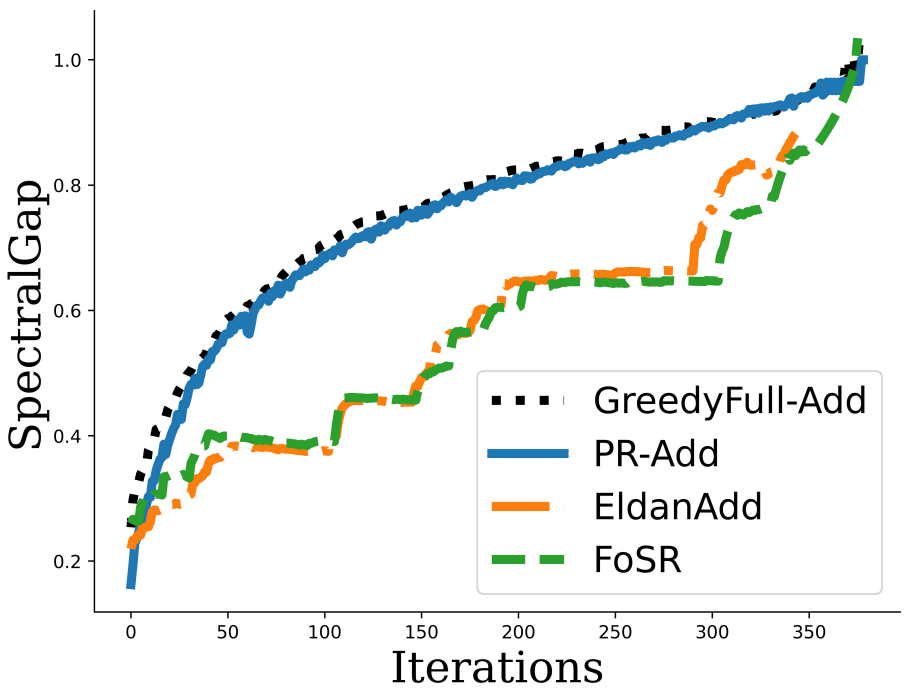


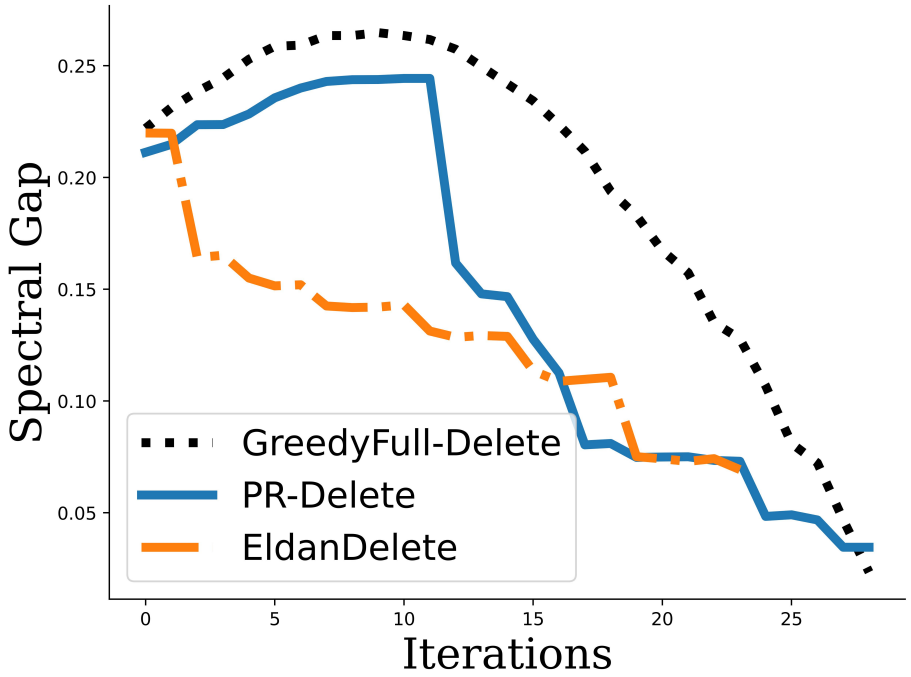




How good is the proxy?

- Use Eldan's criterion to prune the graph — Computationally expensive.
- Faster spectral gap approximation using Matrix Perturbation Theory.
- $\hat{\lambda} \approx \lambda_{init} + \Delta w_{u,v}((f_u - f_v)^2 - \lambda_{init}(f_u^2 + f_v^2))$





How fast is the proxy?

- Faster spectral gap approximation using Matrix Perturbation Theory.
- $\hat{\lambda} \approx \lambda_{init} + \Delta w_{u,v}((f_u - f_v)^2 - \lambda_{init}(f_u^2 + f_v^2))$

Method	Cora	Citeseer	Chameleon	Squirrel
FoSR	4.69	5.33	5.04	19.48
SDRF	19.63	173.92	17.93	155.95
ProxyAdd	4.30	3.13	1.15	9.12
ProxyDelete	1.18	0.86	1.46	7.26

Experiments

Dataset	Domain	Task	Node Feat. (dim)	Edge Feat. (dim)	Perf. Metric
PascalVOC-SP COCO-SP	Computer Vision	Node Classif.	Pixel + Coord (14)	Edge Weight (1 or 2)	macro F1
PCQM-Contact	Quantum Chemistry	Link Prediction	Atom Encoder (9)	Bond Encoder (3)	Hits@K, MRR
Peptides-func Peptides-struct	Chemistry	Graph Classif. Graph Regression	Atom Encoder (9)	Bond Encoder (3)	AP MAE

Table 2: Statistics of the five proposed LRGB datasets.

Dataset	Total Graphs	Total Nodes	Avg Nodes	Mean Deg.	Total Edges	Avg Edges	Avg Short.Path.	Avg Diameter
PascalVOC-SP	11,355	5,443,545	479.40	5.65	30,777,444	2,710.48	10.74±0.51	27.62±2.13
COCO-SP	123,286	58,793,216	476.88	5.65	332,091,902	2,693.67	10.66±0.55	27.39±2.14
PCQM-Contact	529,434	15,955,687	30.14	2.03	32,341,644	61.09	4.63±0.63	9.86±1.79
Peptides-func	15,535	2,344,859	150.94	2.04	4,773,974	307.30	20.89±9.79	56.99±28.72
Peptides-struct	15,535	2,344,859	150.94	2.04	4,773,974	307.30	20.89±9.79	56.99±28.72

Credit : Long Range Graph Benchmark (LRGB)

Experiments

Table 1: Results on Long Range Graph Benchmark datasets.

Method	PascalVOC-SP (Test F1 ↑)	Peptides-Func (Test AP ↑)	Peptides-Struct (Test MAE ↓)
Baseline-GCN	0.1268±0.0060	0.5930±0.0023	0.3496±0.0013
DRew+GCN	0.1848±0.0107	**0.6996±0.0076**	0.2781±0.0028
FoSR+GCN	0.2157±0.0057	0.6526±0.0014	0.2499±0.0006
ProxyAdd+GCN	**0.2213±0.0011**	0.6789±0.0002	**0.2465±0.0004**
ProxyDelete+GCN	0.2170±0.0015	0.6908±0.0007	**0.2470±0.0080**

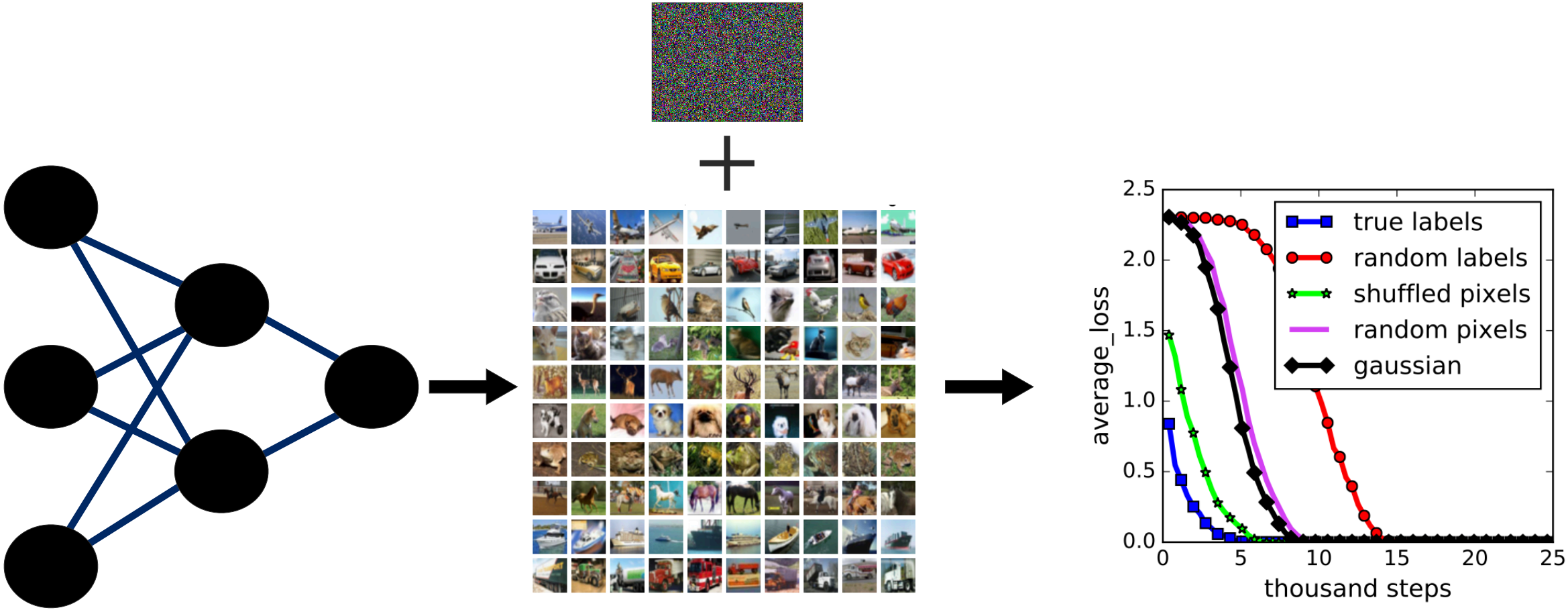
Summary

- Input graphs can have topological bottlenecks. Results in over-squashing.
- Adding edges to resolve bottlenecks results in over-smoothing.
- Edge deletions can also increase spectral gap because of Braess Paradox.
- Allows for mitigating over-squashing without making over-smoothing worse.

Finding Memo(rization) in Graph Neural Networks

Adarsh Jamadandi*, Jing Xu, Adam Dziedzic and Franziska Boenisch.
Pre-print 2025 (Under-review).

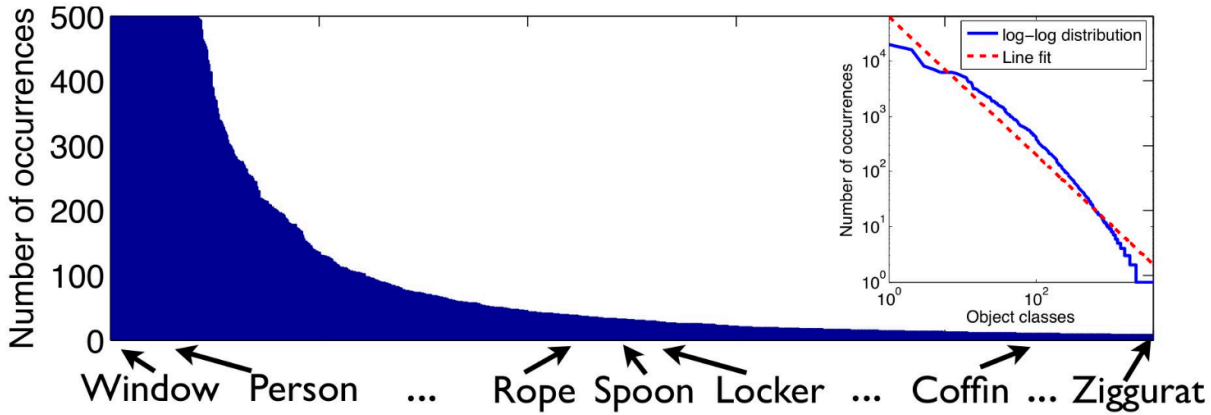
Deep Neural Networks Fit Random Labels



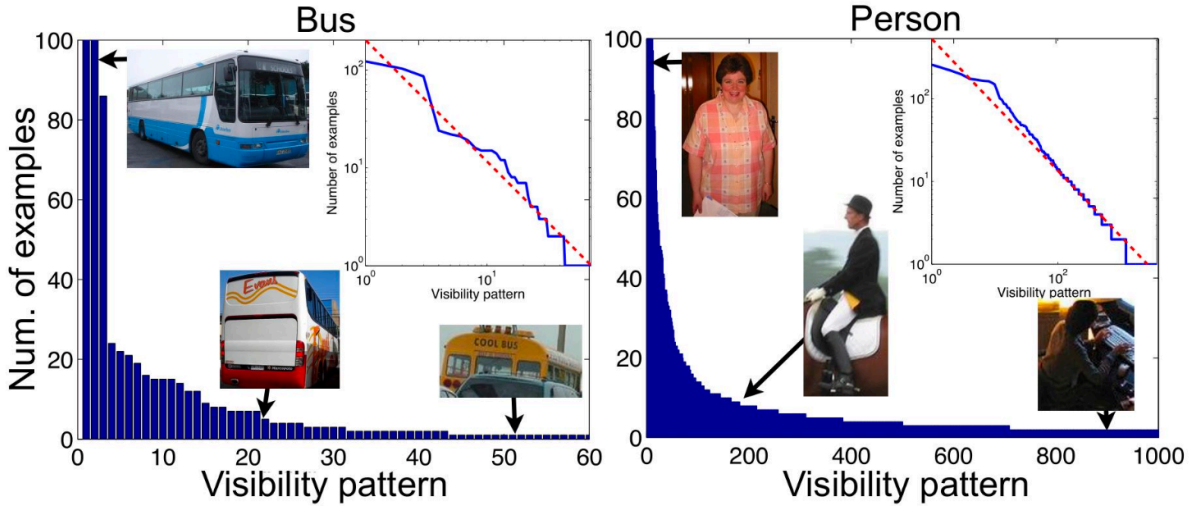
Credits: [Deep Learning requires rethinking generalization.]

Deep Neural Networks Memorize Atypical Examples

- Atypical data points in the form of mislabeled samples, visually difficult examples are more prone to memorization.
- Usually occur at the tail of the distribution.



(a) The number of examples by object class in SUN dataset

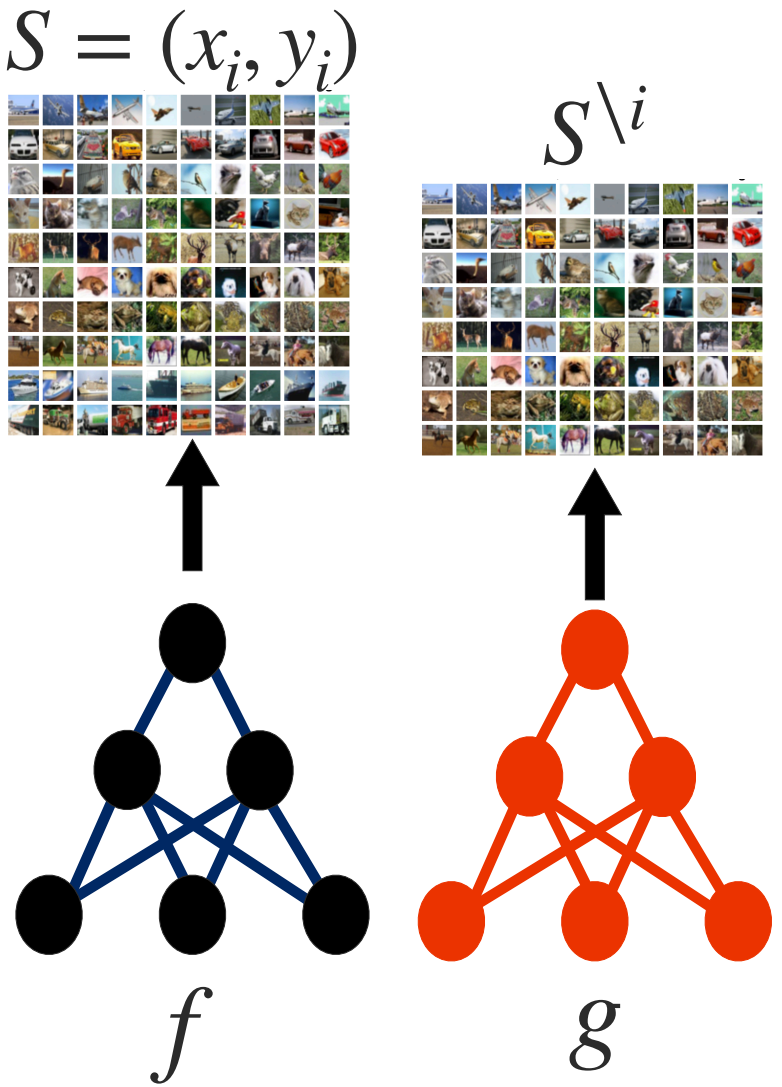


Credits: [Does Learning Require Memorization? A Short Tale about a Long Tail]

Definition of Memorization

- Train model f on S and train model g on $S^{\setminus i}$.
- Compare the behavior of the two models.
- Memorization score defined as

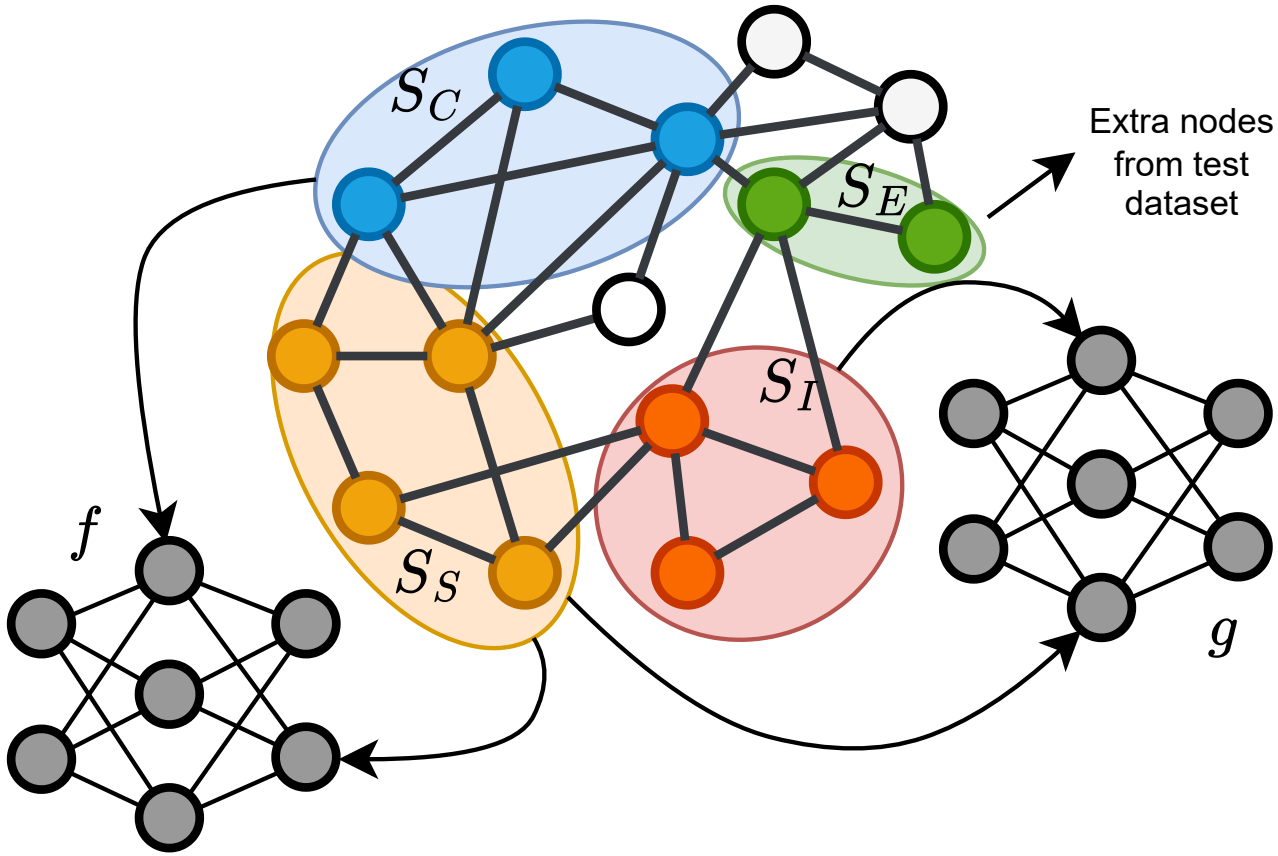
$$\mathscr{M}(x_i) = \mathop{\mathbb{E}}_{f \sim \mathscr{T}(S)} [\Pr[f(x_i) = y_i]] - \mathop{\mathbb{E}}_{g \sim \mathscr{T}(S \setminus x_i)} [\Pr[g(x_i) = y_i]]$$



Credits: Does Learning Require Memorization? A Short Tale about a Long Tail

Measuring Memorization in GNNs

- Problem setting: Semi-supervised node classification.
- Divide the nodes into Shared (S_S), Candidate (S_C), Independent (S_I) and Extra nodes (S_E).
- Train model $f = S_S \cup S_C$ and model $g = S_S \cup S_I$ respectively.



Does it work?

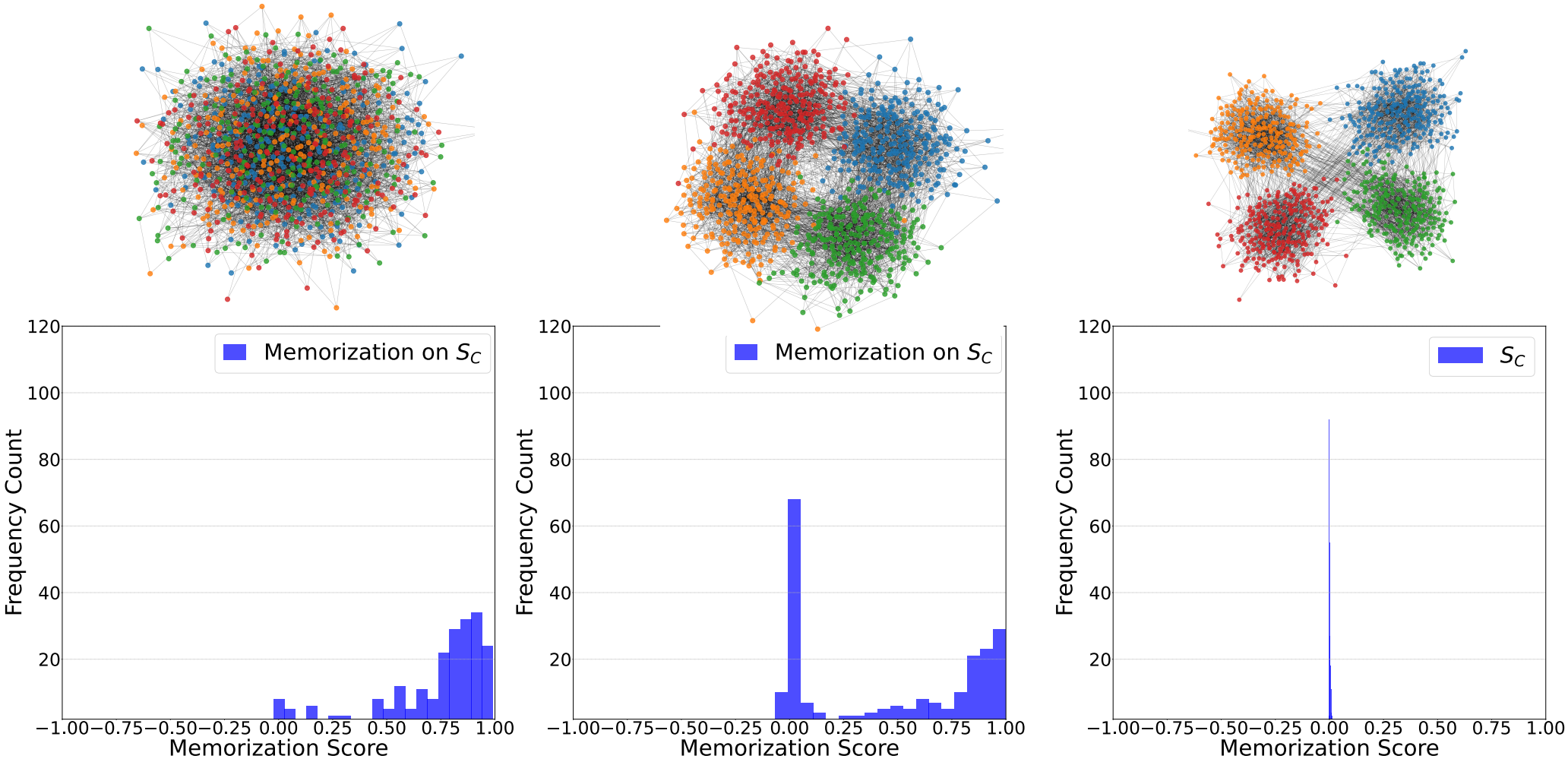
- Lets check if GNNs also show memorization by applying it to a toy dataset.
- We will measure memorization by comparing the behavior of models f and g by

$$\mathscr{M}(v_i) = \mathop{\mathbb{E}}_{f \sim \mathscr{T}(S)} [\Pr[f(v_i) = y_i]] - \mathop{\mathbb{E}}_{g \sim \mathscr{T}(S \setminus v_i)} [\Pr[g(v_i) = y_i]]$$

- A node is said to be memorized if $\mathscr{M}(v_i) > 0.5$ and we will measure memorization rate as

$$\text{MR}(\%) = \frac{1}{|\mathcal{S}|} \sum_{v_i \in S} \mathbb{I}(\mathscr{M}(v_i) > \tau) \times 100$$

Does it work?


Memorization on S_C
Frequency Count
Memorization Score
Memorization on S_C
Frequency Count
Memorization Score
S_C
Frequency Count
Memorization Score

TL;DR

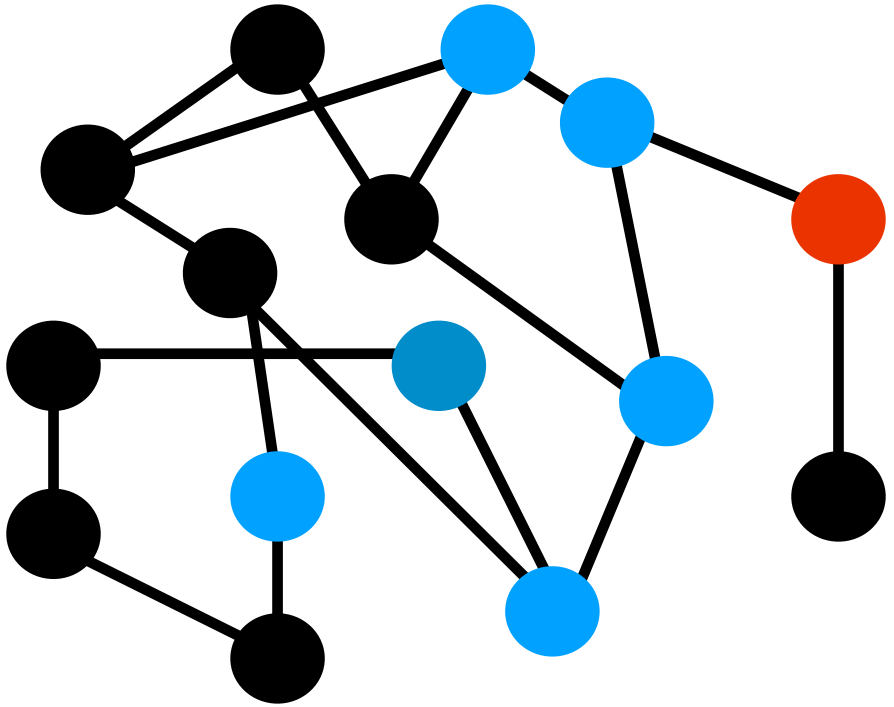
- **Deep neural networks can fit random labels → 100% Training Accuracy!**
- **Deep neural networks memorize atypical examples.**
- **GNNs also exhibit similar behavior → memorize node labels!**

Uncovering Mechanisms for Memorization

Why is it difficult to explain memorization in GNNs?

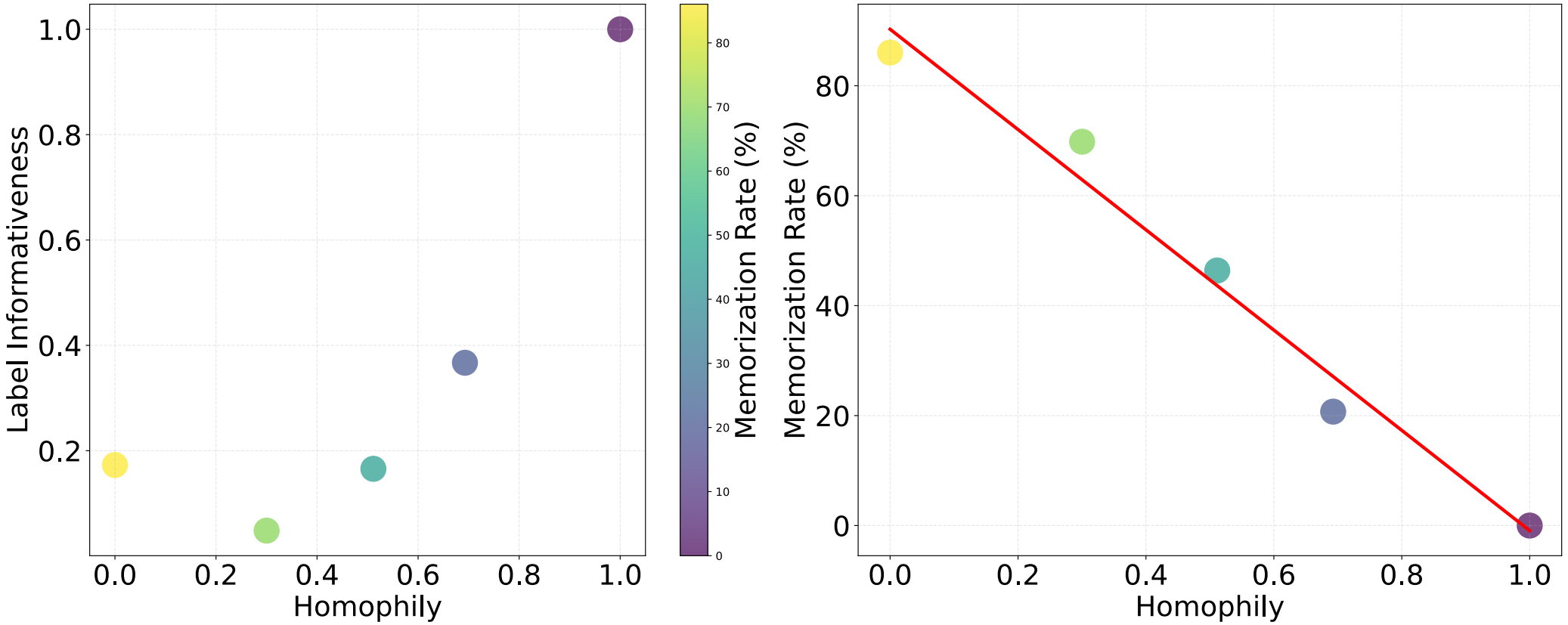
Explaining Memorization in GNNs

- Explaining memorization in GNNs is highly non-trivial.
- We need to account for various factors such as graph homophily, node features, label/feature distribution of neighbours, the GNN training dynamics!



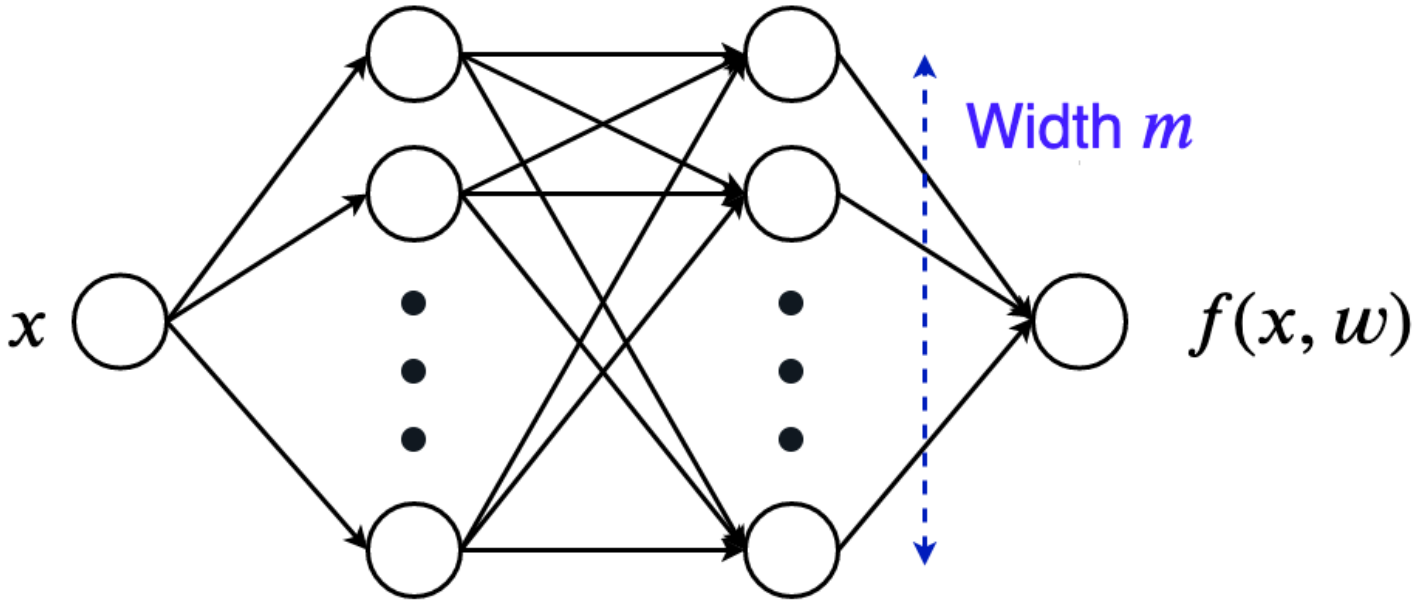
Homophily ↑ Memorization ↓

- As graph homophily increases, the rate of memorization decreases.



Detour: Neural Tangent Kernel

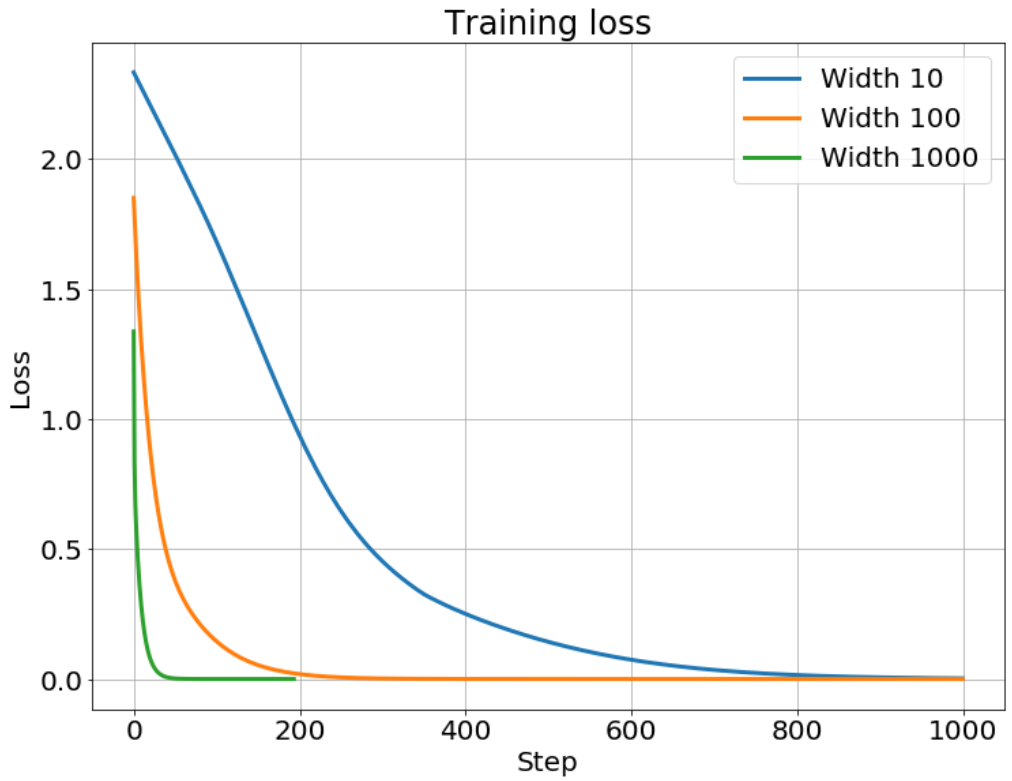
- Neural Tangent Kernel (NTK) - a theoretical tool to understand training dynamics of neural networks (NNs).
- Consider a NN with width m.

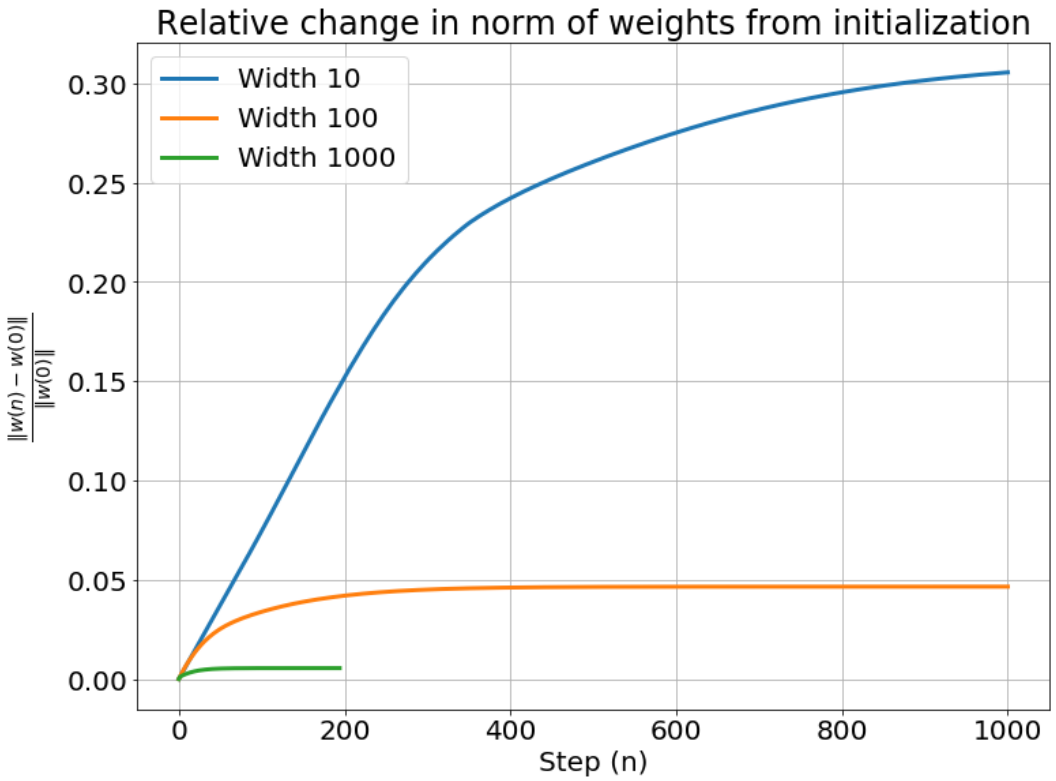


Credits: [Understanding the Neural Tangent Kernel]

Detour: Neural Tangent Kernel

- As $m \to \infty$, the weights barely change.
- The entire NN can be seen as a linear function!

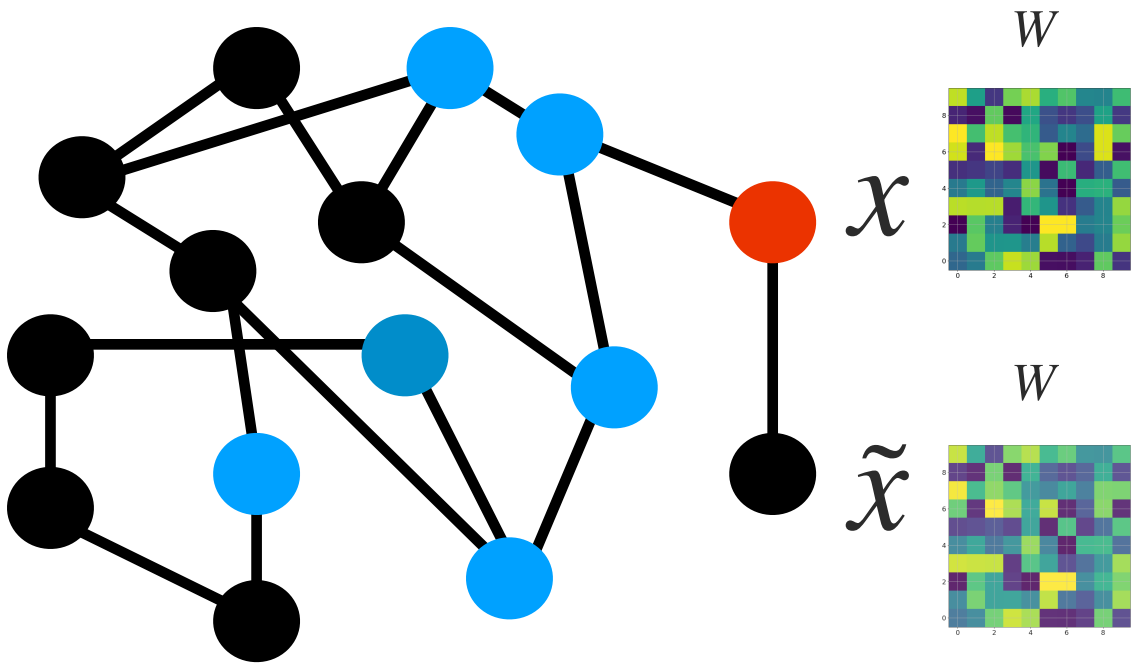




Credits: [Understanding the Neural Tangent Kernel]

Node Level Graph Neural Tangent Kernel

- GNTK - $\mathbf{\Theta}_t^l(x, \tilde{x}; \mathbf{A}) = \nabla_W f(x; \mathbf{A})^T \nabla_W f(\tilde{x}; \mathbf{A})$



Credits: [How Graph Neural Networks Learn: Lessons from Training Dynamics]

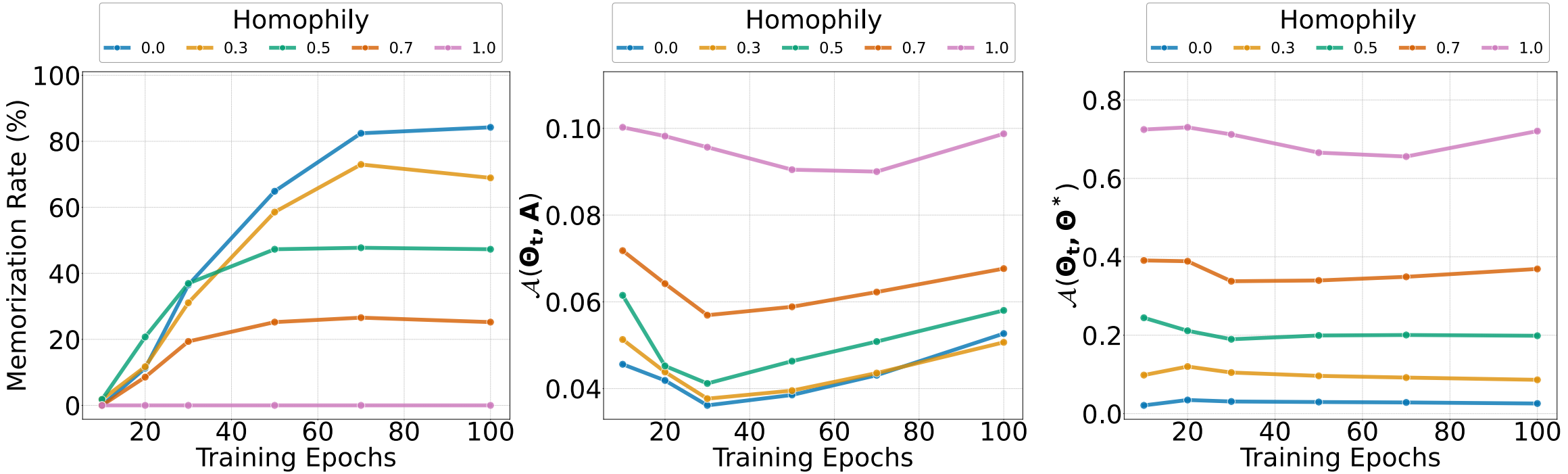
Graph Structural Bias and Kernel Alignment

- We will track three matrices -
- The alignment between the adjacency matrix $\mathbf{A}$ and the optimal kernel ($\mathbf{\Theta}^* = \bar{\mathbf{Y}}\bar{\mathbf{Y}}^T$) - $\mathscr{A}(\mathbf{A}, \mathbf{\Theta}^*)$ (homophily level)
- The Kernel-Target Alignment - $\mathscr{A}(\mathbf{\Theta_t}, \mathbf{\Theta}^*)$ (how well a GNN generalizes)
- The Kernel-Graph Alignment - $\mathscr{A}(\mathbf{\Theta_t}, \mathbf{A})$ (implicit bias of GNN to leverage the graph structure)

Credits: How Graph Neural Networks Learn: Lessons from Training Dynamics

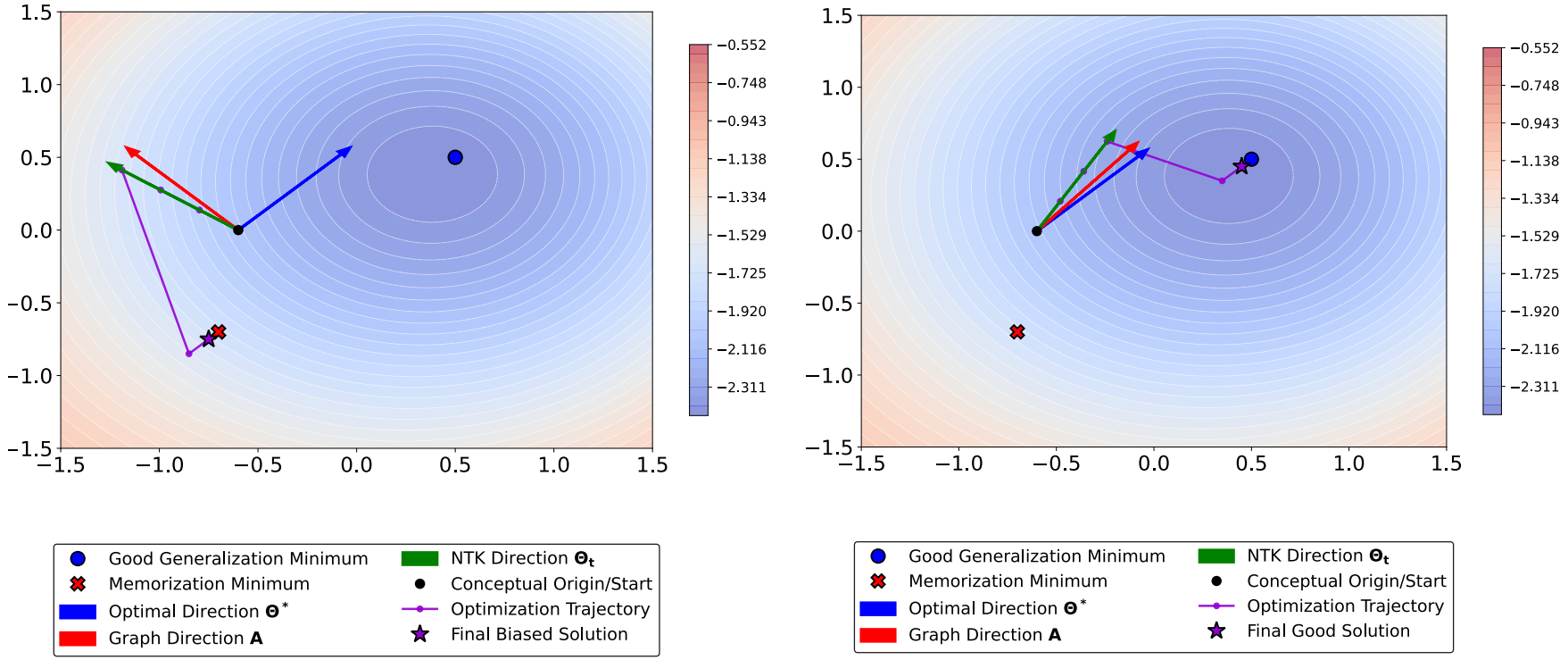
Graph Structural Bias and Kernel Alignment

- Low homophily → Memorization rate increases.
- Low homophily (graph structure is less informative) → $\mathscr{A}(\mathbf{\Theta_t}, \mathbf{A})$ improves.
- Low homophily → $\mathscr{A}(\mathbf{\Theta_t}, \mathbf{\Theta}^*)$ poor, suggests memorization.



Graph Structural Bias and Kernel Alignment

- In low-homophily settings, a GNN that aligns its NTK with the adjacency matrix cannot simultaneously align well with optimal kernel.
- Only way to achieve 0 training loss is via Memorization!

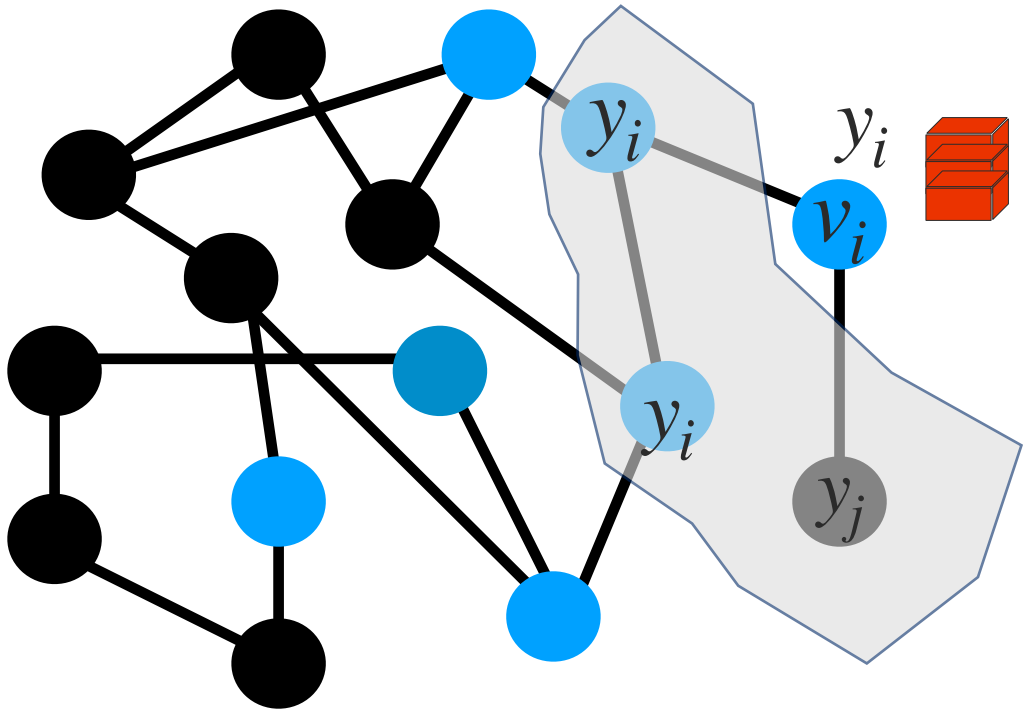


TL;DR

- Memorization rate $\propto \frac{1}{\text{Homophily}}$
- Low homophily → graph structure less informative → GNNs still use it!
- Only way to achieve 0 training loss → Memorize node labels!

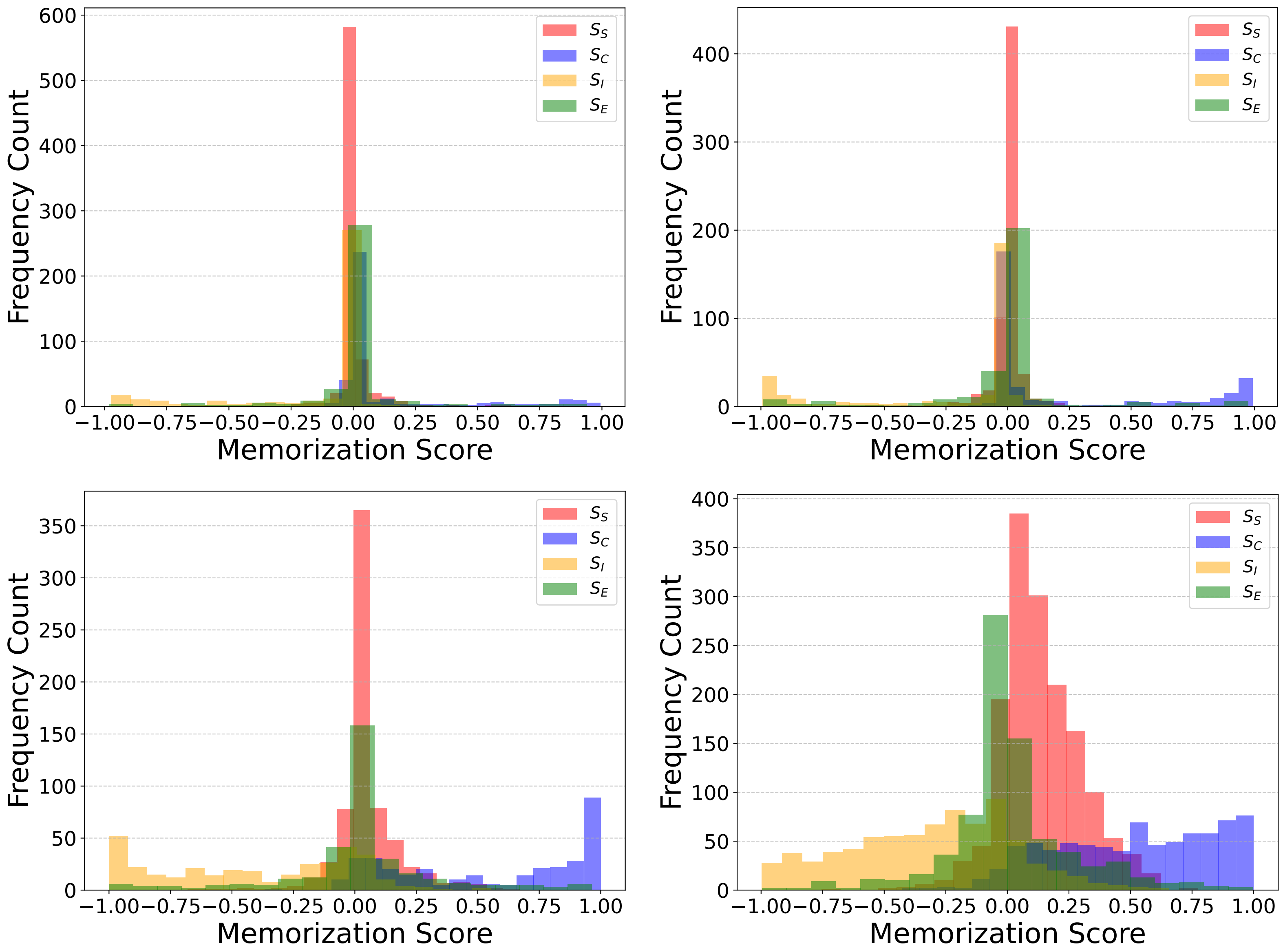
Node Atypicality

- We propose a novel Label Disagreement Score (LDS) that measures local structural anomaly in the feature space of the nodes.
- We find nodes with high LDS are the ones that get memorized.



$$\mathrm{LDS}_k(v_i) = \frac{1}{k} \sum_{v_j \in N_k(v_i)} \mathbb{I}[y_j \neq y_i]$$

Memorization Scores on Real World Graphs



LDS on Real-World Graphs

Dataset	MemNodes LDS	Non-MemNodes LDS	p-value (< 0.01)	Effect Size
Cora	0.8016±0.0186	0.5754±0.0010	0.0024	14.39
Citeseer	0.7985±0.0114	0.6192±0.0040	0.0029	13.13
Chameleon	0.8433±0.0006	0.7362±0.0016	0.0000	113.68
Squirrel	0.8617±0.0032	0.7675±0.0040	0.0020	15.98

Summary

- GNNs also memorize node labels.
- As the homophily level of graph increases, the memorization substantially decreases.
- In low-homophily settings, the graph is unhelpful for the task. But GNNs have an implicit bias to use the graph structure despite that.
- How to achieve 0 train loss? Memorize!
- Nodes with high label disagreement score usually get memorized.