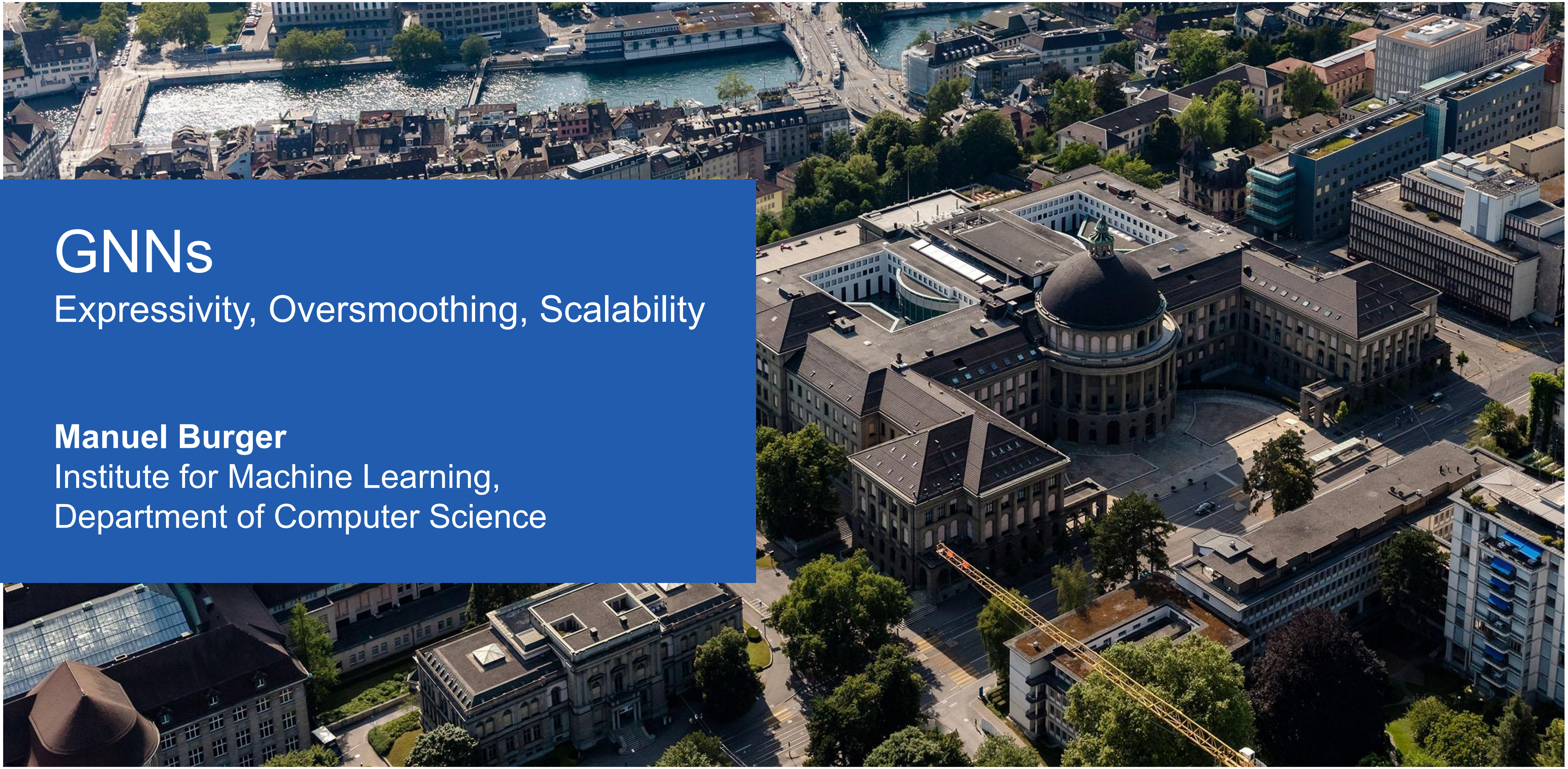


GNNs

Expressivity, Oversmoothing, Scalability

Manuel Burger
Institute for Machine Learning,
Department of Computer Science

Contents

- Expressivity:
  - *Weisfeiler-Lehman Test,*
  - *Graph Isomorphism Operator*
- Oversmoothing
  - *Dirichlet Energy*
- Scalability by Subsampling
  - *Node-wise*
  - *Graph-wise*

Expressivity

Lecture 03: Neural Message Passing

1. **AGGREGATE** function takes as input the set of embeddings of the nodes in v's graph neighborhood $\mathcal{N}(v)$ and generates the message based on this aggregated information.

$$h_v^0 = x_v, \forall v \in V$$

For each step (layer) k = *1, ...,* K:

$$\boxed{m_{\mathcal{N}(v)}^{k-1} = \text{AGGREGATE}\left(\{h_u^{k-1}, \forall u \in \mathcal{N}(v)\}\right)}$$

2. **UPDATE** function combines this message with previous node's v embedding to create the new embedding (k = *1, ...,* K):

$$h_v^k = \text{UPDATE}(h_v^{k-1}, m_{\mathcal{N}(v)}^{k-1})$$

Graph Sample and Aggregate (GraphSAGE)

$$h_v^k = f^k\left(W^k\left[\mathrm{AGG}_{u\in\mathcal{N}(v)}(\{h_u^{k-1}\}),\, h_v^{k-1}\right]\right),\ \forall v \in V$$

AGG choice:

- Mean (similar to the GCN);
- Pool**:** transform neighbor vectors and apply Mean(·) or Max(·):

$$\max\left(\{\mathrm{MLP}(\, h_u^{k-1}),\ \forall u \in \mathcal{N}(v)\}\right)$$

- LSTM (after ordering the sequence of neighbours).

But what can we actually learn with these operators?

The Weisfeiler-Lehman Test

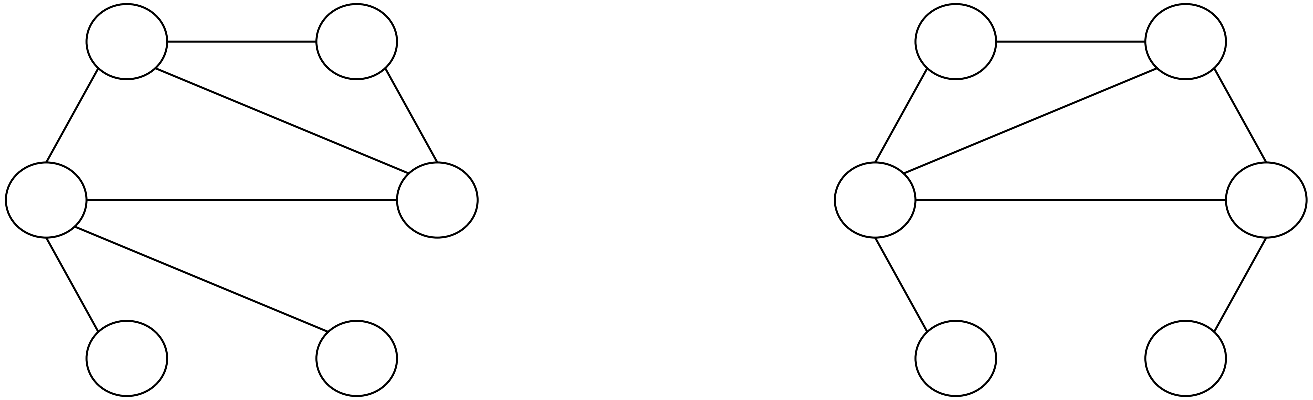
A Reduction of a Graph to a Canonical Form and an Algebra Arising during This Reduction, Weisfeiler and Lehman, 1968

- Problem: Given two graphs, decide whether they are ***isomorphic*** (topologically identical).
- *1-dimensional* version of the *WL-algorithm*:
  - *Color Refinement Algorithm*, *Naive Vertex Classification*
- Algorithm for graph $\boldsymbol{G}$ with nodes $v \in V$:
  - Initial color for each node: $c^{(0)}(v)$
  - Iterative update: $c^{(k+1)}(v) = \mathrm{HASH}\left(c^{(k)}(v), \left\{c^{(k)}(v)\right\}_{u \in N(v)}\right)$
  - $\mathbf{HASH}$ maps inputs to diff. colors
  - After K steps of color refinement $c^{(K)}(v)$ summarizes K-*hop neighborhood*

{} for **multi-sets**

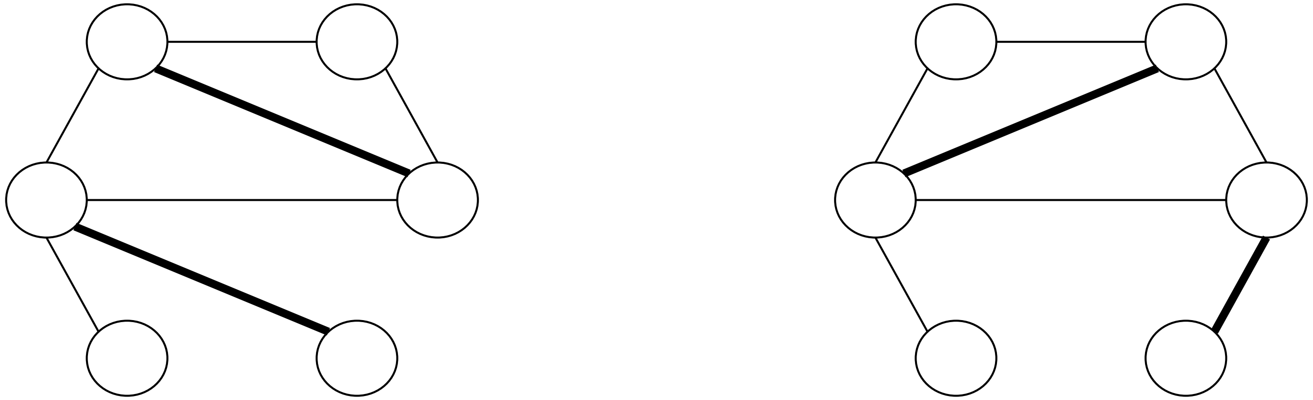
The Weisfeiler-Lehman Test – Example

- Problem: Given two graphs, decide whether they are ***isomorphic*** (topologically identical).



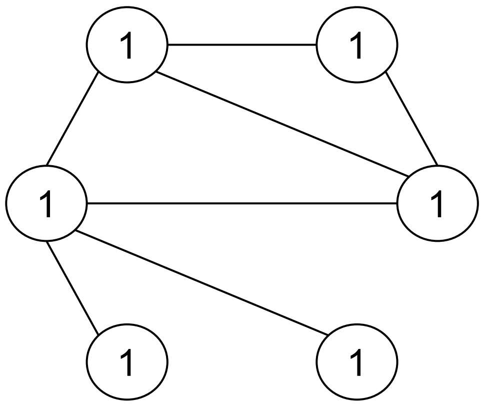
The Weisfeiler-Lehman Test – Example

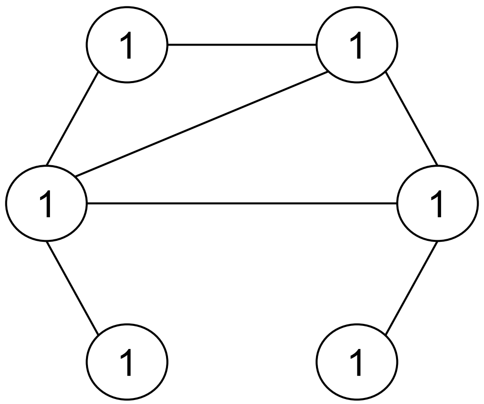
- Problem: Given two graphs, decide whether they are ***isomorphic*** (topologically identical).



The Weisfeiler-Lehman Test – Example

Assign Initial Colors

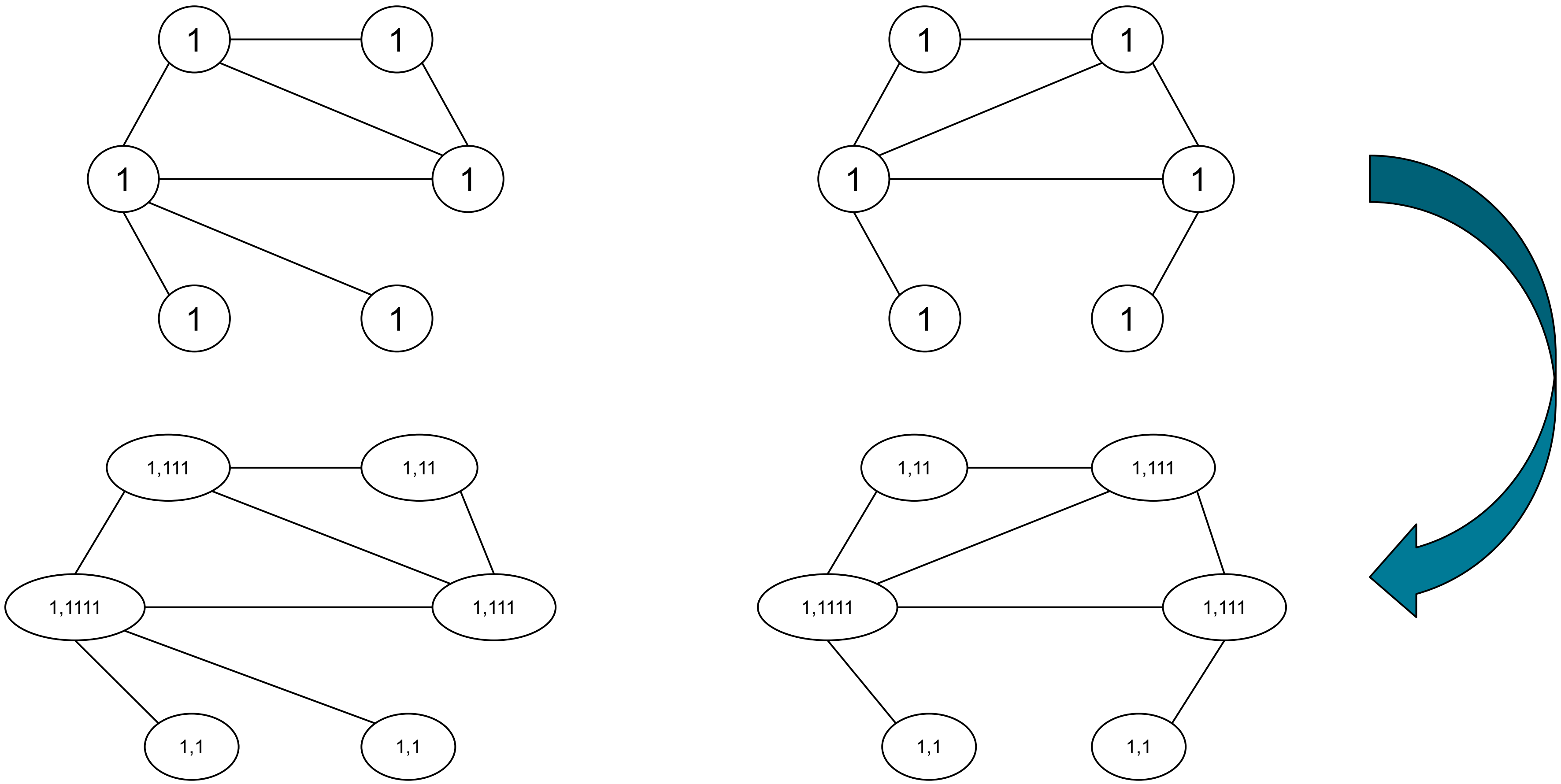




Hash Table

The Weisfeiler-Lehman Test – Example

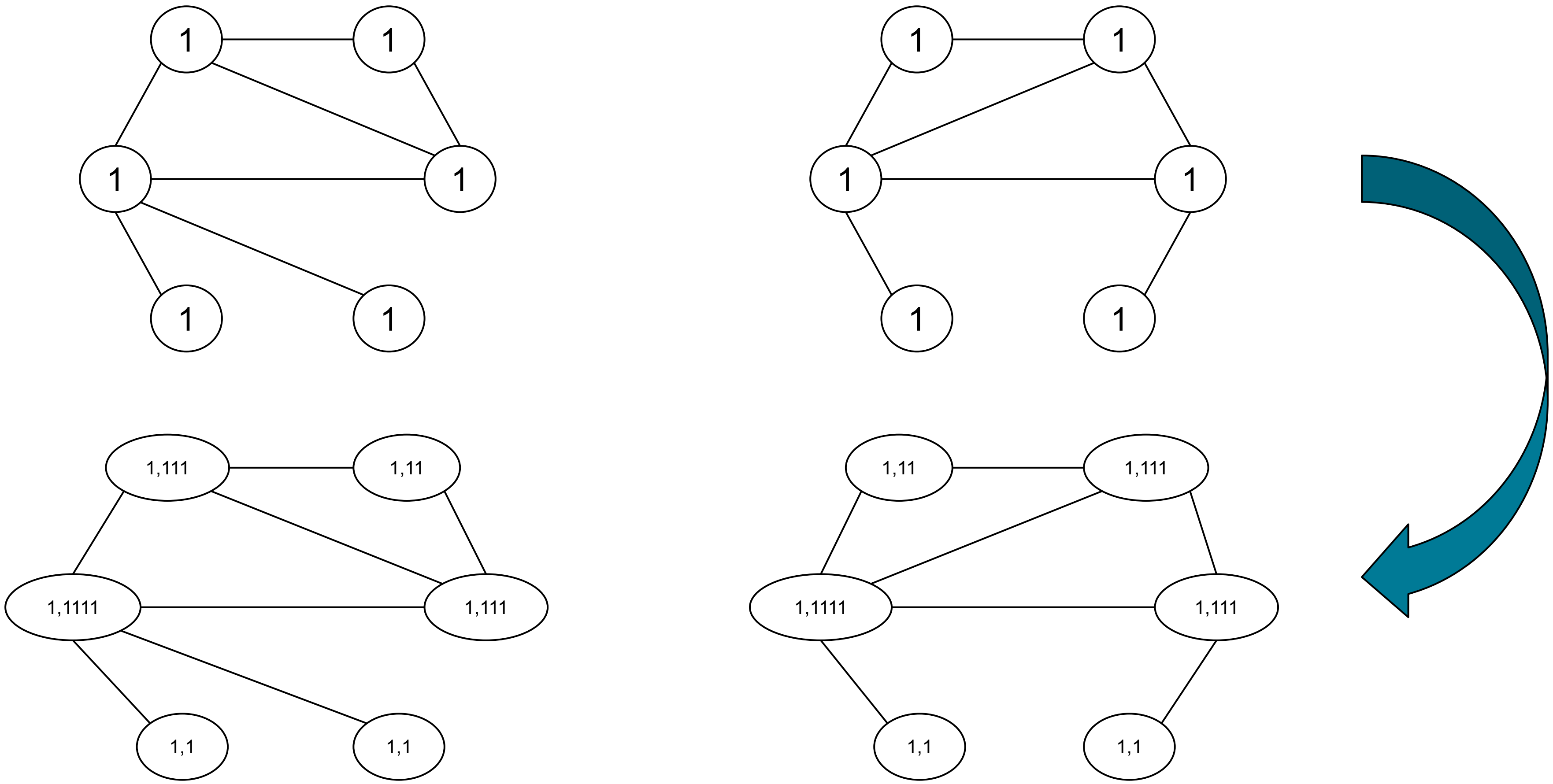
Assign initial colors and aggregate



Hash Table

The Weisfeiler-Lehman Test – Example

Compute hashes

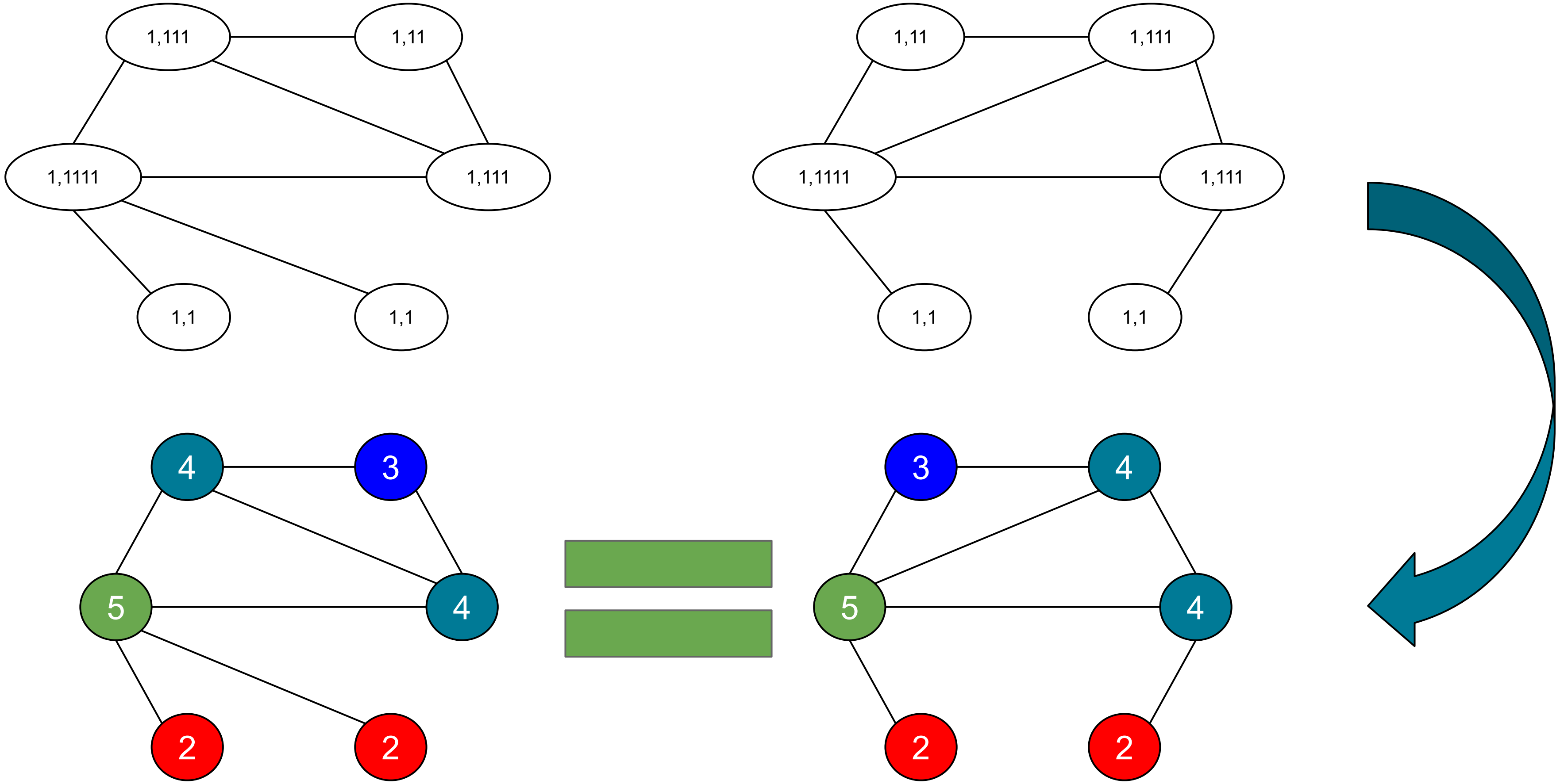


Hash Table

1,1	**2**
1,11	**3**
1,111	**4**
1,1111	**5**

The Weisfeiler-Lehman Test – Example

Refine coloring

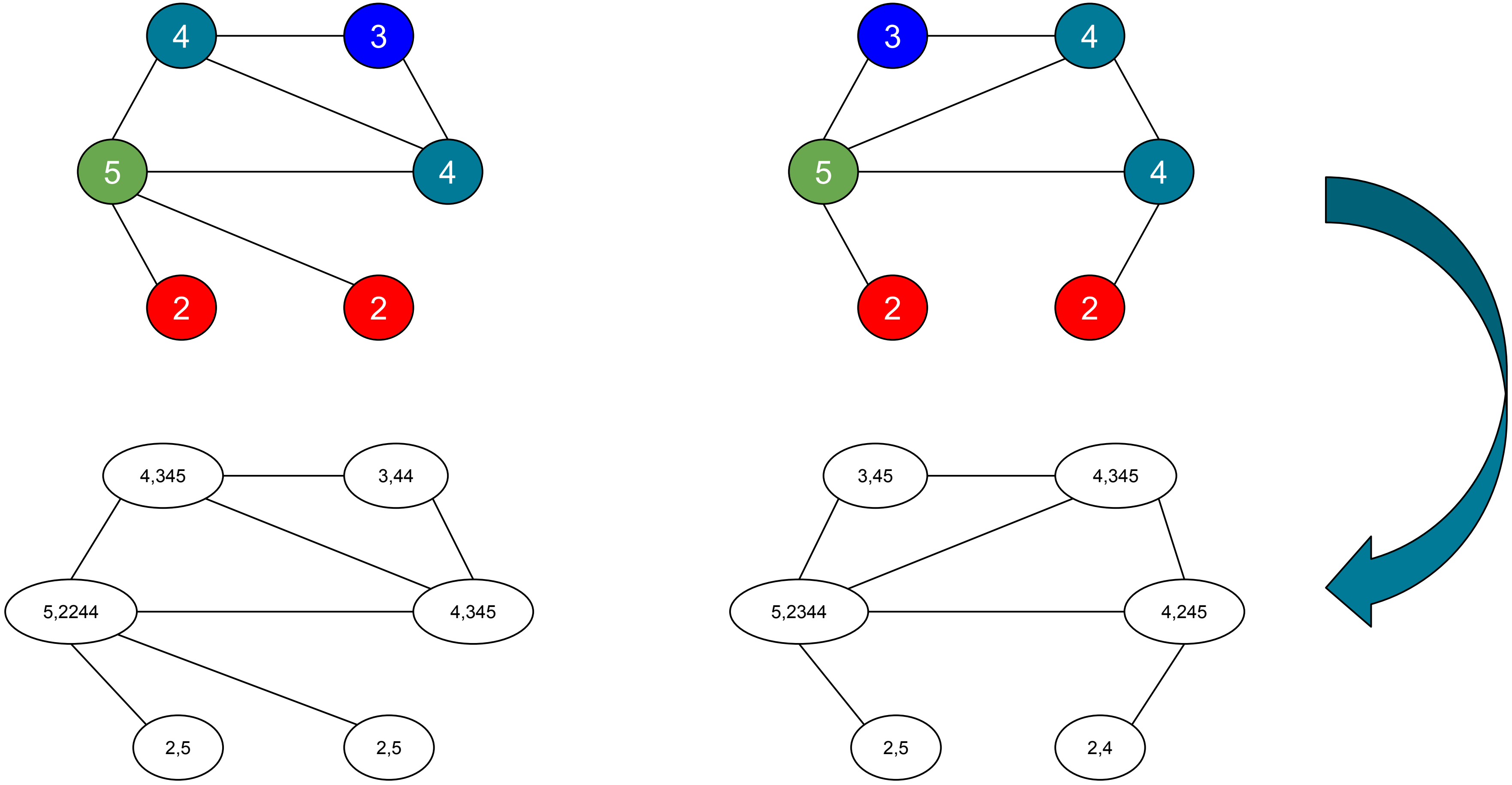


Hash Table

1,1	2
1,11	3
1,111	4
1,1111	5

The Weisfeiler-Lehman Test – Example

Aggregate neighbours

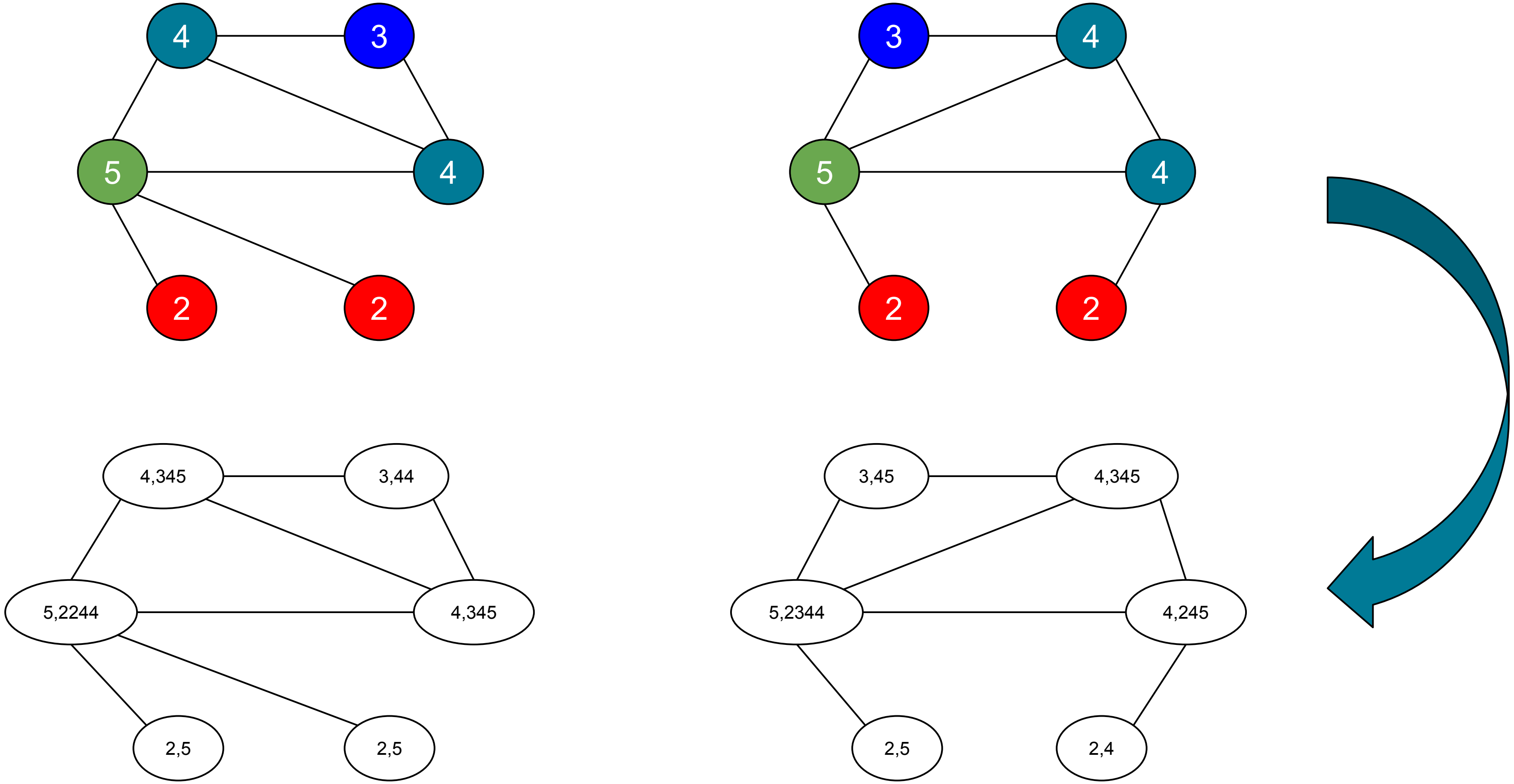


Hash Table

1,1	**2**
1,11	**3**
1,111	**4**
1,1111	**5**

The Weisfeiler-Lehman Test – Example

Compute hashes

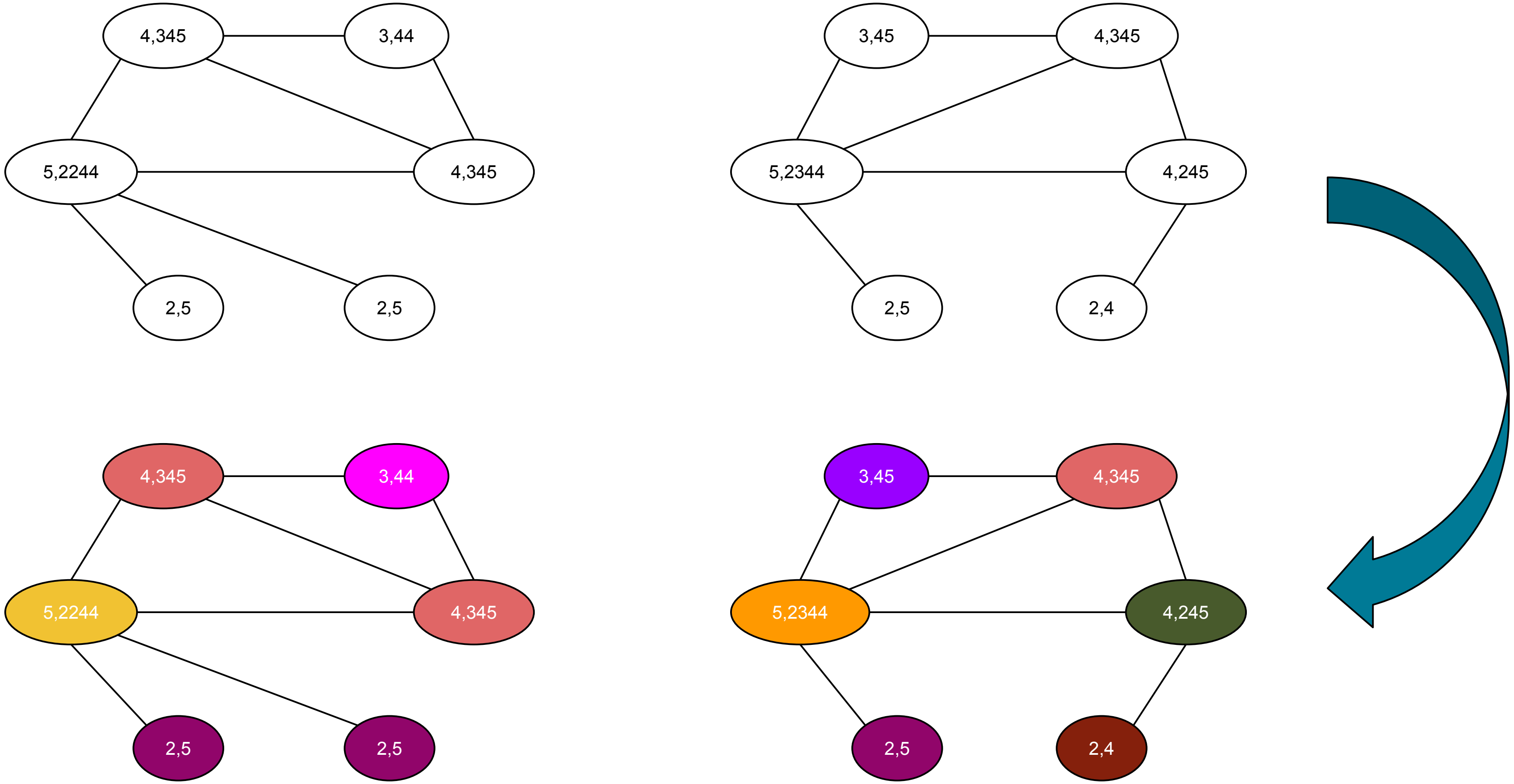


Hash Table

1,1	2
1,11	3
1,111	4
1,1111	5
4,345	6
3,44	7
5,2244	8
2,5	9
3,45	10
4,245	11
5,2344	12
2,4	13

The Weisfeiler-Lehman Test – Example

Refine coloring

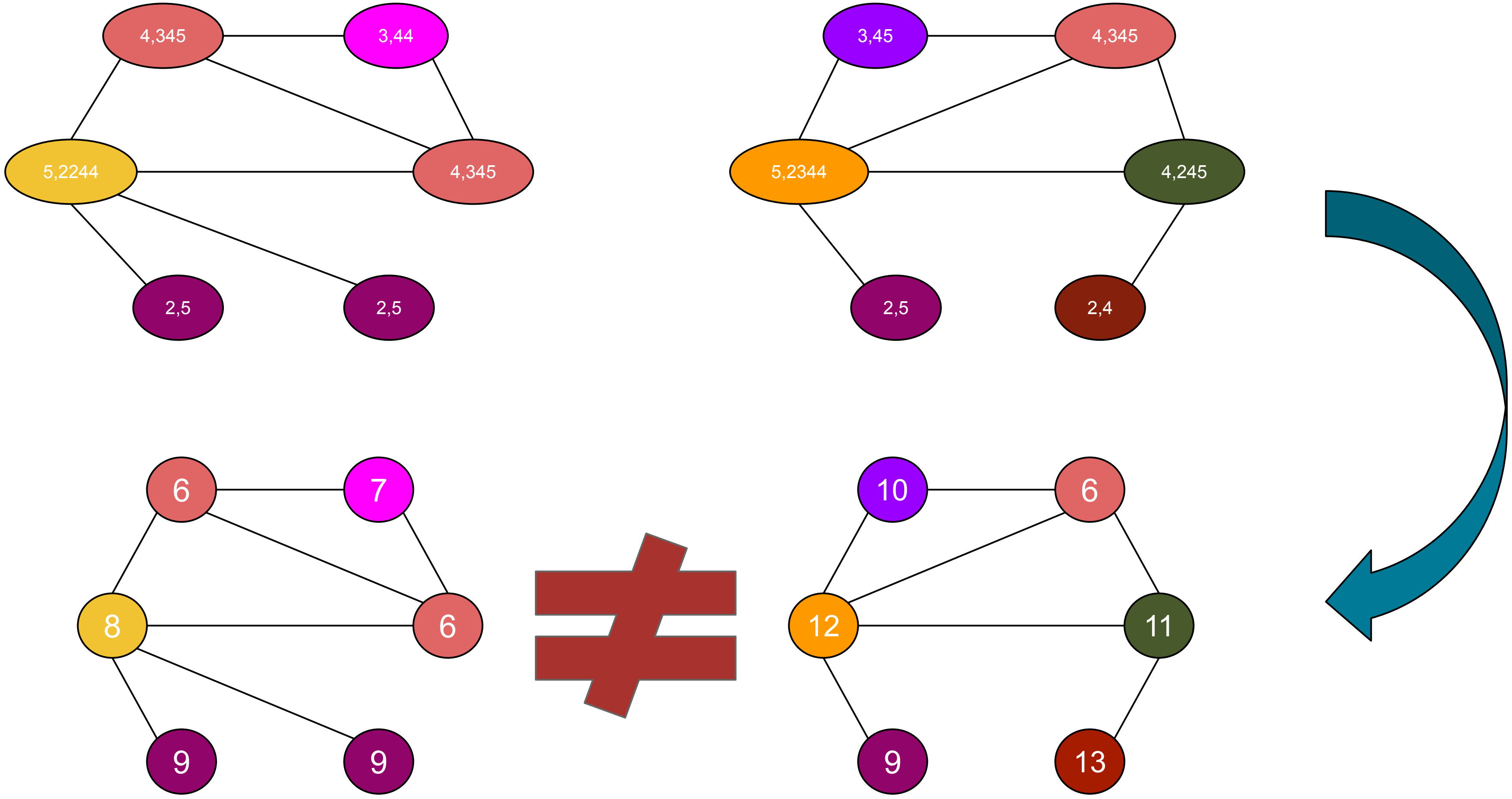


Hash Table

1,1	2
1,11	3
1,111	4
1,1111	5
4,345	6
3,44	7
5,2244	8
2,5	9
3,45	10
4,245	11
5,2344	12
2,4	13

The Weisfeiler-Lehman Test – Example

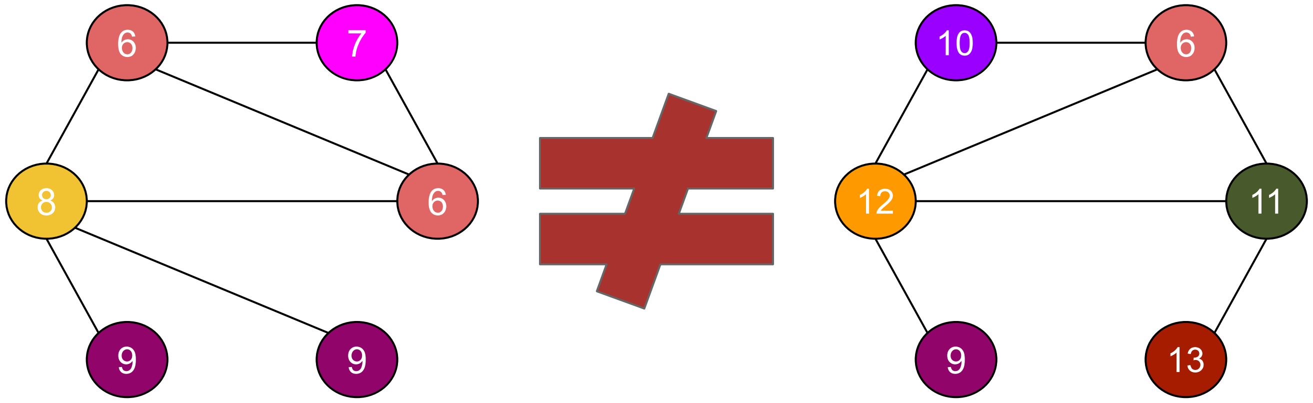
Refine coloring



Hash Table

1,1	2
1,11	3
1,111	4
1,1111	5
4,345	6
3,44	7
5,2244	8
2,5	9
3,45	10
4,245	11
5,2344	12
2,4	13

The Weisfeiler-Lehman Test – Example



Careful!

The result is only guaranteed to be correct if the test fails. If the coloring matches, we do not have a guarantee for isomorphism

Hash Table

1,1	2
1,11	3
1,111	4
1,1111	5
4,345	6
3,44	7
5,2244	8
2,5	9
3,45	10
4,245	11
5,2344	12
2,4	13

Can we use these ideas to analyse and improve our message passing operators?

These two look suspiciously similar?!

MPNNs

$$m_{\mathcal{N}(v)}^{k-1} = \text{AGGREGATE}\left(\{h_u^{k-1}, \forall u \in \mathcal{N}(v)\}\right)$$

$$h_v^k = \text{UPDATE}(h_v^{k-1}, m_{\mathcal{N}(v)}^{k-1})$$

1-WL

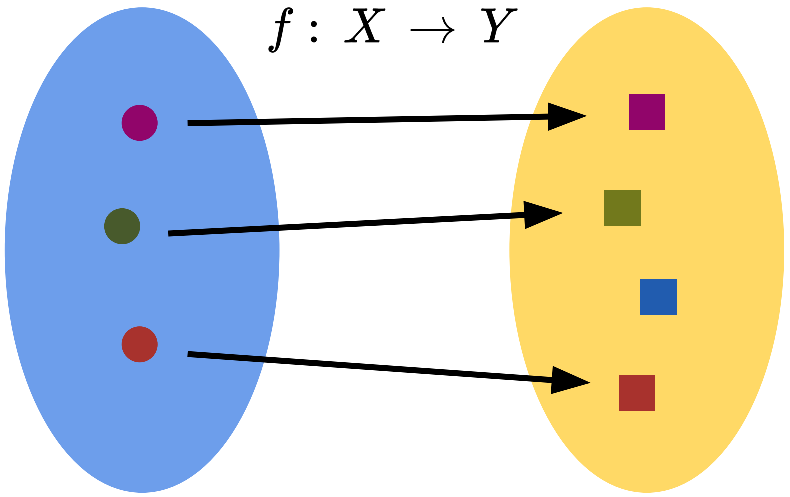
$$c^{(k+1)}(v) = \text{HASH}\left(c^{(k)}(v), \left\{c^{(k)}(v)\right\}_{u \in N(v)}\right)$$

Neighbour Aggregation

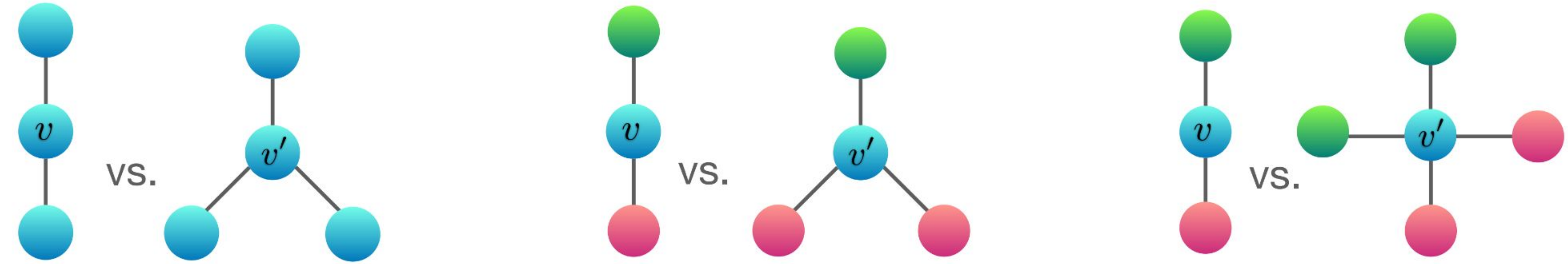
- MPNNs perform *neighborhood aggregation*
- 1-WL shows us:
  - *Neighborhood aggregation* can be abstracted as a function over a multi-set
- **HASH** is an *injective function*
  - A *injective function* $f : X \to Y$ maps different inputs to different outputs
  - Intuition: We retain all information from our source in the target space. Target space is at least as “big” as source space

$$m_{\mathcal{N}(v)}^{k-1} = \text{AGGREGATE}\left(\{h_u^{k-1}, \forall u \in \mathcal{N}(v)\}\right)$$

$$c^{(k+1)}(v) = \text{HASH}\left(c^{(k)}(v), \left\{c^{(k)}(v)\right\}_{u \in N(v)}\right)$$



Neighbour Aggregation



(a) Mean and Max both fail

(b) Max fails

(c) Mean and Max both fail

$$m_{\mathcal{N}(v)}^{k-1} = \text{AGGREGATE}\left(\{h_u^{k-1}, \forall u \in \mathcal{N}(v)\}\right)$$

- Common aggregation functions like *mean* and *max* fail to distinguish structures, *not injective*
- ***SUM*** aggregations preserve input information

Xu et al. conclude this by looking at multi-set aggregations

Update Function

- The *update* function transforms our aggregated information
- We use a *Multi-Layer Perceptron (MLP)*
- Hornik et al., 1989, Universal Approximation Theorem:
  - *"1-hidden-layer MLP with sufficiently-large hidden dimensionality and appropriate non-linearity (including ReLU and sigmoid) can approximate any continuous function to an arbitrary accuracy."*
- Xu et al., 2018 propose:
  - *GIN: Graph Isomorphism Operator*

$$h_v^k = \mathrm{UPDATE}(h_v^{k-1}, m_{\mathcal{N}(v)}^{k-1})$$

$$MLP(x) = W_1\sigma(W_2 x)$$

$$c^{(k+1)}(v) = MLP_\theta\left((1+\epsilon)\cdot MLP_\psi\left(c^{(k)}(v)\right) + \sum_{u\in N(v)} MLP_\psi\left(c^{(k)}(u)\right)\right)$$

Multilayer feedforward networks are universal approximators, Hornik et al., 1989

GIN: Graph Isomorphism Operator

How Powerful are Graph Neural Networks?, Xu et al., 2018

- Xu et al., 2018 propose:
  - *GIN: Graph Isomorphism Operator*
- Theorem, Xu et al., 2018: *GIN's neighborhood aggregation functions is injective.*
- *GIN* is the most *expressive* MPNN of the introduced operators

GIN

$$c^{(k+1)}(v) = MLP_\theta\left((1+\epsilon)\cdot MLP_\psi\left(c^{(k)}(v)\right) + \sum_{u\in N(v)} MLP_\psi\left(c^{(k)}(u)\right)\right)$$

1-WL

$$c^{(k+1)}(v) = \mathrm{HASH}\left(c^{(k)}(v), \left\{c^{(k)}(v)\right\}_{u\in N(v)}\right)$$

Multilayer feedforward networks are universal approximators, Hornik et al., 1989

Notebook: `wl-algorithm.ipynb`

Oversmoothing

Oversmoothing

- *"We define over-smoothing [...] as the layer-wise exponential convergence of the node-similarity measure to zero…" - Rusch et al., 2023*

Definition 1 (Over-smoothing). *Let $\mathcal{G}$ be an undirected, connected graph and $\mathbf{X}^n \in \mathbb{R}^{v \times m}$ denote the n-th layer hidden features of an N-layer GNN defined on $\mathcal{G}$. Moreover, we call $\mu : \mathbb{R}^{v \times m} \longrightarrow \mathbb{R}_{\geq 0}$ a* ***node-similarity measure*** *if it satisfies the following axioms:*

1. $\exists \mathbf{c} \in \mathbb{R}^m$ *with* $\mathbf{X}_i = \mathbf{c}$ *for all nodes* $i \in \mathcal{V} \Leftrightarrow \mu(\mathbf{X}) = 0,$ *for* $\mathbf{X} \in \mathbb{R}^{v \times m}$
2. $\mu(\mathbf{X} + \mathbf{Y}) \leq \mu(\mathbf{X}) + \mu(\mathbf{Y}),$ *for all* $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{v \times m}$

We then define **over-smoothing with respect to** μ *as the layer-wise exponential convergence of the node-similarity measure μ to zero, i.e.,*

3. $\mu(\mathbf{X}^n) \leq C_1 e^{-C_2 n}$, *for* $n = 0, \ldots, N$ *with some constants* $C_1, C_2 > 0.$

Latent Dirichlet Energy - Quantify Oversmoothing

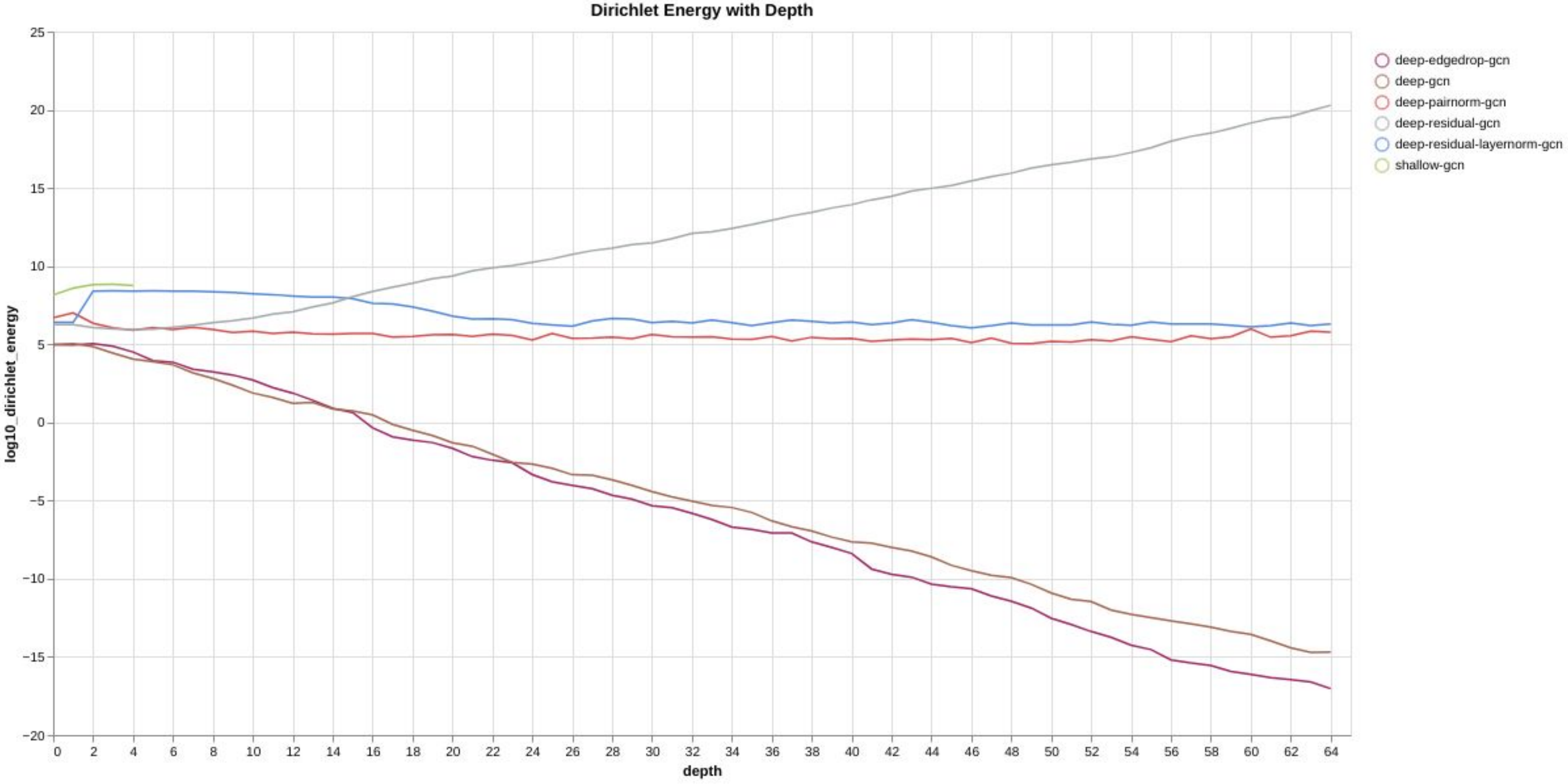
- *"We define over-smoothing [...] as the layer-wise exponential convergence of the node-similarity measure to zero…" - Rusch et al., 2023*
- *Dirichlet Energy* of node features at layer *n:* $X^n \in \mathbb{R}^{|V| \times m}$

$$\mathcal{E}\left(X^n\right) = \frac{1}{|V|} \sum\nolimits_{i \in \mathcal{V}} \sum\nolimits_{j \in \mathcal{N}_i} \left|\left|X_i^n - X_j^n\right|\right|_2^2$$

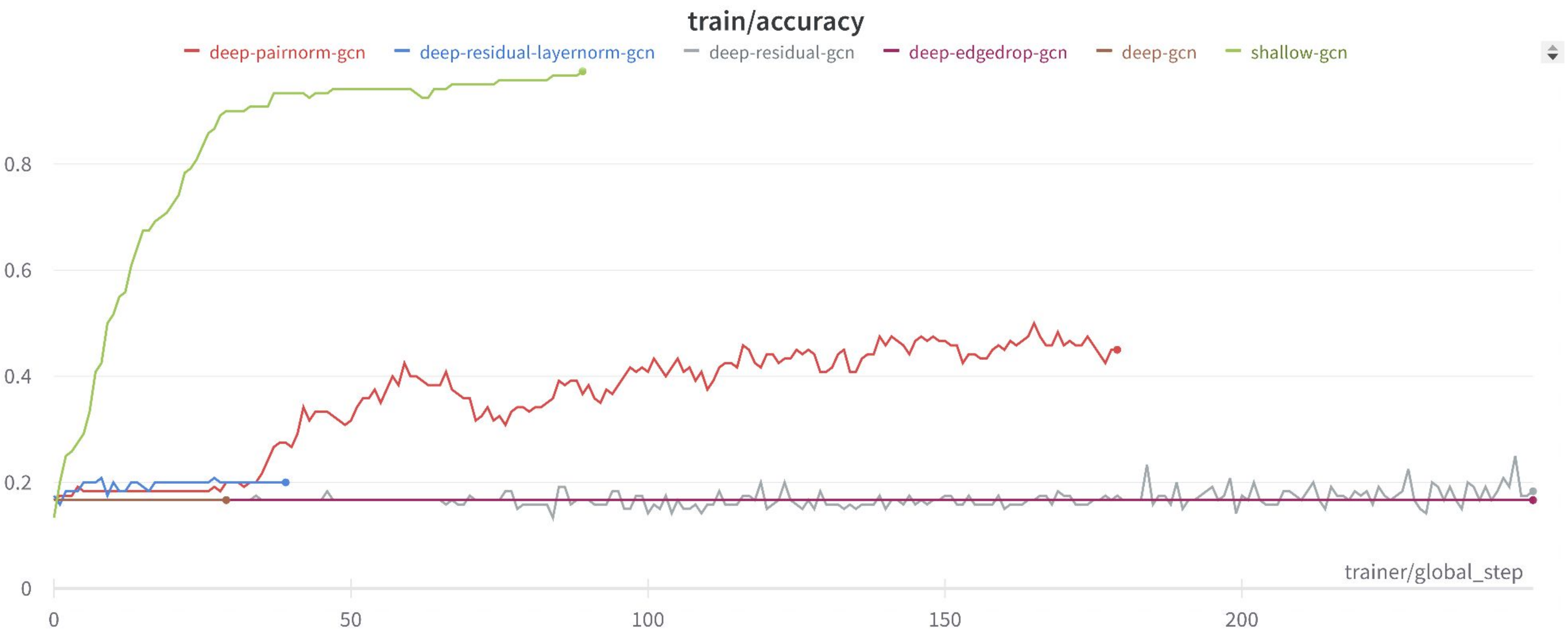
- Then the following node-similarity satisfies the previous definition: $\mu(X^n) = \sqrt{\mathcal{E}\left(X^n\right)}$

Notebook: `oversmoothing.ipynb`

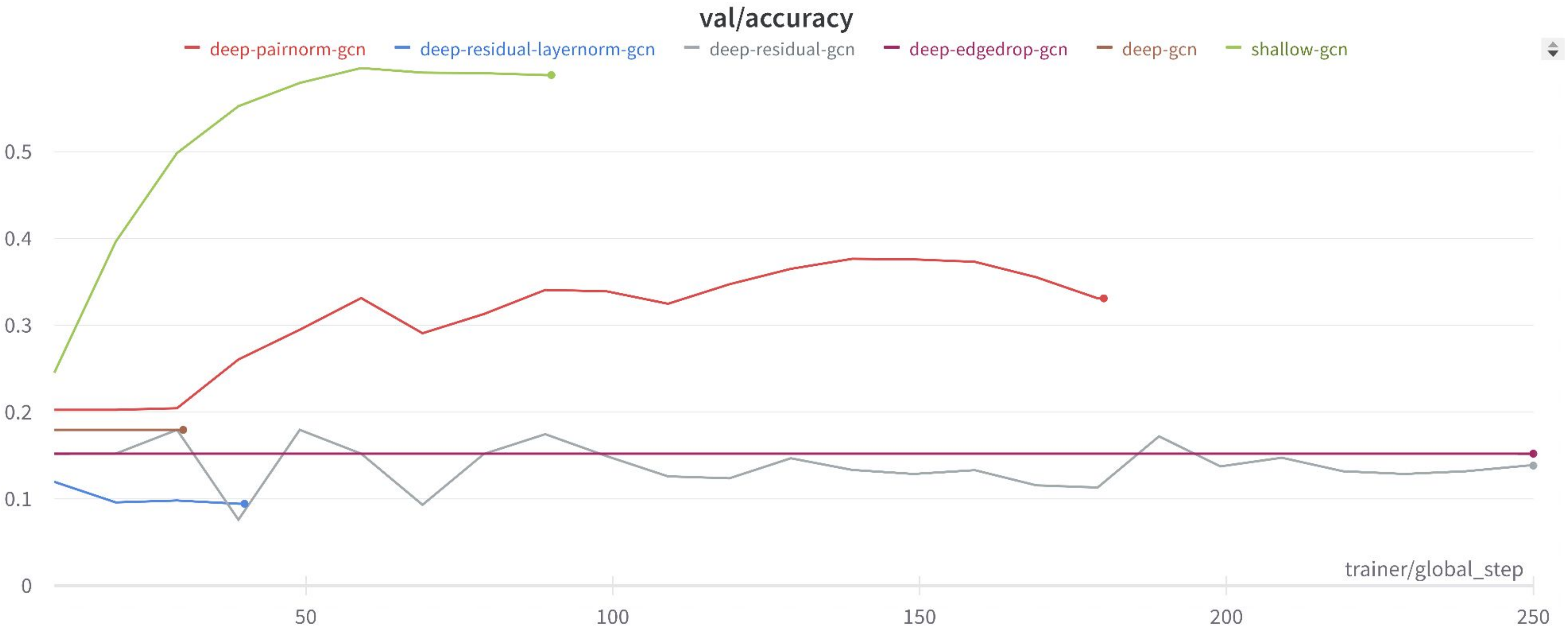
Experiments – CiteSeer



Experiments – CiteSeer



Experiments – CiteSeer



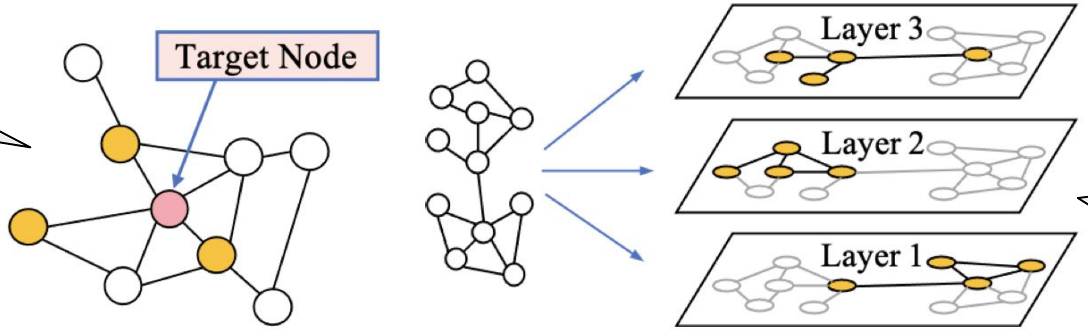
Oversmoothing – Insights

- Regularization techniques can help:
  - DropEdge [Rong et al., 2019], PairNorm [Zhao et al.]
- Residual connections (with appropriate normalization)
- “A Survey on Oversmoothing in Graph Neural Networks”, Rusch et al., 2023:
  - *“It turns out that simply adding a bias vector to a deep GCN with shared parameters among layers [...] with weights W and bias b, is sufficient for the optimizer to keep the resulting layer-wise Dirichlet energy of the model approximately constant”*
- “Dirichlet Energy Constrained Learning for Deep Graph Neural Networks”, Zhou et al., 2021:
  - Constrain the dirichlet energy in each layer (lower and upper bound)
  - Customize: Initialization, add regularization terms, activation function, residual connections

Scalability

Subsampling and Partitioning Graphs

We have seen this: *GraphSAGE* by Hamilton et al.

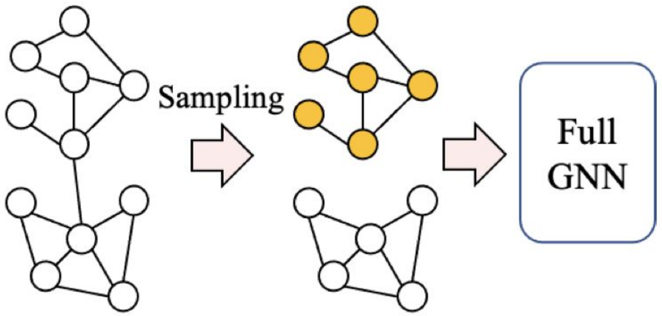


(a) Node-wise. (b) Layer-wise.

Given a node, show different views at different depths based on sampling methods:

- *FastGCN: Chen et al., 2018*
- *ASGCN: Huang et al., 2018*

Explicit Partitioning as a preprocessing step



(c) Graph-wise.

Subsampling and Partitioning Graphs

Node-wise:

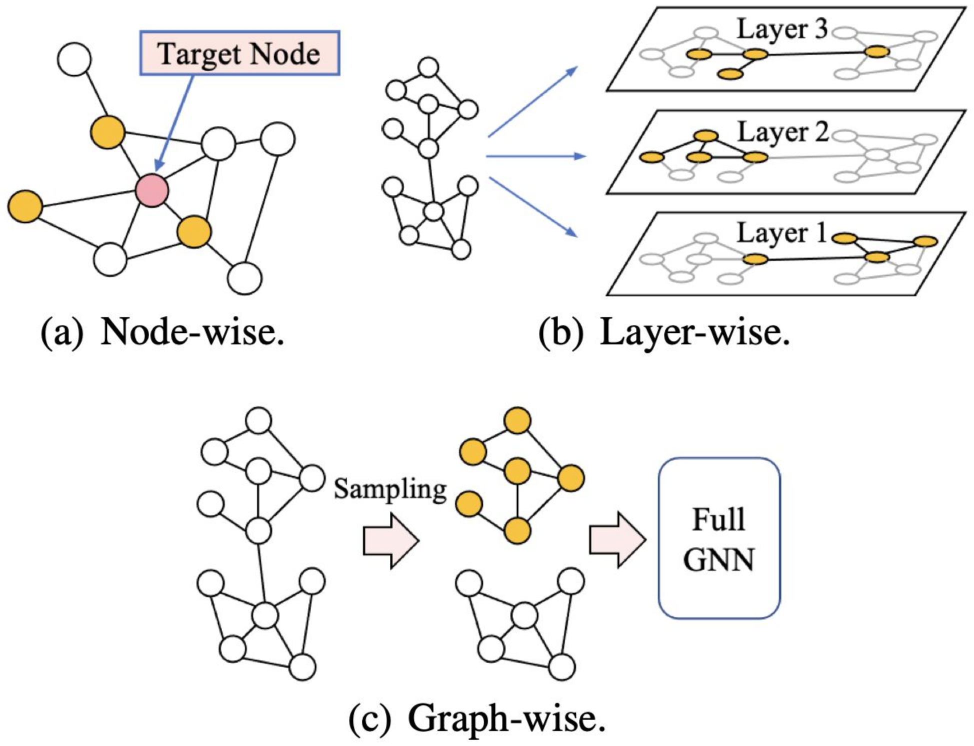
- **Goal:** Compute representation for a sampled node
- **Problem:** Neighborhood Explosion
- **Approach:** Random walks (with fixed number of neighbors considered)
- **Methods:** *GraphSAGE, Hamilton et al., 2017*

Layer-wise:

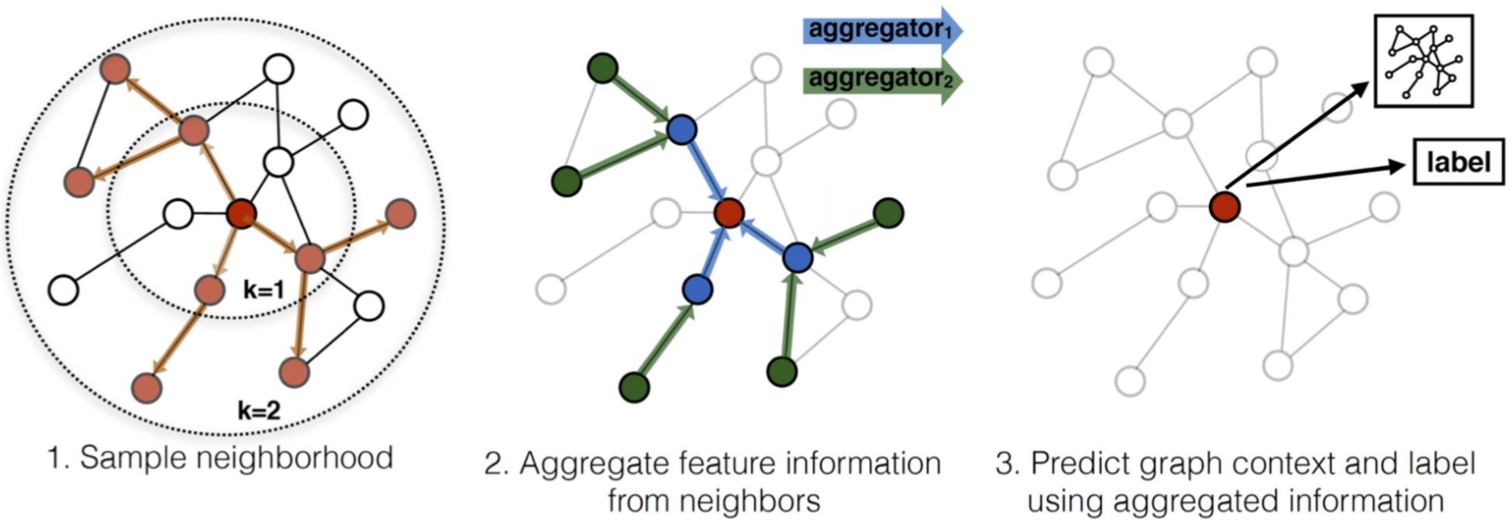
- **Goal:** Create the same amount of signal at each layer and use the same compute budget
- **Problem:** Lose some structural correlations across layers
- **Approach:** Different sampling methods
- **Methods:** *FastGCN, Chen et al.* or *ASGCN, Huang et al.,*

Graph-wise:

- **Goal:** Partition graph and compute representation for all nodes
- **Problem:** Loses some connections across partitions
- **Approach:** Subsampling or partitioning algorithms
- **Methods:** *Cluster-GCN, Chiang et al.* or *GraphSAINT, Zeng et al.*



Node-Wise: *GraphSAGE, Hamilton et al. 2017*

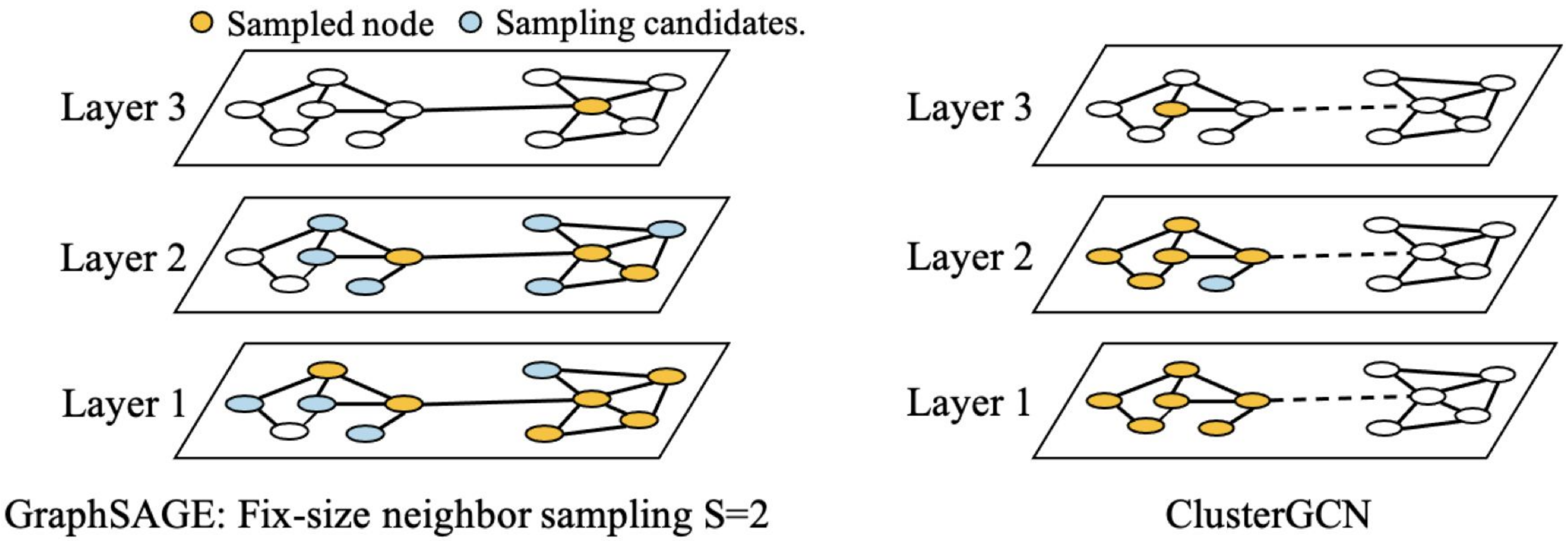


- During training, we sample a *neighborhood* around an anchor node to form a subgraph of *bounded size*
- We then collect a mini-batch of subgraphs
- Inference:
  - Pass the whole graph
  - Pass a subgraph for every node

GraphSAGE, Hamilton et al., 2018: https://arxiv.org/abs/1706.02216

Graph-Wise: *Cluster-GCN, Chiang et al., 2019*

An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks



- Efficient graph partitioning algorithms extract subgraphs before training
- Different partitions never share connections during training
- Proposals to reintroduce omitted links between partitions stochastically
- Strong dependency on the partitioning algorithm and its relation to the target application
- Tackles the neighborhood expansion problem: bounded by the partition
- Useful for distributed computing

Notebook: `sampling.ipynb`

BIOMEDICAL
INFORMATICS

Publications

- Inductive Representation Learning on Large Graphs, Hamilton et al., 2017
- How Powerful are Graph Neural Networks?, Xu et al., 2018
- A Reduction of a Graph to a Canonical Form and an Algebra Arising during This Reduction,Weisfeiler and Leman, 1968
- Multilayer feedforward networks are universal approximators, Hornik et al., 1989
- A Survey on Oversmoothing in Graph Neural Networks, Rusch et al., 2023
- Dirichlet Energy Constrained Learning for Deep Graph Neural Networks, Zhou et al., 2021
- PairNorm: Tackling Oversmoothing in GNNs, Zhao et al., 2019
- DropEdge: Towards Deep Graph Convolutional Networks on Node Classification, Rong et al., 2019
- Hierarchical Graph Representation Learning with Differentiable Pooling, Ying et al., 2018
- GNNBook, Wu et al., 2023
- FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling, Chen et al., 2018
- Adaptive Sampling Towards Fast Graph Representation Learning, Huang et al., 2018
- Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks, Chiang et al., 2019
- GraphSAINT: Graph Sampling Based Inductive Learning Method, Zeng et al., 2019

Slides & Image Credits

1. CS224W: Machine Learning with Graphs: https://web.stanford.edu/class/cs224w/
    a. https://web.stanford.edu/class/cs224w/slides/06-theory.pdf
2. https://bioicons.com/