HOW POWERFUL ARE GRAPH NEURAL NETWORKS ?

刘希阳

6-29

1 INTRODUCTION

- GNNs revolutionizing graph representation learning
- But, limited understanding of their representational properties and limitations

- Design of new GNNs is mostly based on:
  - empirical intuition
  - Heuristics
  - experimental trial-and-error

1 INTRODUCTION

- This paper
  - characterize how expressive different GNN variants are in learning to represent and distinguish between different graph structures
  - show that GNNs are at most as powerful as the WL test in distinguishing graph structures.
  - identify graph structures that cannot be distinguished by popular GNN variants, such as GCN (Kipf & Welling, 2017) and GraphSAGE (Hamilton et al., 2017a)
  - develop a simple neural architecture, Graph Isomorphism Network (GIN)

2 PRELIMINARIES

- Common Graph Neural Networks

$$a_v^{(k)} = \mathbf{AGGREGATE}^{(k)}\left(\left\{h_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right)$$

$$h_v^{(k)} = \mathbf{COMBINE}^{(k)}\left(h_v^{(k-1)}, a_v^{(k)}\right)$$

- For Graphsage

$$a_v^{(k)} = \mathrm{MAX}\left(\left\{\mathrm{ReLU}\left(W \cdot h_u^{(k-1)}\right), \forall u \in \mathcal{N}(v)\right\}\right)$$

- For GCN

$$h_v^{(k)} = \mathrm{ReLU}\left(W \cdot \mathrm{MEAN}\left\{h_u^{(k-1)}, \ \forall u \in \mathcal{N}(v) \cup \{v\}\right\}\right)$$

2 PRELIMINARIES

- Common Prediction Task:
  - For node classification:
    - Directly using $h_v^{(k)}$
  - For graph classification:
    - Form a representation of whole graph with READOUT function

$$h_G = \mathrm{READOUT}\big(\{h_v^{(K)} \mid v \in G\}\big)$$

    - READOUT function can be
      - Summation
      - Sophisticated graph-level pooling function
      - etc.

2 PRELIMINARIES

- Weisfeiler-Lehman test(1968)
  - For graph isomorphism problem(NPC problem)
  - effective and computationally efficient

- Two phases:
  - Aggregates the labels of nodes and their neighborhoods
  - Hashes the aggregated labels into unique new labels

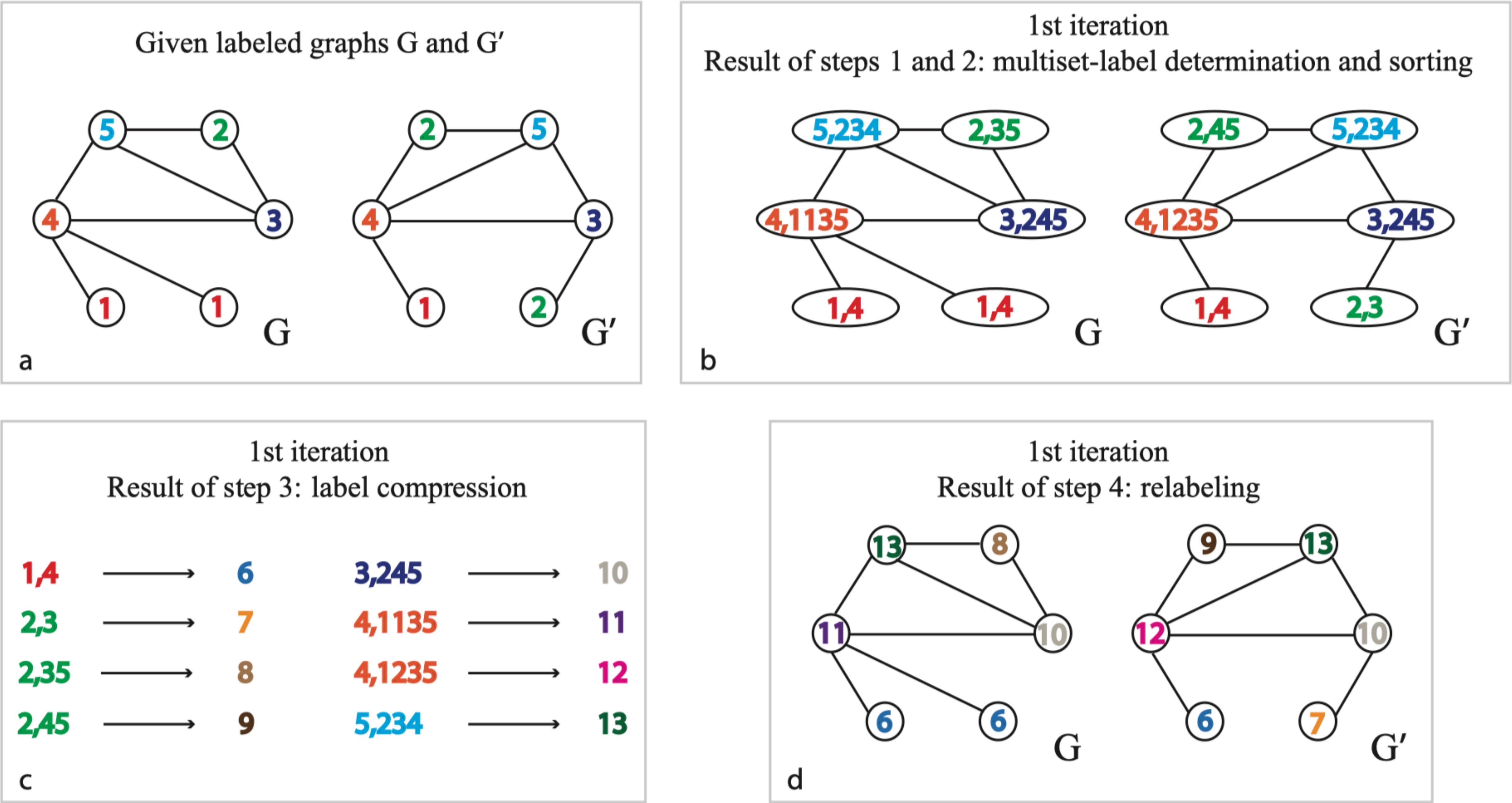
- GCN framework is a similar to WL test

2 PRELIMINARIES

- Weisfeiler-Lehman Graph Kernels(2011)

- Based on the WL test, Shervashidze et al. proposed the WL subtree kernel that measures the similarity between graphs

WL subtree kernel



2 PRELIMINARIES

- Weisfeiler-Lehman Graph Kernels(2011)

End of the 1st iteration

Feature vector representations of G and G′

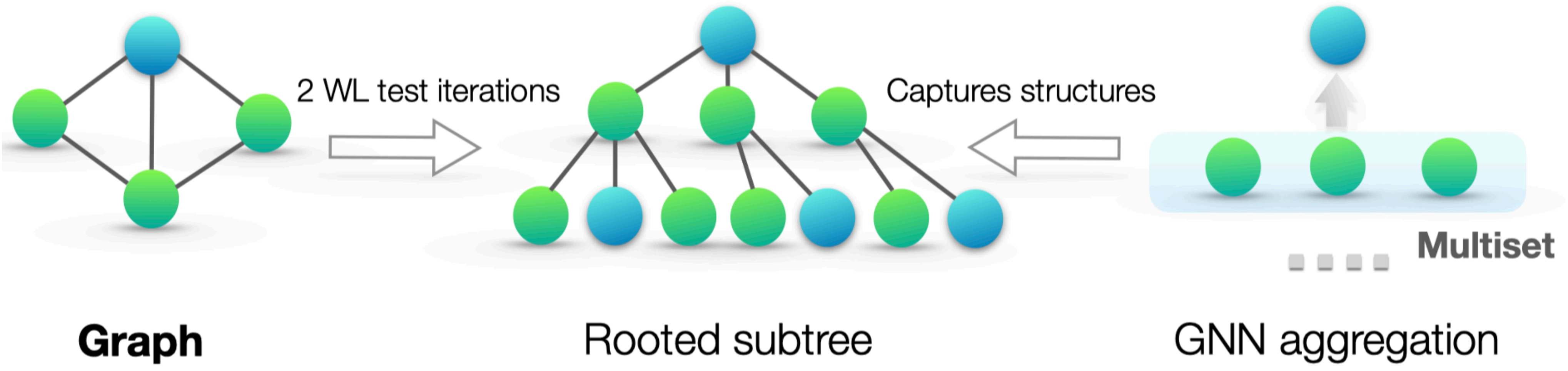
$$\varphi^{(1)}_{WLsubtree}(G) = (2, 1, 1, 1, 1, 2, 0, 1, 0, 1, 1, 0, 1)$$

$$\varphi^{(1)}_{WLsubtree}(G') = (\underbrace{1, 2, 1, 1, 1}_{\text{Counts of original node labels}}, \underbrace{1, 1, 0, 1, 1, 0, 1, 1}_{\text{Counts of compressed node labels}})$$

$$k^{(1)}_{WLsubtree}(G,G') = \langle \varphi^{(1)}_{WLsubtree}(G), \varphi^{(1)}_{WLsubtree}(G') \rangle = 11.$$

e

3 THEORETICAL FRAMEWORK



- To study the representational power of a GNN:
  - whether a GNN maps two neighborhoods (i.e., two multisets) to the same embedding or representation
  - A maximally powerful GNN would never map two different neighborhoods to the same representation
  - **This means its aggregation scheme must be injective**

4 BUILDING POWERFUL GRAPH NEURAL NETWORKS

Lemma 2. *Let G_1 and G_2 be any two non-isomorphic graphs. If a graph neural network $\mathcal{A} : \mathcal{G} \to \mathbb{R}^d$ maps G_1 and G_2 to different embeddings, the Weisfeiler-Lehman graph isomorphism test also decides G_1 and G_2 are not isomorphic.*

- Any aggregation-based GNN is at most as powerful as the WL test in distinguishing different graphs.
- GNN is as powerful as the WL test, if
  - neighbor aggregation is injective
  - graph-level readout function is injective
- an important benefit of GNNs beyond distinguishing different graphs
  - capturing similarity of graph structures
  - WL test are essentially one-hot encodings
  - GNN embed the subtrees to low-dimensional space

4 BUILDING POWERFUL GRAPH NEURAL NETWORKS

- GRAPH ISOMORPHISM NETWORK (GIN)
  - generalizes the WL test and hence achieves maximum discriminative power among GNNs
- For node embedding

$$h_v^{(k)} = \mathrm{MLP}^{(k)}\left(\left(1 + \epsilon^{(k)}\right) \cdot h_v^{(k-1)} + \sum\nolimits_{u \in \mathcal{N}(v)} h_u^{(k-1)}\right)$$

- For graph embedding

$$h_G = \mathrm{CONCAT}\Big(\mathrm{READOUT}\Big(\left\{h_v^{(k)} | v \in G\right\}\Big) \mid k = 0, 1, \ldots, K\Big)$$

5 LESS POWERFUL BUT STILL INTERESTING GNNS

- Study two aspects of the aggregator
  - 1-layer perceptrons instead of MLPs
  - mean or max-pooling instead of the sum

- MLP: Universal approximation theorem (Hornik et al., 1989; Hornik, 1991)
- Unlike models using MLPs, the 1-layer perceptron (even with the bias term) is not a universal approximator of multiset functions

Lemma 7. *There exist finite multisets* $X_1 \neq X_2$ *so that for any linear mapping* W, $\sum_{x \in X_1} \mathrm{ReLU}(Wx) = \sum_{x \in X_2} \mathrm{ReLU}(Wx)$.

5 LESS POWERFUL BUT STILL INTERESTING GNNS

- Mean aggregators captures the proportion/distribution of elements of a given type
- Max aggregator ignores multiplicities, learns sets with distinct elements

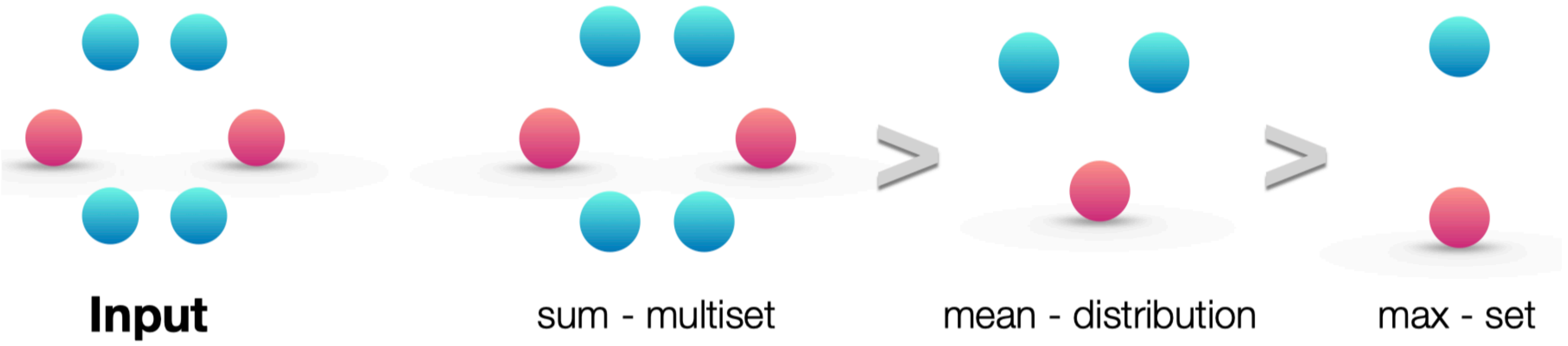


Figure 2: **Ranking by expressive power for sum, mean and max aggregators over a multiset.**

5 LESS POWERFUL BUT STILL INTERESTING GNNS

- Mean aggregator is as powerful as the sum aggregator when node features are diverse and rarely repeat
- Max-pooling may be suitable for tasks where it is important to identify representative elements or the “skeleton”, rather than to distinguish the exact structure or distribution

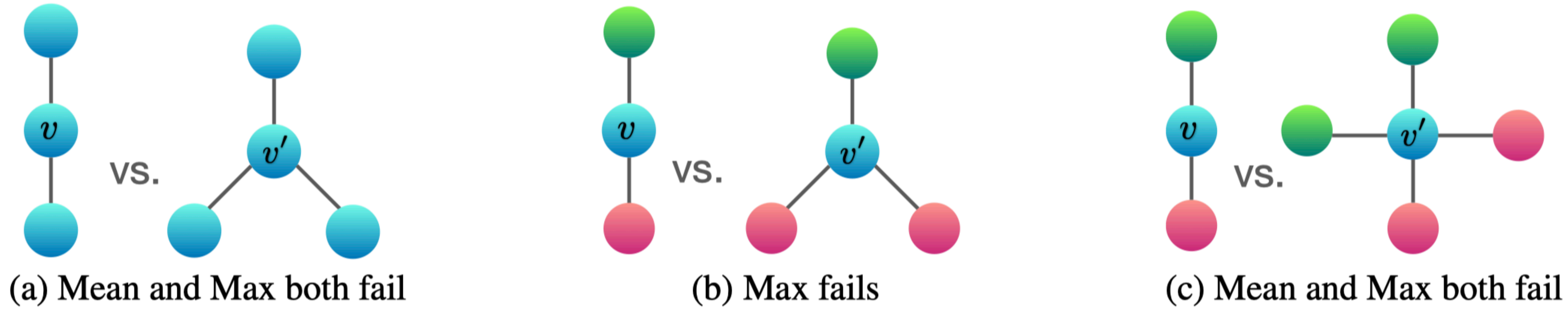


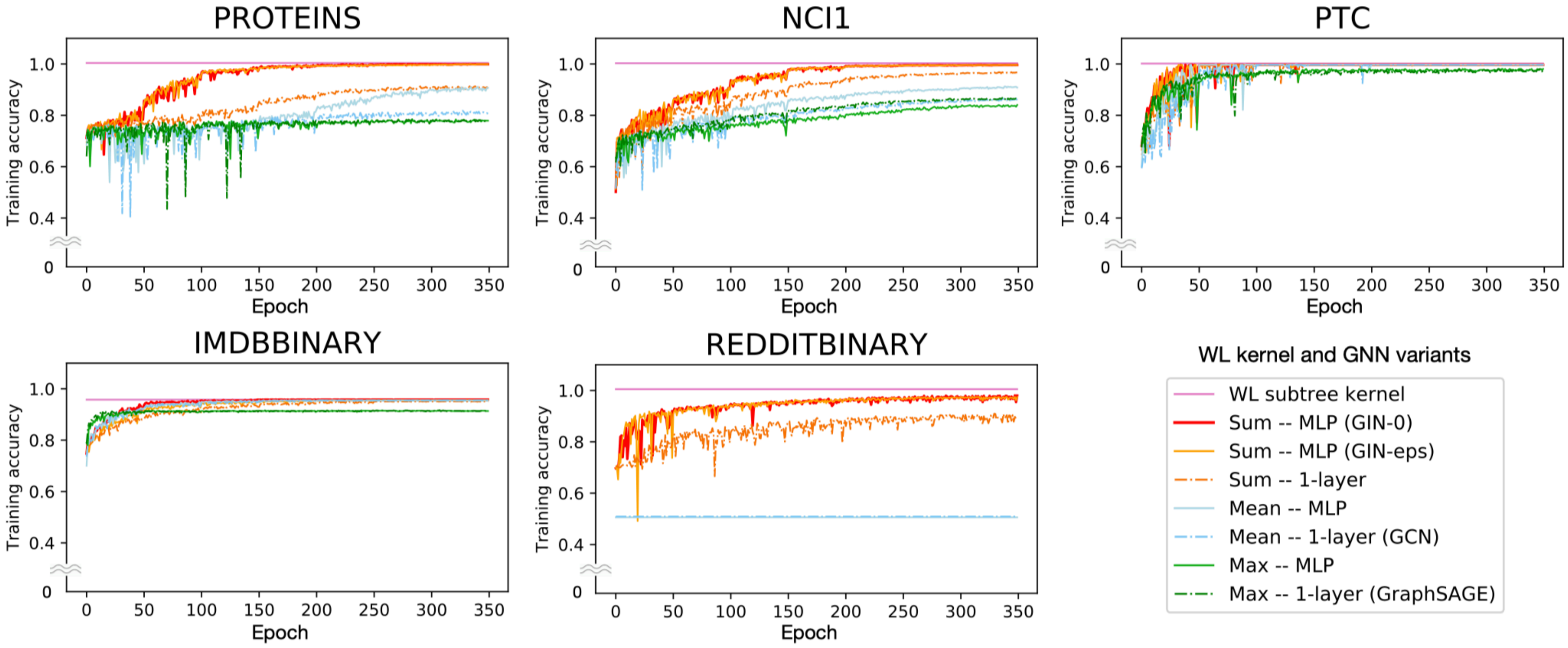
Figure 3: **Examples of graph structures that mean and max aggregators fail to distinguish.**

6 EXPERIMENTS

- 9 graph classification benchmarks
  - 4 bioinformatics datasets (MUTAG, PTC, NCI1, PROTEINS)
  - 5 social network datasets (COLLAB, IMDB-BINARY, IMDB-MULTI, REDDITBINARY and REDDIT-MULTI5K)
- Models and configurations:
  - 10-fold cross-validation
  - 5 GNN layers (including the input layer)
  - MLPs have 2 layers
  - learning rate 0.01

6 EXPERIMENTS

- Training set performance



6 EXPERIMENTS

- Test set performance

	Datasets	IMDB-B	IMDB-M	RDT-B	RDT-M5K	COLLAB	MUTAG	PROTEINS	PTC	NCI1
Datasets	# graphs	1000	1500	2000	5000	5000	188	1113	344	4110
	# classes	2	3	2	5	3	2	2	2	2
	Avg # nodes	19.8	13.0	429.6	508.5	74.5	17.9	39.1	25.5	29.8
Baselines	WL subtree	73.8 ± 3.9	50.9 ± 3.8	81.0 ± 3.1	52.5 ± 2.1	78.9 ± 1.9	90.4 ± 5.7	75.0 ± 3.1	59.9 ± 4.3	$\mathbf{86.0 \pm 1.8}$ *
	DCNN	49.1	33.5	–	–	52.1	67.0	61.3	56.6	62.6
	PatchySan	71.0 ± 2.2	45.2 ± 2.8	86.3 ± 1.6	49.1 ± 0.7	72.6 ± 2.2	$\mathbf{92.6 \pm 4.2}$ *	75.9 ± 2.8	60.0 ± 4.8	78.6 ± 1.9
	DGCNN	70.0	47.8	–	–	73.7	85.8	75.5	58.6	74.4
	AWL	74.5 ± 5.9	51.5 ± 3.6	87.9 ± 2.5	54.7 ± 2.9	73.9 ± 1.9	87.9 ± 9.8	–	–	–
GNN variants	Sum–MLP (**GIN-0**)	$\mathbf{75.1 \pm 5.1}$	$\mathbf{52.3 \pm 2.8}$	$\mathbf{92.4 \pm 2.5}$	$\mathbf{57.5 \pm 1.5}$	$\mathbf{80.2 \pm 1.9}$	$\mathbf{89.4 \pm 5.6}$	$\mathbf{76.2 \pm 2.8}$	$\mathbf{64.6 \pm 7.0}$	$\mathbf{82.7 \pm 1.7}$
	Sum–MLP (**GIN-ϵ**)	$\mathbf{74.3 \pm 5.1}$	$\mathbf{52.1 \pm 3.6}$	$\mathbf{92.2 \pm 2.3}$	$\mathbf{57.0 \pm 1.7}$	$\mathbf{80.1 \pm 1.9}$	$\mathbf{89.0 \pm 6.0}$	$\mathbf{75.9 \pm 3.8}$	63.7 ± 8.2	$\mathbf{82.7 \pm 1.6}$
	Sum–1-Layer	74.1 ± 5.0	$\mathbf{52.2 \pm 2.4}$	90.0 ± 2.7	55.1 ± 1.6	$\mathbf{80.6 \pm 1.9}$	$\mathbf{90.0 \pm 8.8}$	$\mathbf{76.2 \pm 2.6}$	63.1 ± 5.7	82.0 ± 1.5
	Mean–MLP	73.7 ± 3.7	$\mathbf{52.3 \pm 3.1}$	50.0 ± 0.0	20.0 ± 0.0	79.2 ± 2.3	83.5 ± 6.3	75.5 ± 3.4	$\mathbf{66.6 \pm 6.9}$	80.9 ± 1.8
	Mean–1-Layer (GCN)	74.0 ± 3.4	51.9 ± 3.8	50.0 ± 0.0	20.0 ± 0.0	79.0 ± 1.8	85.6 ± 5.8	76.0 ± 3.2	64.2 ± 4.3	80.2 ± 2.0
	Max–MLP	73.2 ± 5.8	51.1 ± 3.6	–	–	–	84.0 ± 6.1	76.0 ± 3.2	64.6 ± 10.2	77.8 ± 1.3
	Max–1-Layer (GraphSAGE)	72.3 ± 5.3	50.9 ± 2.2	–	–	–	85.1 ± 7.6	75.9 ± 3.2	63.9 ± 7.7	77.7 ± 1.5

7 CONCLUSION

- ❑Weighted average via attention and LSTM pooling aggregation were not covered.
- ❑Sum aggregator outperforms mean and max-pooling.
- ❑MLP outperforms 1-layer perception which commonly used.

THANKS