How Powerful are Graph Neural Networks?

Keyulu Xu, Weihua Hu*, Jure Leskovec, Stefanie Jegelka*

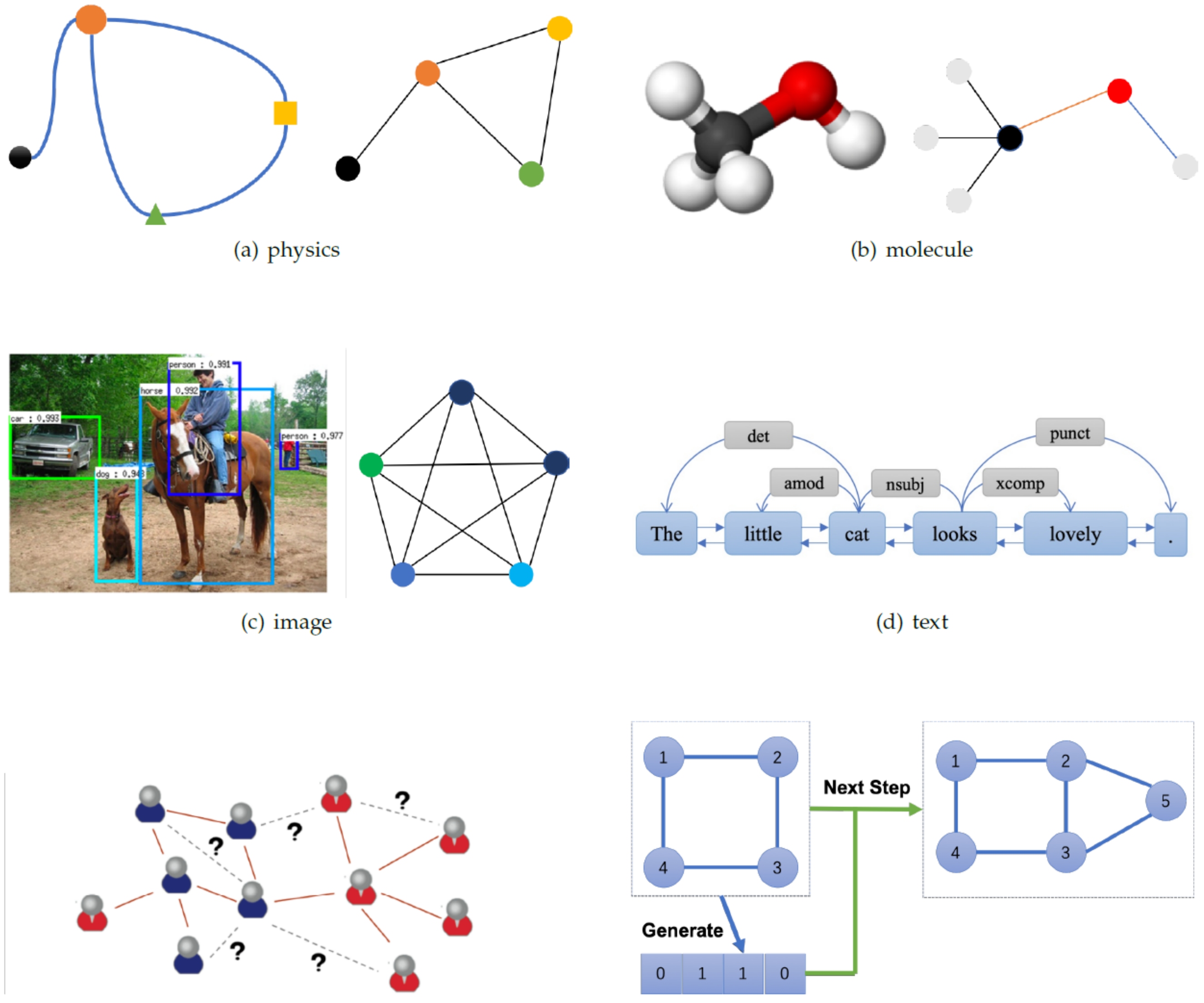
Reporter: 邵云帆 (19210240032)
yfshao19@fudan.edu.cn

https://arxiv.org/abs/1810.00826

Introduction

- Success of deep learning on simple structured data
  - grids/images
  - sequences/text
  - more …

- Networks/graphs allow us to model complex domains
  - Social networks
  - Chemical structure
  - Biomedicine
  - more …

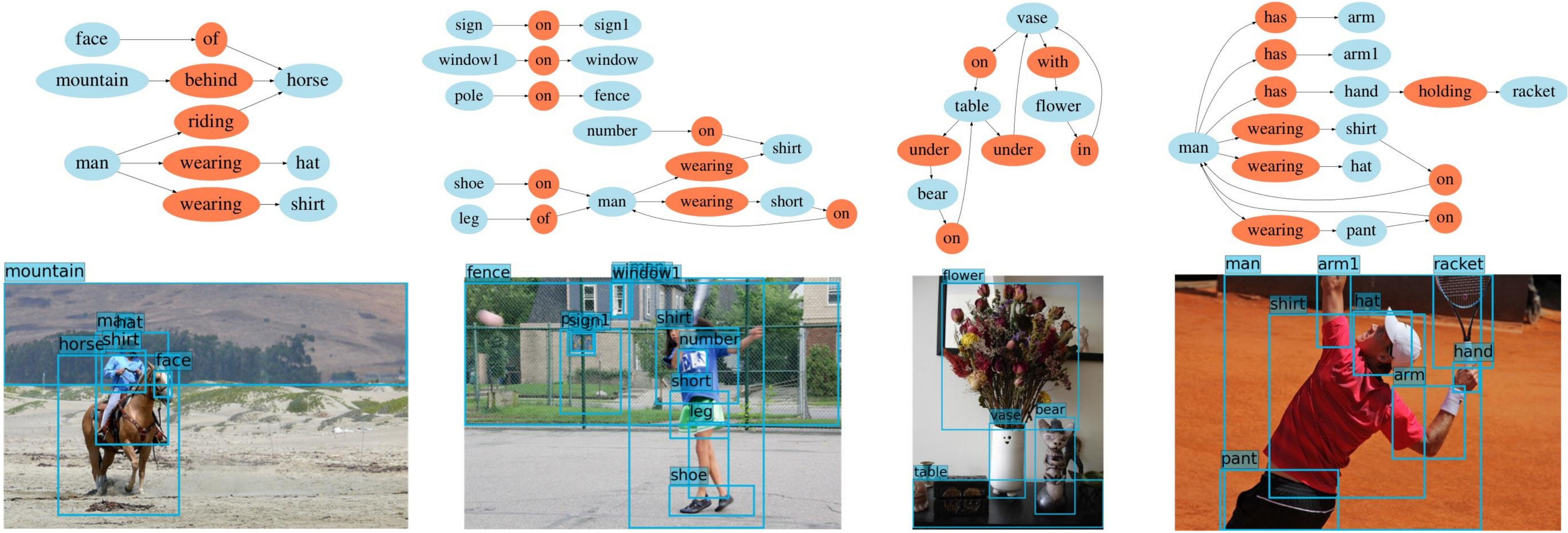


(a) physics (b) molecule (c) image (d) text (e) social network (f) generation

Applications

Scene graph

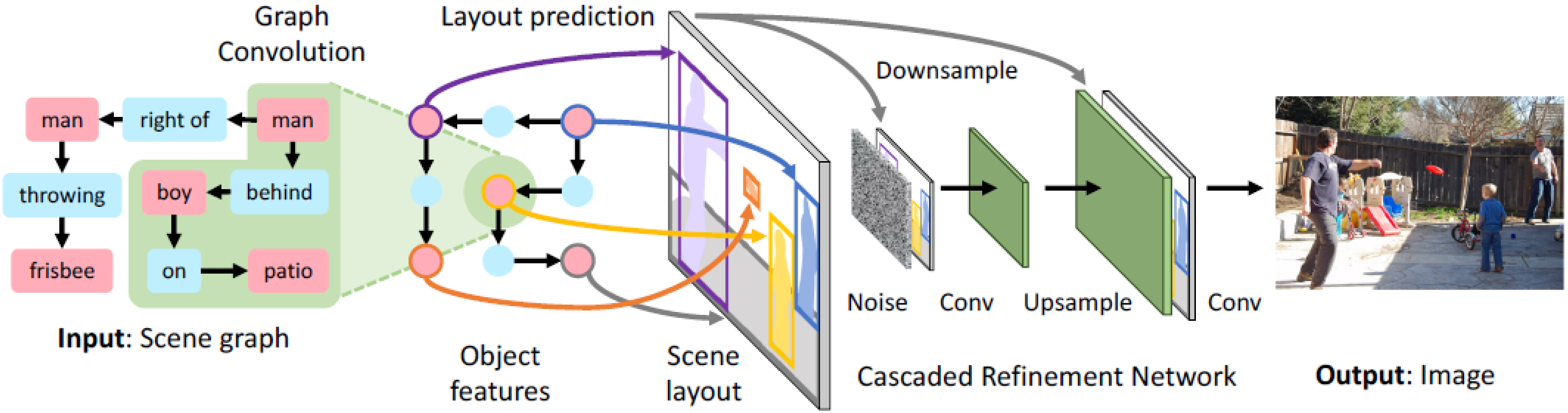
- A structured formal graphical representation of an image.
- Encodes objects as nodes connected via pairwise relationships as edges.



Applications

Scene graph to image:

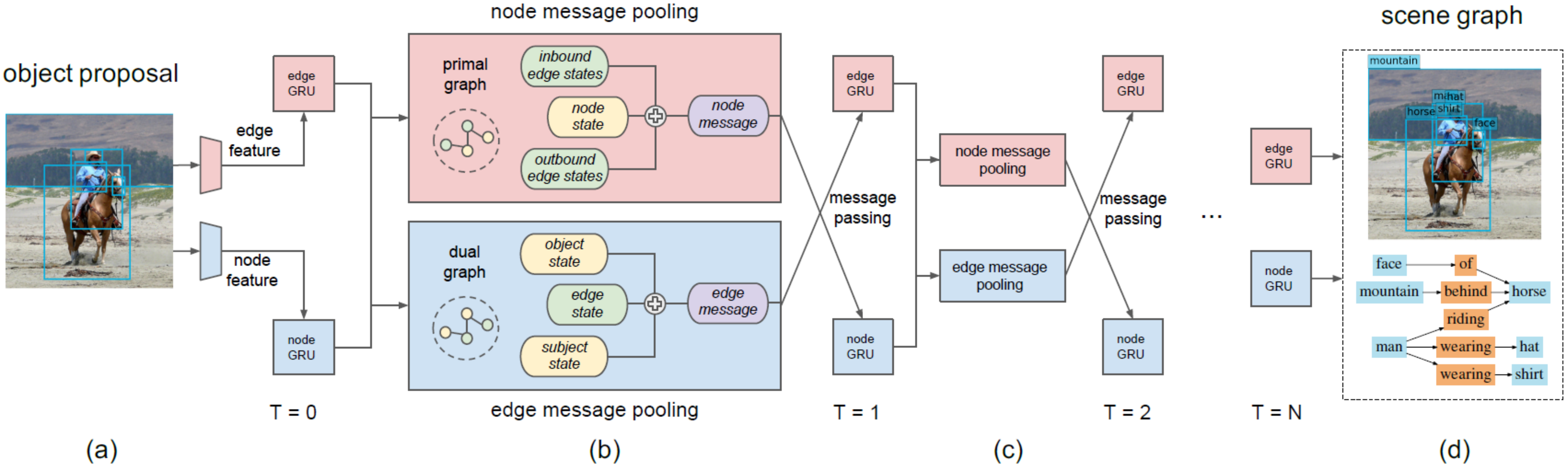
- **Image Generation from Scene Graphs** (CVPR 2018)
  - Generate scene layouts.
  - Use layouts generate images.



Applications

Image to scene graph

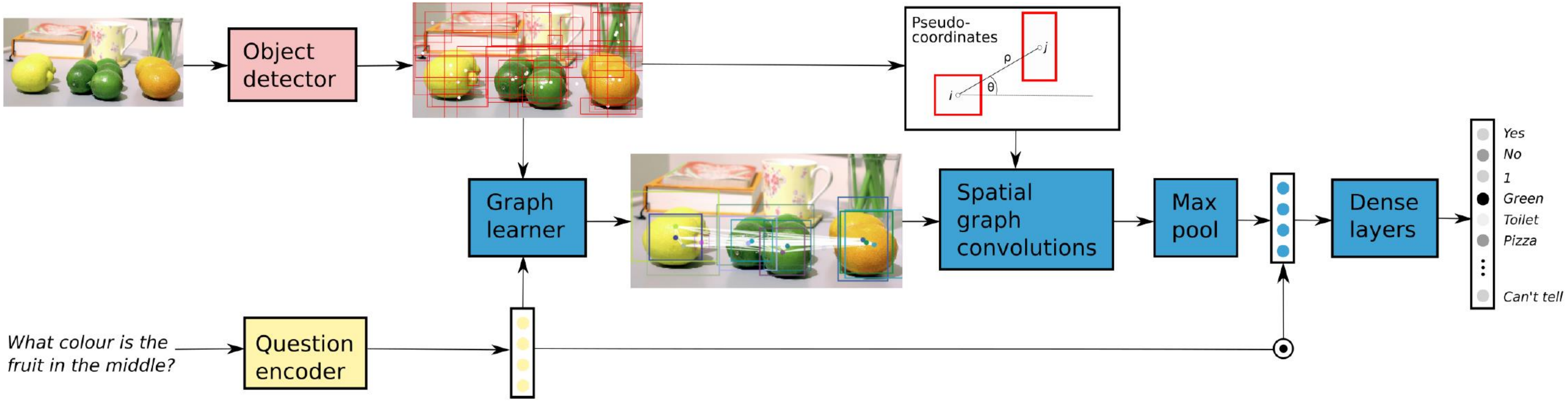
- **Scene Graph Generation by Iterative Message Passing** (CVPR 2018)
  - Use Region Proposal Network (RPN) generates bounding box proposals.
  - Predict node labels and edge labels.



Applications

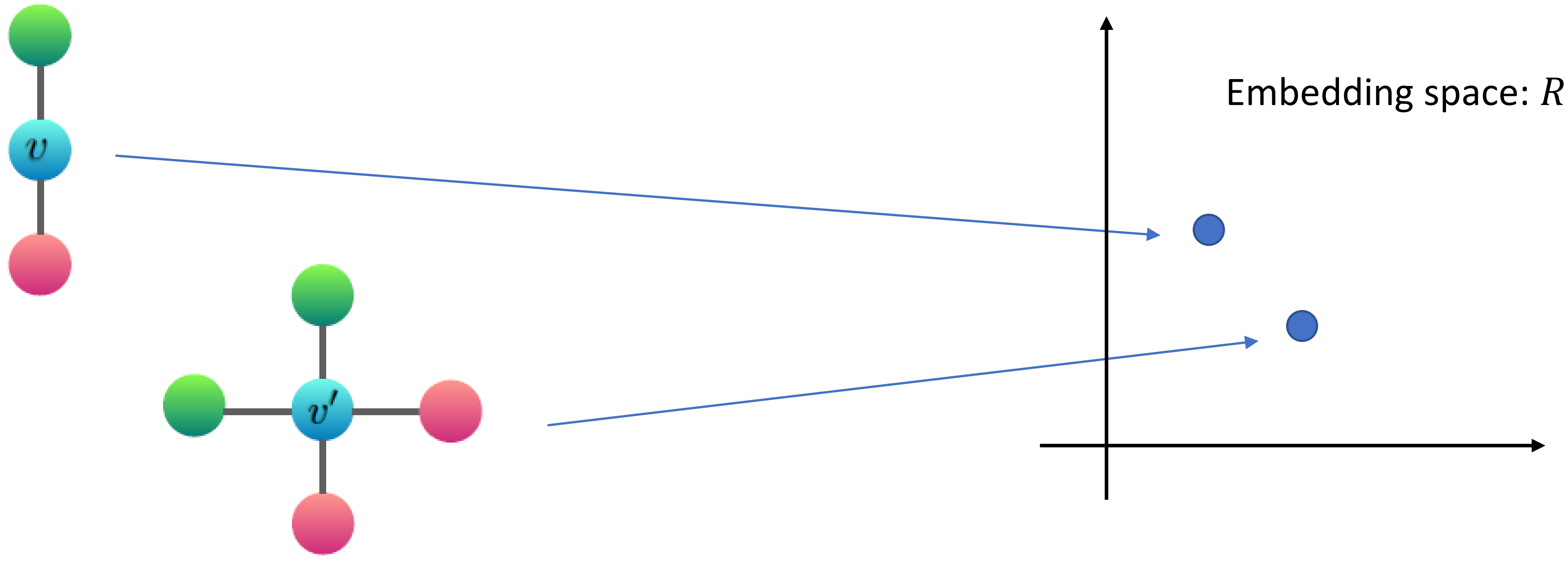
VQA

- [Learning Conditioned Graph Structures for Interpretable Visual Question Answering](). (NeurIPS 2018)
- Input a question encoding, and a set of object bounding boxes.
- Obtain graph representations to learn image features.



Graph Representation Learning

- **Input**: Graph with node features
- **Task**: Node/Graph classification
- **Goal**: Learning node/graph embeddings that capture structure information



Graph Neural Networks (GNNs)

Input:

Graph $G = (V, E)$ with Node features X_v

GNN: Learning graph features using graph structure and node features.

- Aggregate

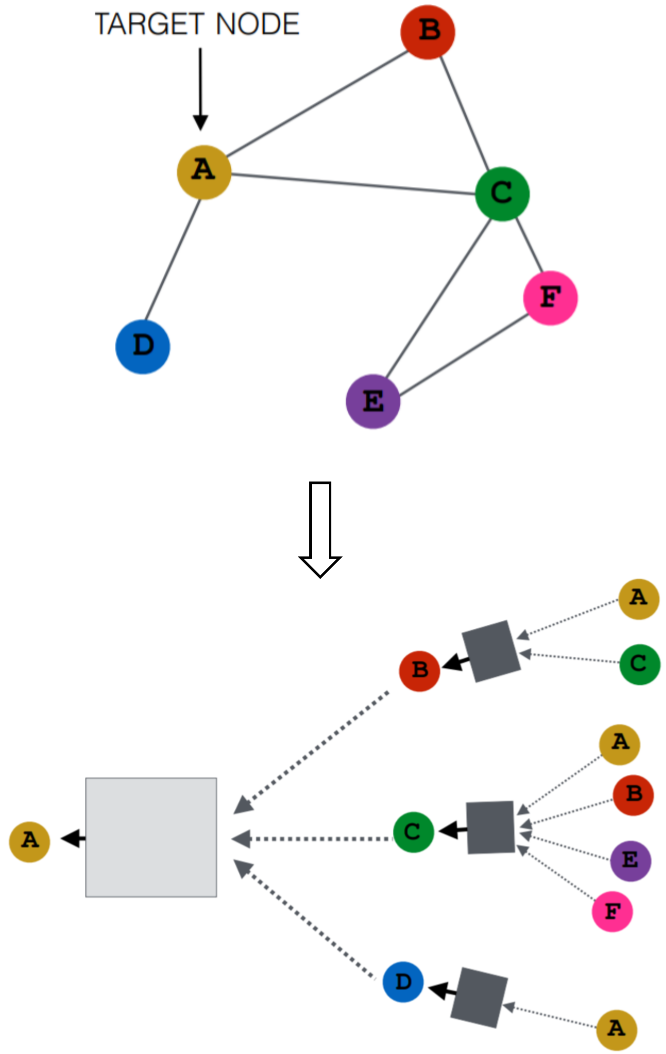
$$a_v^{(k)} = \text{AGGREGATE}^{(k)}\left(\left\{h_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right)$$

- Combine

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

- Readout

$$h_G = \text{READOUT}\left(\left\{h_v^{(K)} \mid v \in G\right\}\right)$$



Graph Neural Networks (GNNs)

Different variants of graph neural networks.

Name	Variant	Aggregator	Updater
Spectral Methods	ChebNet	$\mathbf{N}_k = \mathbf{T}_k(\tilde{\mathbf{L}})\mathbf{X}$	$\mathbf{H} = \sum_{k=0}^{K} \mathbf{N}_k \mathbf{\Theta}_k$
	1^{st}-order model	$\mathbf{N}_0 = \mathbf{X}$ $\mathbf{N}_1 = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\mathbf{X}$	$\mathbf{H} = \mathbf{N}_0\mathbf{\Theta}_0 + \mathbf{N}_1\mathbf{\Theta}_1$
	Single parameter	$\mathbf{N} = (\mathbf{I}_N + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{X}$	$\mathbf{H} = \mathbf{N}\mathbf{\Theta}$
	GCN	$\mathbf{N} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}$	$\mathbf{H} = \mathbf{N}\mathbf{\Theta}$
Non-spectral Methods	Neural FPs	$\mathbf{h}_{\mathcal{N}_v}^t = \mathbf{h}_v^{t-1} + \sum_{k=1}^{\mathcal{N}_v} \mathbf{h}_k^{t-1}$	$\mathbf{h}_v^t = \sigma(\mathbf{h}_{\mathcal{N}_v}^t \mathbf{W}_L^{\mathcal{N}_v})$
	DCNN	Node classification: $\mathbf{N} = \mathbf{P}^*\mathbf{X}$ Graph classification: $\mathbf{N} = 1_N^T\mathbf{P}^*\mathbf{X}/N$	$\mathbf{H} = f\left(\mathbf{W}^c \odot \mathbf{N}\right)$
	GraphSAGE	$\mathbf{h}_{\mathcal{N}_v}^t = \text{AGGREGATE}_t\left(\{\mathbf{h}_u^{t-1}, \forall u \in \mathcal{N}_v\}\right)$	$\mathbf{h}_v^t = \sigma\left(\mathbf{W}^t \cdot [\mathbf{h}_v^{t-1} \Vert \mathbf{h}_{\mathcal{N}_v}^t]\right)$

Graph Neural Networks: A Review of Methods and Applications. J. Zhou, G. Cui, Z. Zhang, et al.

Graph Neural Networks (GNNs)

Graph Attention Networks	GAT	$\alpha_{vk} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T[\mathbf{W}\mathbf{h}_v \Vert \mathbf{W}\mathbf{h}_k]\right)\right)}{\sum_{j\in\mathcal{N}_v}\exp\left(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{h}_v \Vert \mathbf{W}\mathbf{h}_j])\right)}$ $\mathbf{h}_{\mathcal{N}_v}^t = \sigma\left(\sum_{k\in\mathcal{N}_v}\alpha_{vk}\mathbf{W}\mathbf{h}_k\right)$ Multi-head concatenation: $\mathbf{h}_{\mathcal{N}_v}^t = \Vert_{m=1}^{M}\sigma\left(\sum_{k\in\mathcal{N}_v}\alpha_{vk}^m\mathbf{W}^m\mathbf{h}_k\right)$ Multi-head average: $\mathbf{h}_{\mathcal{N}_v}^t = \sigma\left(\frac{1}{M}\sum_{m=1}^{M}\sum_{k\in\mathcal{N}_v}\alpha_{vk}^m\mathbf{W}^m\mathbf{h}_k\right)$	$\mathbf{h}_v^t = \mathbf{h}_{\mathcal{N}_v}^t$
Gated Graph Neural Networks	GGNN	$\mathbf{h}_{\mathcal{N}_v}^t = \sum_{k\in\mathcal{N}_v}\mathbf{h}_k^{t-1} + \mathbf{b}$	$\mathbf{z}_v^t = \sigma(\mathbf{W}^z\mathbf{h}_{\mathcal{N}_v}^t + \mathbf{U}^z\mathbf{h}_v^{t-1})$ $\mathbf{r}_v^t = \sigma(\mathbf{W}^r\mathbf{h}_{\mathcal{N}_v}^t + \mathbf{U}^r\mathbf{h}_v^{t-1})$ $\widetilde{\mathbf{h}_v^t} = \tanh(\mathbf{W}\mathbf{h}_{\mathcal{N}_v}^t + \mathbf{U}(\mathbf{r}_v^t \odot \mathbf{h}_v^{t-1}))$ $\mathbf{h}_v^t = (1-\mathbf{z}_v^t)\odot\mathbf{h}_v^{t-1} + \mathbf{z}_v^t\odot\widetilde{\mathbf{h}_v^t}$
Graph LSTM	Tree LSTM (Child sum)	$\mathbf{h}_{\mathcal{N}_v}^t = \sum_{k\in\mathcal{N}_v}\mathbf{h}_k^{t-1}$	$\mathbf{i}_v^t = \sigma(\mathbf{W}^i\mathbf{x}_v^t + \mathbf{U}^i\mathbf{h}_{\mathcal{N}_v}^t + \mathbf{b}^i)$ $\mathbf{f}_{vk}^t = \sigma\left(\mathbf{W}^f\mathbf{x}_v^t + \mathbf{U}^f\mathbf{h}_k^{t-1} + \mathbf{b}^f\right)$ $\mathbf{o}_v^t = \sigma(\mathbf{W}^o\mathbf{x}_v^t + \mathbf{U}^o\mathbf{h}_{\mathcal{N}_v}^t + \mathbf{b}^o)$ $\mathbf{u}_v^t = \tanh(\mathbf{W}^u\mathbf{x}_v^t + \mathbf{U}^u\mathbf{h}_{\mathcal{N}_v}^t + \mathbf{b}^u)$ $\mathbf{c}_v^t = \mathbf{i}_v^t\odot\mathbf{u}_v^t + \sum_{k\in\mathcal{N}_v}\mathbf{f}_{vk}^t\odot\mathbf{c}_k^{t-1}$ $\mathbf{h}_v^t = \mathbf{o}_v^t\odot\tanh(\mathbf{c}_v^t)$
	Tree LSTM (N-ary)	$\mathbf{h}_{\mathcal{N}_v}^{ti} = \sum_{l=1}^{K}\mathbf{U}_l^i\mathbf{h}_{vl}^{t-1}$ $\mathbf{h}_{\mathcal{N}_v k}^{tf} = \sum_{l=1}^{K}\mathbf{U}_{kl}^f\mathbf{h}_{vl}^{t-1}$ $\mathbf{h}_{\mathcal{N}_v}^{to} = \sum_{l=1}^{K}\mathbf{U}_l^o\mathbf{h}_{vl}^{t-1}$ $\mathbf{h}_{\mathcal{N}_v}^{tu} = \sum_{l=1}^{K}\mathbf{U}_l^u\mathbf{h}_{vl}^{t-1}$	$\mathbf{i}_v^t = \sigma(\mathbf{W}^i\mathbf{x}_v^t + \mathbf{h}_{\mathcal{N}_v}^{ti} + \mathbf{b}^i)$ $\mathbf{f}_{vk}^t = \sigma(\mathbf{W}^f\mathbf{x}_v^t + \mathbf{h}_{\mathcal{N}_v k}^{tf} + \mathbf{b}^f)$ $\mathbf{o}_v^t = \sigma(\mathbf{W}^o\mathbf{x}_v^t + \mathbf{h}_{\mathcal{N}_v}^{to} + \mathbf{b}^o)$ $\mathbf{u}_v^t = \tanh(\mathbf{W}^u\mathbf{x}_v^t + \mathbf{h}_{\mathcal{N}_v}^{tu} + \mathbf{b}^u)$ $\mathbf{c}_v^t = \mathbf{i}_v^t\odot\mathbf{u}_v^t + \sum_{l=1}^{K}\mathbf{f}_{vl}^t\odot\mathbf{c}_{vl}^{t-1}$ $\mathbf{h}_v^t = \mathbf{o}_v^t\odot\tanh(\mathbf{c}_v^t)$

Graph Neural Networks: A Review of Methods and Applications. J. Zhou, G. Cui, Z. Zhang, et al.

How powerful are GNNs

Motivation

- **Many GNN variants** are designed based on **empirical intuition**, heuristics, and experimental trial-and error.
- There is **little theoretical understanding** of the properties and limitations of GNNs.
- Formal analysis of GNNs' **representational capacity** is limited.

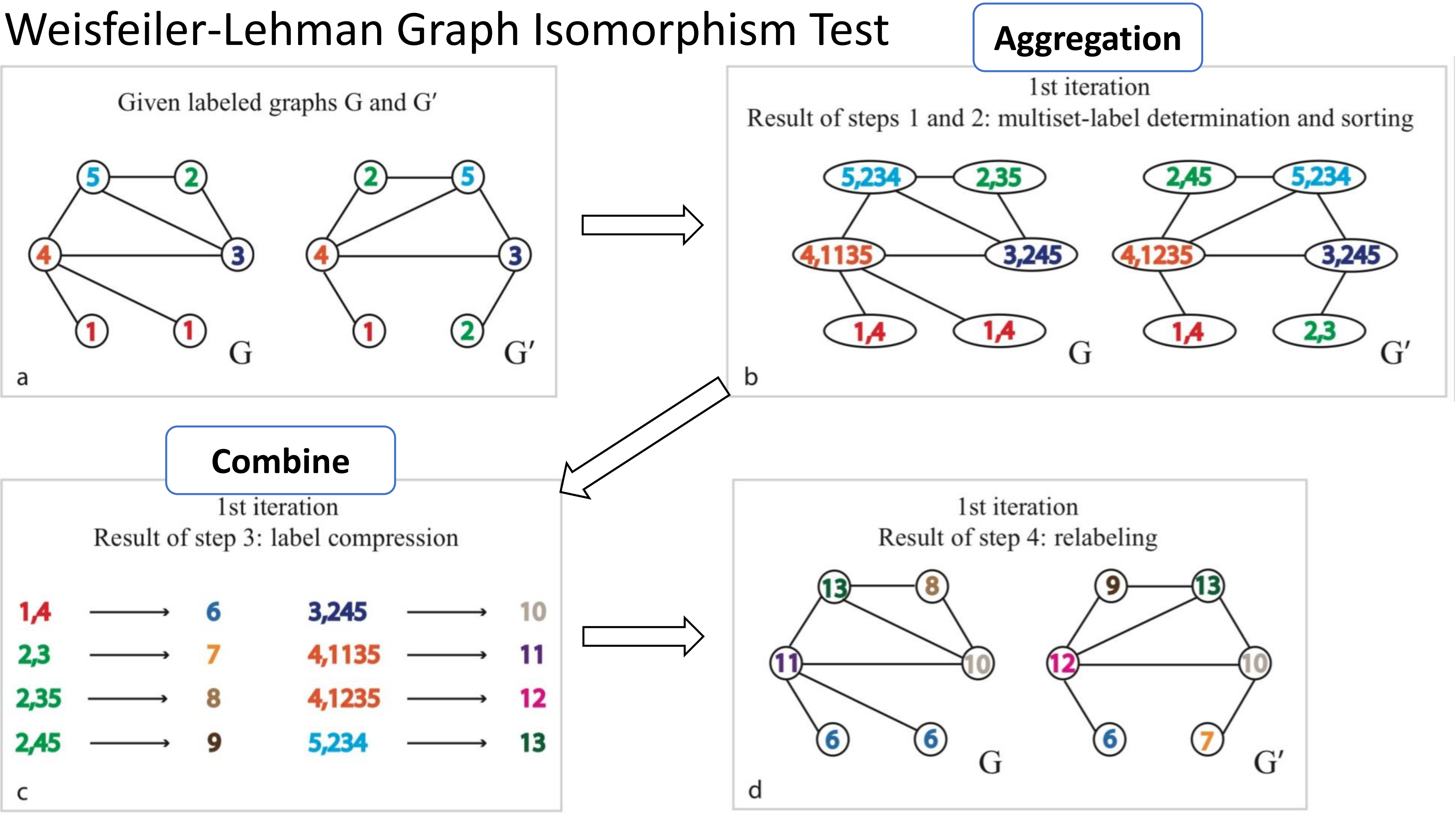
How powerful are GNNs

Contribution

- Figure out the **discriminative power of GNNs**
  - Discriminative power: Map different graphs to different embedding
- Show the **maximum capacity** of GNNs theoretically
- Identify **structures that cannot be distinguished** by popular GNN variants
- Develop a **new GNN architecture** (GIN) and show its discriminative power

WL Test

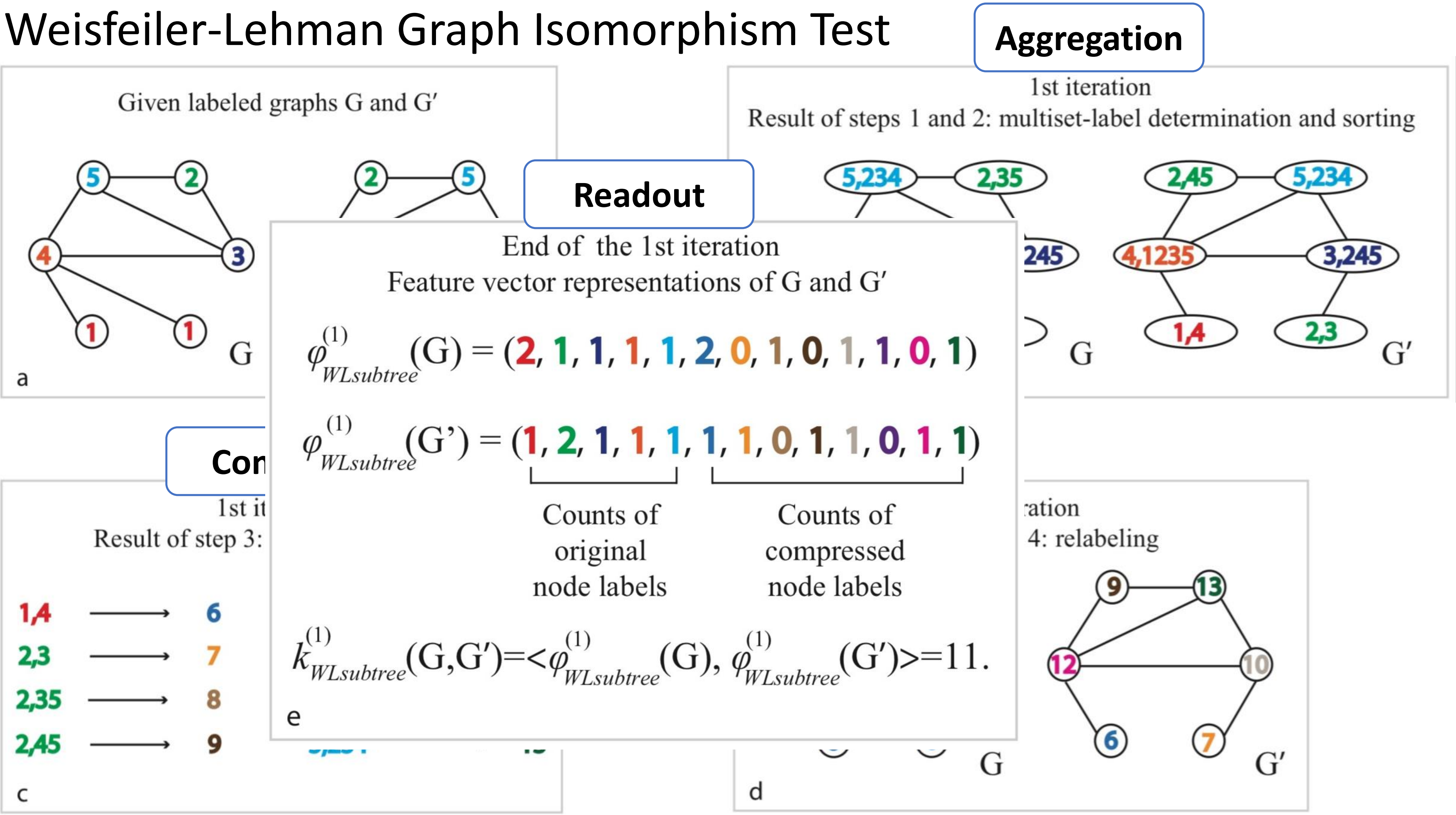
- Weisfeiler-Lehman Graph Isomorphism Test



[Weisfeiler-Lehman Graph Kernels](). N. Shervashidze, et al.

WL Test

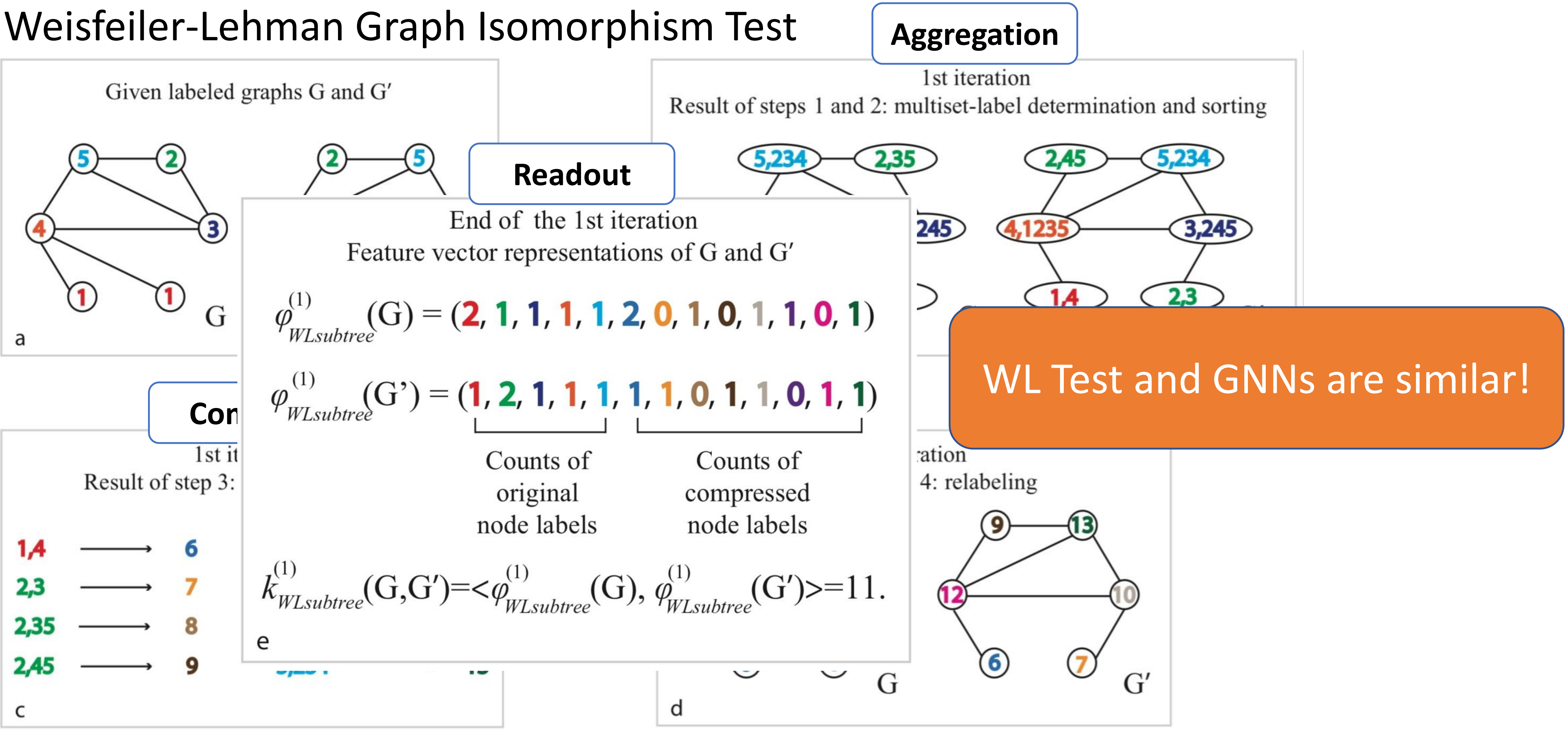
- Weisfeiler-Lehman Graph Isomorphism Test



Weisfeiler-Lehman Graph Kernels. N. Shervashidze, et al.

WL Test

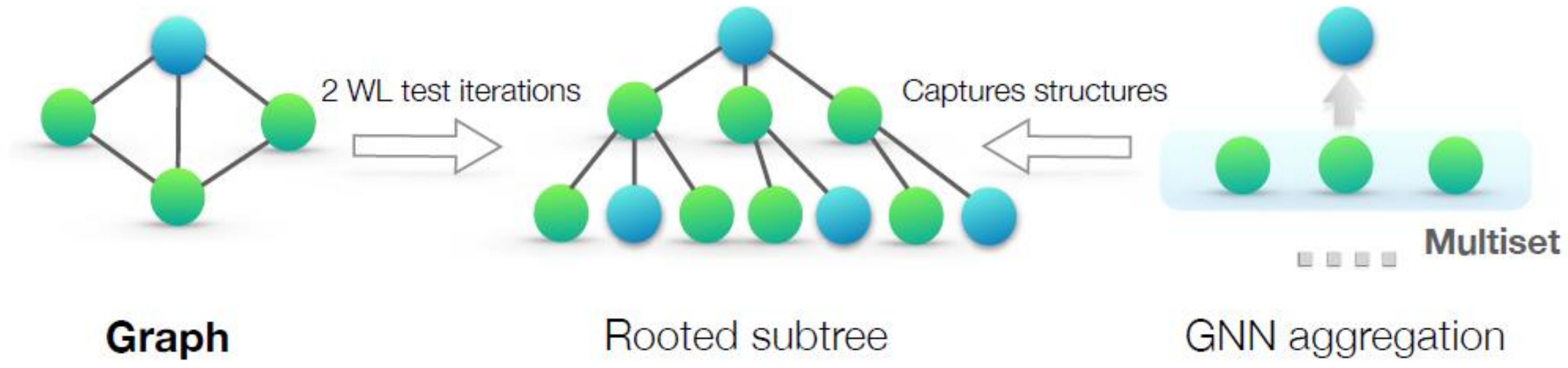
- Weisfeiler-Lehman Graph Isomorphism Test



Weisfeiler-Lehman Graph Kernels. N. Shervashidze, et al.

Theorem Framework

- Graph nodes as Multiset
- Both WL test & GNNs capture graph structures



An overview of our theoretical framework. Middle panel: rooted subtree structures (at the blue node) that the WL test uses to distinguish different graphs. Right panel: if a GNN's aggregation function captures the *full multiset* of node neighbors, the GNN can capture the rooted subtrees in a recursive manner and be as powerful as the WL test.

Theorem Framework

Theorem 1:

GNNs can be at most as powerful as the Weisfeiler-Lehman graph isomorphism test (a.k.a. canonical labeling or color refinement)

Theorem 2:

A maximally powerful GNN would never map two different neighborhoods, i.e., multisets of feature vectors, to the same representation. This means its aggregation scheme must be injective.

Theorem Framework

Theorem 3. *Let $\mathcal{A}: \mathcal{G} \rightarrow \mathbb{R}^d$ be a GNN. With a sufficient number of GNN layers, $\mathcal{A}$ maps any graphs G_1 and G_2 that the Weisfeiler-Lehman test of isomorphism decides as non-isomorphic, to different embeddings if the following conditions hold:*

a) $\mathcal{A}$ aggregates and updates node features iteratively with

$$h_v^{(k)} = \phi\left(h_v^{(k-1)}, f\left(\left\{h_u^{(k-1)} : u \in \mathcal{N}(v)\right\}\right)\right),$$

where the functions f, which operates on multisets, and ϕ are injective.

b) $\mathcal{A}$'s graph-level readout, which operates on the multiset of node features $\left\{h_v^{(k)}\right\}$, is injective.

Refer the paper for more theorems.

GIN

In theory, $h_v^k = \phi(f(\{h_v^{k-1}, for\ v\ in\ V\}))$

$$h_v^{(k)} = \underbrace{\mathrm{MLP}^{(k)}\left(\left(1+\epsilon^{(k)}\right)\cdot h_v^{(k-1)} + \sum_{u\in\mathcal{N}(v)}\right.}_{\text{injective } \phi\ \&\ f} h_u^{(k-1)}\Bigg)$$

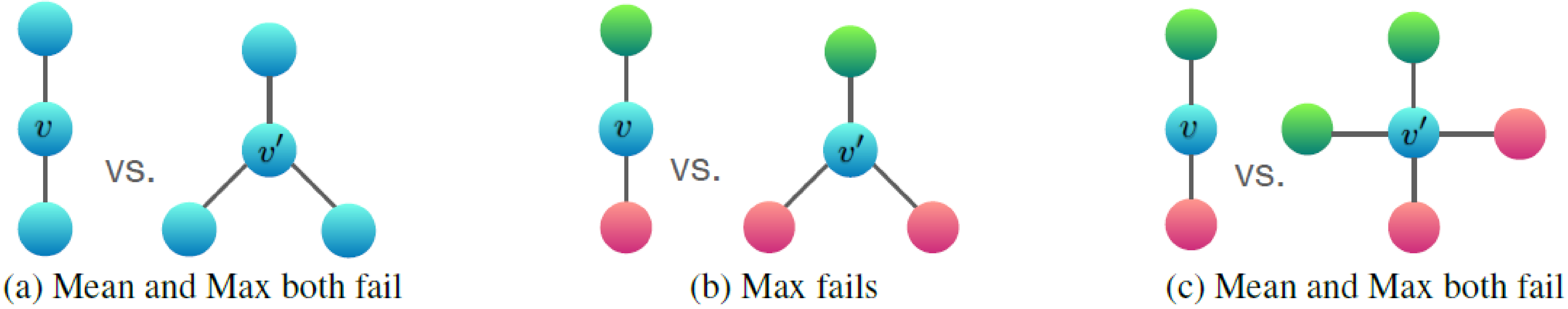
- Based on Theorems, we can:
  - Use sum pooling as **Aggregator & Readout**
  - Use MLP as **Combiner**

- Generally, there may exist many other powerful GNNs. GIN is the one being simple.

Choice of Aggregator

- Choose between **sum, mean and max** pooling.

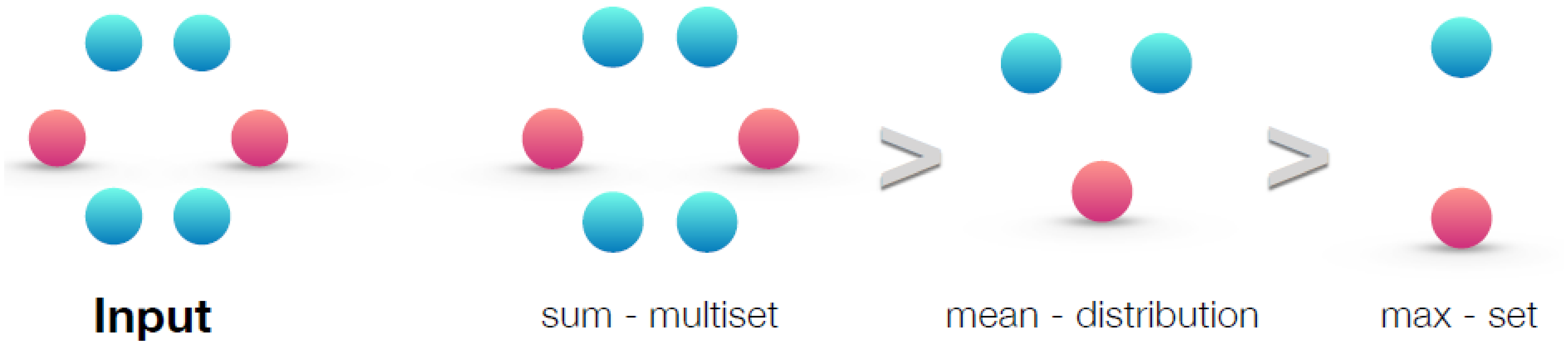
Blue as 20
Green as 10
Red as 5



Examples of graph structures that mean and max aggregators fail to distinguish. Between the two graphs, nodes v and v' get the same embedding even though their corresponding graph structures differ.

Choice of Aggregator

- **Sum**: captures the full multiset
- **Mean**: captures the proportion/distribution of elements of a given type
- **Max**: reduces the multiset to a simple set



Choice of Combiner

- **1-layer perceptrons are not sufficient**

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

So the GNN layers degenerate into simply summing over neighborhood features.

- GIN **use 2-layer MLP** instead.

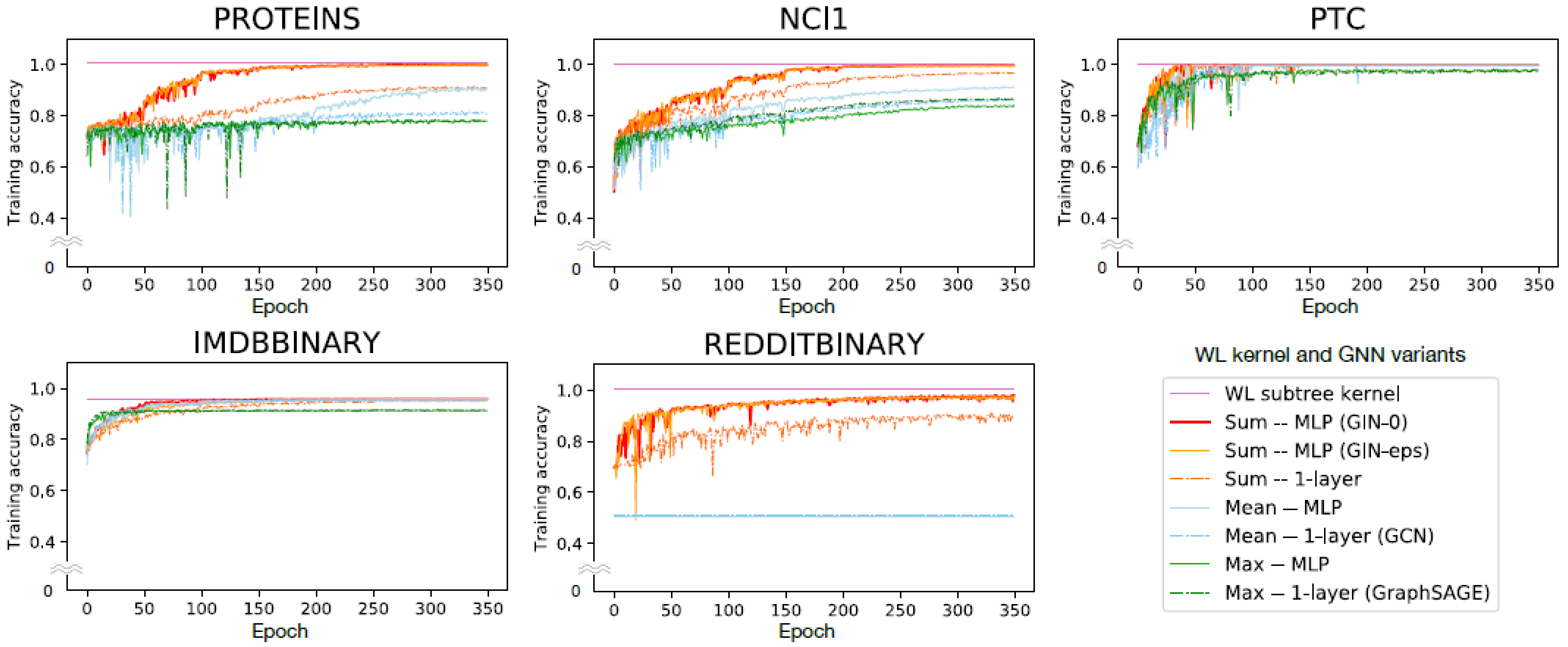
Dataset

- **9 graph classification benchmarks**
  - 4 bioinformatics datasets
    - MUTAG, PTC, NCI1, PROTEINS
  - 5 social network datasets
    - COLLAB, IMDB-BINARY, IMDB-MULTI, REDDITBINARY and REDDIT-MULTI5K

- Remove node features, force GNNs mainly learn from graph structures.

Experiment

- Expressive power demonstrated by **training accuracy**



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	**86.0 ± 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	**92.6 ± 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)	**75.1 ± 5.1**	**52.3 ± 2.8**	**92.4 ± 2.5**	**57.5 ± 1.5**	**80.2 ± 1.9**	**89.4 ± 5.6**	**76.2 ± 2.8**	**64.6 ± 7.0**	**82.7 ± 1.7**
	SUM–MLP (GIN-ϵ)	**74.3 ± 5.1**	**52.1 ± 3.6**	**92.2 ± 2.3**	**57.0 ± 1.7**	**80.1 ± 1.9**	**89.0 ± 6.0**	**75.9 ± 3.8**	63.7 ± 8.2	**82.7 ± 1.6**
	SUM–1-LAYER	74.1 ± 5.0	**52.2 ± 2.4**	90.0 ± 2.7	55.1 ± 1.6	**80.6 ± 1.9**	**90.0 ± 8.8**	**76.2 ± 2.6**	63.1 ± 5.7	82.0 ± 1.5
	MEAN–MLP	73.7 ± 3.7	**52.3 ± 3.1**	50.0 ± 0.0	20.0 ± 0.0	79.2 ± 2.3	83.5 ± 6.3	75.5 ± 3.4	**66.6 ± 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

Test set classification accuracies (%).

Summary

- The most powerful GNNs are **as powerful as** the WL test.
- **Powerful GNNs** have **injective** aggregation and graph readout.
- **GIN** is maximally powerful GNN. Key is to use **sum and MLP**.

Thank You!

Q & A