How Powerful Are Graph Neural Networks

Keyulu Xu, Weihua Hu, Jure Leskovec, Stefanie Jegelka

Presented by Mikhail Mishin[1]

[1]Aalto University
mikhail.mishin@aalto.fi

2019

Agenda

- Intro to Graph Neural Networks (GNNs)
- Theoretical framework: Characterize GNN's discriminative power
- Propose a Maximally Powerful GNN
- Experiments
- Summary & Conclusion
- Demo & Discussion

Setup

Assume we have a graph $G = (V, E)$

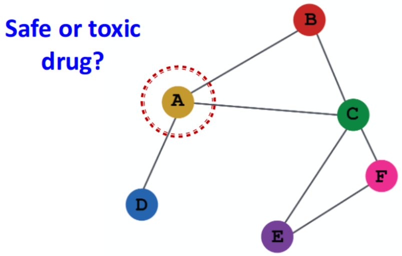
- $N(v)$ is the neighborhood of node v (the set of nodes adjacent to v)
- X_v is the input feature vector of node v
  - Social networks: user profile, user image
  - Biological networks: gene expression profile
  - No features: node degree, constant
- Assume the node input features are from a countable universe
- Can assign each feature vector a unique label in $\{a, b, c, \ldots\}$
- Feature vectors of a set of neighbouring nodes form a multiset (allows multiple instances)

Supervised Learning on Graphs

Node classification

- Given a label y_v for each node $v \in V$
- Learn an embedding h_v of v
- To help predict v's label, $y_v = f(h_v)$

Example: a drug interaction network

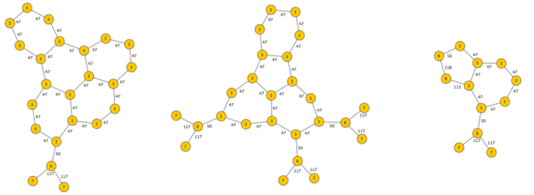


Supervised Learning on Graphs

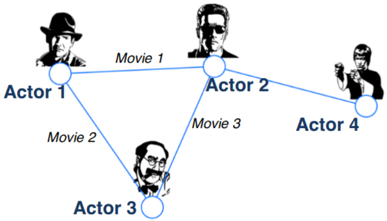
Graph classification

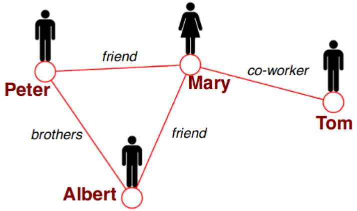
- Given a set of graphs $\{G_1, \ldots, G_N\} \subseteq \mathcal{G}$
- And their labels $\{y_1, \ldots, y_N\} \subseteq \mathcal{Y}$
- Learn an embedding h_G of a graph G
- To help predict G's label, $y_G = g(h_G)$

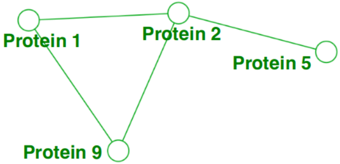
Example: aromatic and heteroaromatic nitro compounds, MUTAG

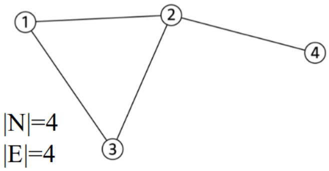


More Graph Examples



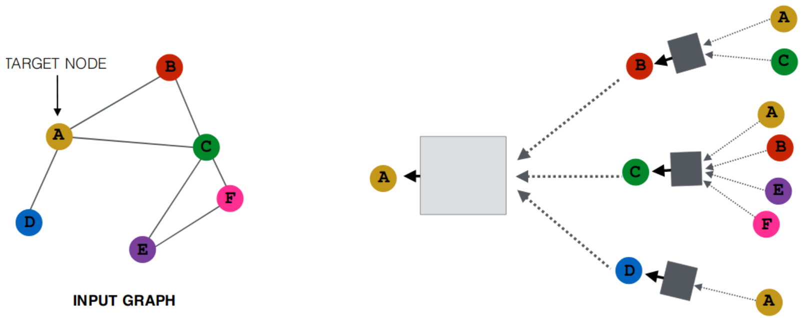






Graph Neural Networks[1]

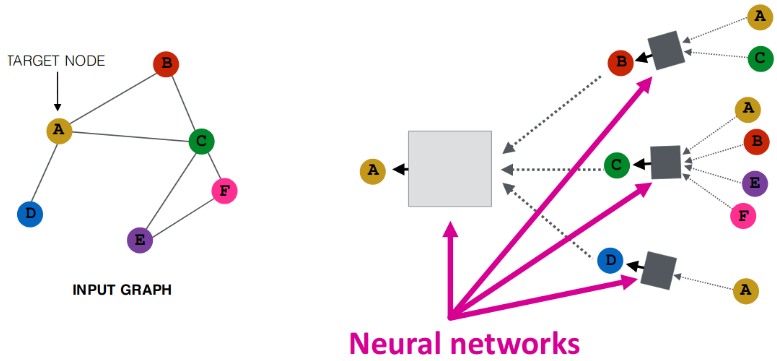
Key idea: generate node embeddings based on local network neighborhoods



[1] Leskovec, *Stanford CS224W: Machine Learning with Graphs*.

Graph Neural Networks[2]

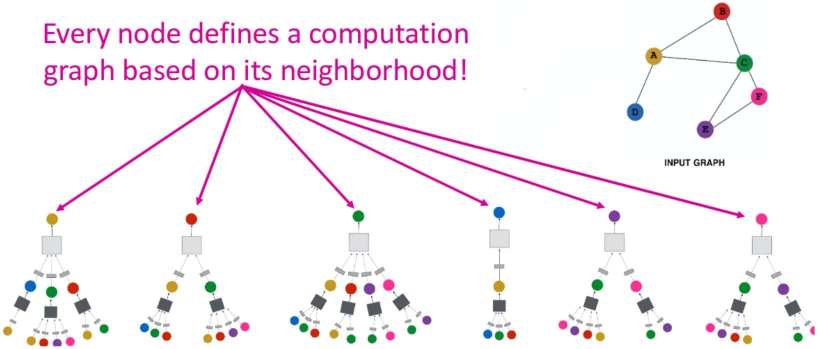
Intuition: nodes aggregate information from their neighbors using neural networks



[2]Leskovec, *Stanford CS224W: Machine Learning with Graphs*.

Graph Neural Networks[3]

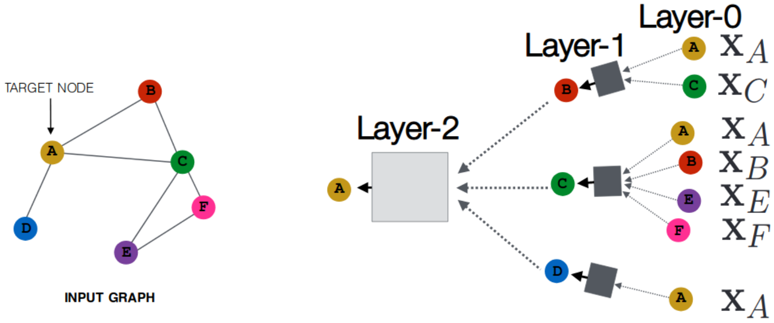
Intuition: network neighborhood defines a computational graph


Every node defines a computation graph based on its neighborhood!
B
A
C
F
D
E
INPUT GRAPH

[3]Leskovec, *Stanford CS224W: Machine Learning with Graphs.*

Graph Neural Networks[4]

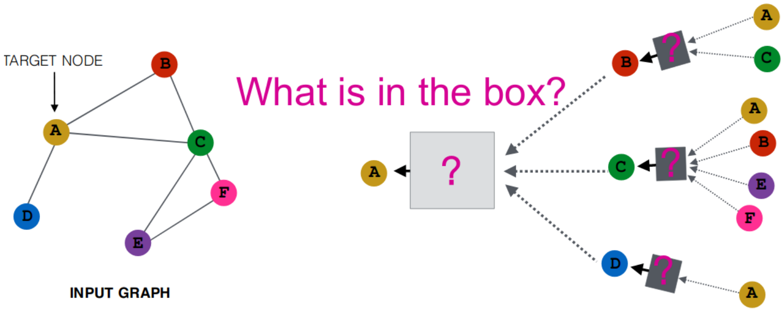
- Model can be of arbitrary depth
- Nodes have embeddings at each layer
- Layer-0 embedding of node v is its input feature vector X_v
- Layer-k embedding gets information that's k hops away



[4]Leskovec, *Stanford CS224W: Machine Learning with Graphs.*

Neighborhood Aggregation[5]

- Key distinctions are in how different approaches aggregate information across layers
- The same aggregation parameters are shared for each layer



[5]Leskovec, *Stanford CS224W: Machine Learning with Graphs.*

Neighborhood Aggregation

- Formally, the *k*-th layer of a GNN is:

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

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

- $h_v^{(k)}$ is the embedding of node v at the k-th iteration/layer.
- $h_v^{(0)} = X_v$, v's input feature vector

GNN Architectures: GraphSAGE[6]

- AGGREGATE is formulated as:

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

- COMBINE is formulated as:

$$h_v^{(k)} = W^{(k)} \cdot \text{CONCAT}\left[h_v^{(k-1)}, a_v^{(k)}\right]$$

- MAX represents the element-wise max pooling operation
- $W_{\text{pool}}^{(k)}$, $W^{(k)}$ are trainable matrices

[6] Hamilton, Ying, and Leskovec, "Inductive Representation Learning on Large Graphs".

GNN Architectures: Graph Convolutional Network (GCN)[7]

- AGGREGATE and COMBINE are formulated as:

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

- MEAN represents the element-wise mean pooling operation
- $W^{(k)}$ is a trainable matrix

[7] Kipf and Welling, "Semi-Supervised Classification with Graph Convolutional Networks".

Supervised Training[8]

Node classification: use the embedding $h_v^{(K)}$ of the final iteration for prediction

$$\mathcal{L} = \sum_{v \in V} \boxed{y_v} \log(\sigma(\boxed{\mathbf{z}_v^\top} \boxed{\boldsymbol{\theta}})) + (1 - \boxed{y_v}) \log(1 - \sigma(\boxed{\mathbf{z}_v^\top} \boxed{\boldsymbol{\theta}}))$$

Encoder output:
node embedding

Classification
weights

Node class
label

Safe or toxic drug?

[8] Leskovec, *Stanford CS224W: Machine Learning with Graphs.*

Supervised Training

Graph classification:

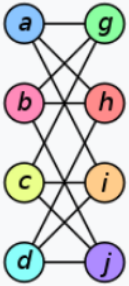
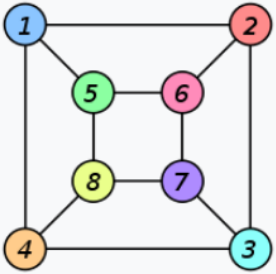
- Aggregate node embeddings from the final iteration with a READOUT function:

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

- E.g. READOUT = SUM or READOUT = MEAN
- Can be a more sophisticated pooling function
- To train use h_G for prediction, same as for the node classification task

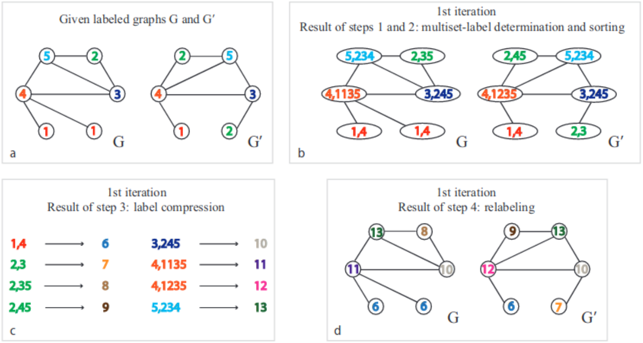
Graph Isomorphism Problem

- Are two graphs topologically identical?
- Known to be in NP, no polynomial-time algorithm
- GNN is able to map two different graphs to different embeddings $\implies$ it solves GI
- Might need a weaker criterion...

Graph G	**Graph H**	**An isomorphism between G and H**
		$f(a) = 1$ $f(b) = 6$ $f(c) = 8$ $f(d) = 3$ $f(g) = 5$ $f(h) = 2$ $f(i) = 4$ $f(j) = 7$

Weisfeiler-Lehman Test of Graph Isomorphism

- The 1-dimensional WL test iteratively:
  - Aggregates the labels of nodes and their neighborhoods
  - Hashes the aggregated labels into unique new labels

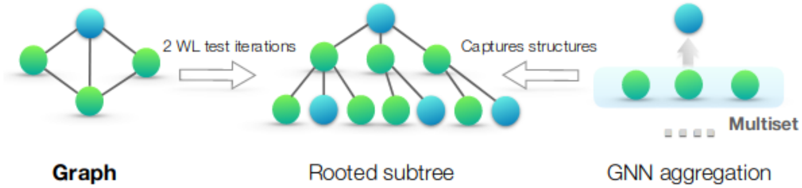

Given labeled graphs G and G′
G
G′
a
1st iteration
Result of steps 1 and 2: multiset-label determination and sorting
5,234
2,35
4,1135
3,245
1,4
2,45
4,1235
2,3
b
1st iteration
Result of step 3: label compression
c
1st iteration
Result of step 4: relabeling
d

1,4	6	3,245	10
2,3	7	4,1135	11
2,35	8	4,1235	12
2,45	9	5,234	13

- Decide two graphs are non-isomorphic if at some iteration the multisets of labels between the two graphs differ

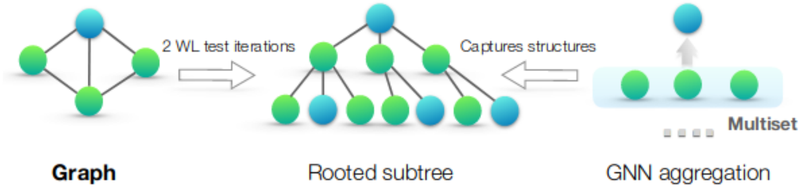
GNNs are At Most As Powerful As WL

- Key contribution 1: GNNs are at most as powerful as the WL test in distinguishing graph structures.
- GNN maps G_1 and G_2 to different embeddings $\Longrightarrow$ the WL test decides G_1 and G_2 are non-isomorphic
- See the formal proof is in the paper (Lemma 2)



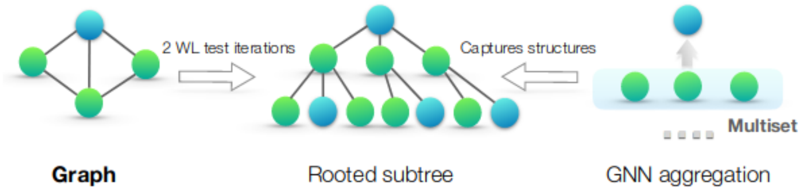
Maximally Powerful GNNs (MP-GNNs)

- A Maximally Powerful GNN would never map two different node neighborhoods to the same representation
- Its AGGREGATE, COMBINE, READOUT must be injective



MP-GNNs are As Powerful As WL

- Key contribution 2: MP-GNNs are as powerful as the WL test in distinguishing graph structures.
- The WL test decides G_1 and G_2 are non-isomorphic $\Longrightarrow$ a MP-GNN maps G_1 and G_2 to different embeddings
- Note, the proof only holds for countable sets of input node features
- See the formal proof in the paper (Theorem 3 and Lemma 4)



GNNs Learn to Embed

- WL test can only discriminate different graph structures
- GNNs learn useful node representations, capturing similarity of graph structures
- WL test provides theoretical context for GNN design

GraphSAGE is Not Maximally Powerful

- AGGREGATE is formulated as:

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

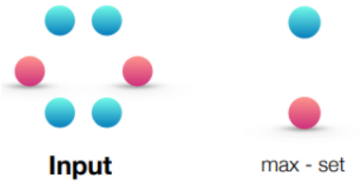
- COMBINE is formulated as:

$$h_v^{(k)} = W^{(k)} \cdot \text{CONCAT}\left[h_v^{(k-1)}, a_v^{(k)}\right]$$

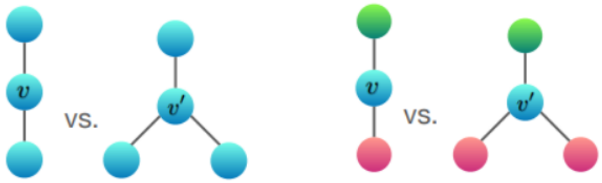
- MAX is not injective

Max-Pooling Learns Sets with Distinct Elements

- MAX aggregator ignores multiplicities



- MAX fails



GCN is Not Maximally Powerful

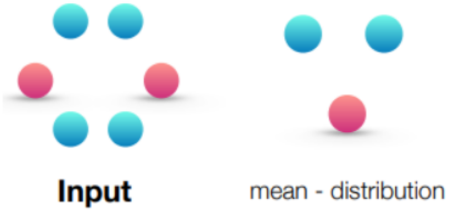
- AGGREGATE and COMBINE are formulated as:

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

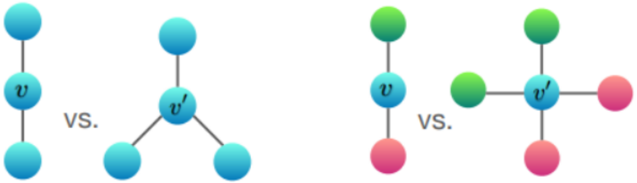
- MEAN is not injective

Mean-Pooling Learns Distributions

- MEAN captures the proportion of elements of a given type

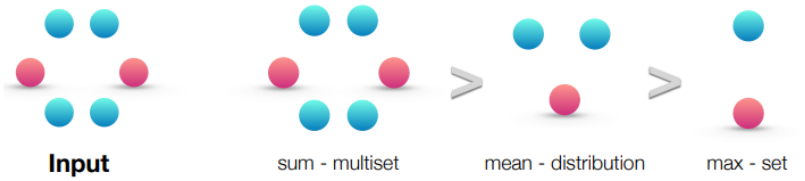


- MEAN fails

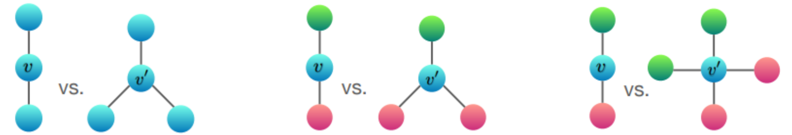


Sum is Injective

- Key contribution 3: Ranking by representational power
- SUM captures the whole multiset



- SUM succeeds



(a) Mean and Max both fail (b) Max fails (c) Mean and Max both fail

Graph Isomorphism Network (GIN)

- Key contribution 4: GIN is provably maximally powerful
- AGGREGATE and COMBINE are formulated as:

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

- MLP is Multi-Layer Perceptron
- ϵ is needed to distinguish the root/central node
- 1-layer perceptron is not enough
- See Lemma 5 and Corollary 6

Graph-Level READOUT of GIN

- Concatenate graph representations across all layers of GIN

$$h_G = \text{CONCAT}\left(\text{READOUT}\left(\left\{h_v^{(k)} | v \in V\right\}\right) | k = 0, 1, \dots K\right)$$

- Node embeddings get more refined and global as the number layers increases
- Earlier layers may generalize better
- If READOUT=SUM, provably maximally powerful
- However READOUT=MEAN, performs better on some datasets

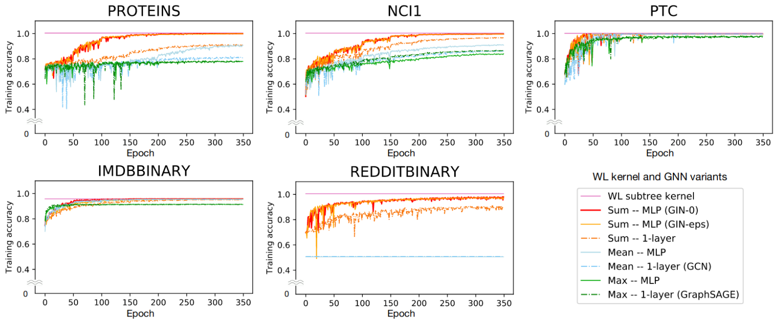
Experiments: Datasets

- The goal is to learn from the network structure, not to rely on input features
- Bioinformatic datasets: MUTAG, PRC, NCI1, PROTEINS
  - Graphs represent chemical compounds
  - MUTAG: aromatic and heteroaromatic nitro compounds
  - Nodes have categorial input features
- Social network datasets: COLLAB, IMDB-BINARY, IMDB-MULTI, REDDIT-BINARY, REDDIT-MULTI5K
  - Graphs represent social communities
  - IMDB: ego-network for each actor/actress, classify genre
  - REDDIT: online discussion, classify subreddit
  - Node input features: IMDB - node degrees, REDDIT - constant

Experiments: Models and Configurations

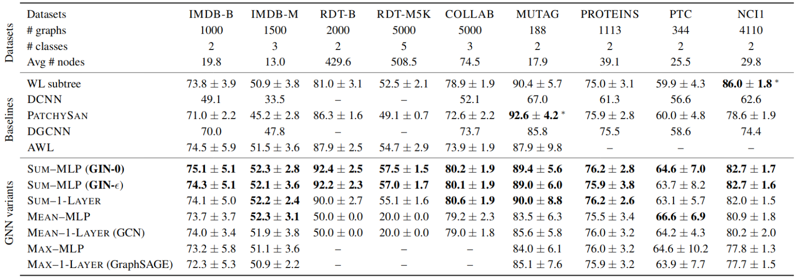
- Evaluate GIN:
  - GIN-ϵ with learning ϵ
  - GIN-0 without learning ϵ
- Evaluate less powerful versions:
  - Replace SUM with MEAN or MAX
  - Replace MLP with 1-layer perceptrons
- Apply the same graph-level READOUT:
  - SUM on bioinformatics datasets
  - MEAN on social datasets (better perfomance on test)
- For all configurations, apply 5 GNN layers (including input)
- All MLPs have 2 layers
- Apply Batch normalization on hidden layers
- Compare with state-of-the-art baselines: PATCHY-SAN, Deep Graph CNN, etc.

Experiments: Training Performance



- GIN-ϵ and GIN-0 almost perfectly fit training
- Others slightly underfit
- Observed training accuracy aligns with representational power
- WL subtree kernel has the best training accuracies

Experiments: Test 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

- GINs outperform less powerful GNN varians
- GIN-0 slightly outperforms GIN-ϵ
- GINs shine on social network datasets
- MEAN GNNs fail to capture any structures of the unlabeled datasets (RDT-B, RDT-M5K)

TL;DR:

- POWER - ability to discriminate graph structures
- Prove POWER(GNNs) $\leq$ POWER(WL)
- Establish conditions for POWER(GNNs) $=$ POWER(WL)
- Show POWER(GraphSAGE) $<$ POWER(WL) and POWER(GCN) $<$ POWER(WL)
- Propose GIN and prove POWER(GIN) $=$ POWER(WL)
- Empirically show GIN performs better than GraphSAGE and GCN

Conclusion[9]

Graph Convolutional Neural Networks:

- Representation learning paradigm can be extended to graphs
- Can effectively combine node attribute data with the network information
- State-of-the-art results in a number of domains/tasks
- Use end-to-end training instead of multi-stage approaches for better performance

[9] Leskovec, *Stanford CS224W: Machine Learning with Graphs*.