The Expressive Power of Graph Neural Networks from Three Perspectives

Zhengdao Chen

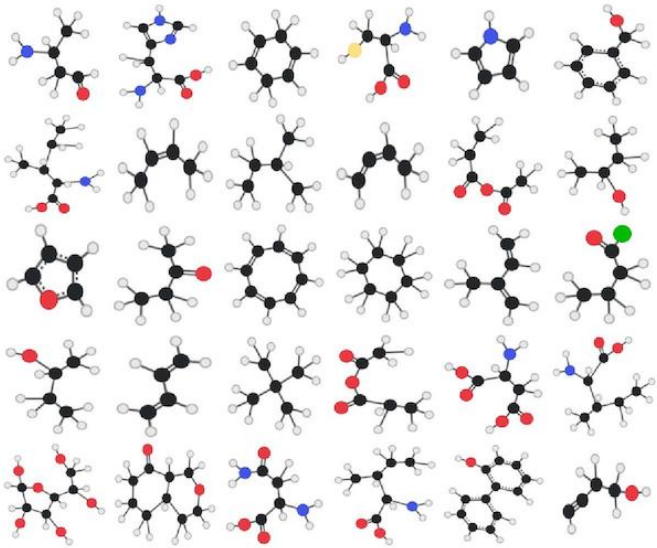
Courant Institute of Mathematical Sciences

SIAM MDS mini-symposium, June 2020

Table of Contents

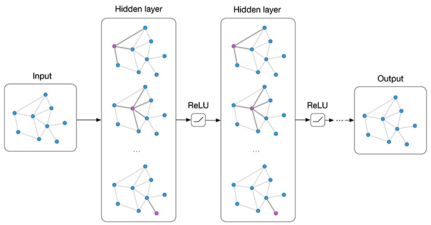
1. Introduction
2. Perspective I: Graph Isomorphism Testing
3. Perspective II: Function Approximation
4. Perspective III: Substructure Counting
5. Case studies
   - Message Passing Neural Networks
   - Invariant Graph Networks
6. Conclusions

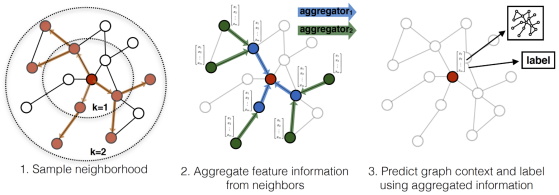
- molecular chemistry
- social networks
- knowledge graphs
- particle physics
- combinatorial optimization

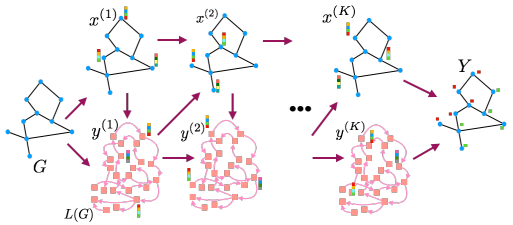


[1]

[1] https://github.com/chemplexity/molecules







2

Advantages of (many) GNN architectures:

- Built-in permutation-invariance
- Efficiency of computation
- Can be applied to graphs of different sizes

Question: Have we sacrificed expressive power to gain such advantages? How should we study the expressive power of GNNs?

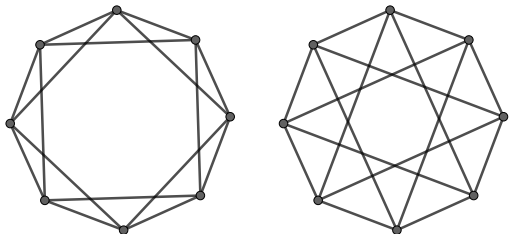
[2] Kipf and Welling (2016); Hamilton et al. (2017); Chen et al. (2019a)

Table of Contents

1. Introduction
2. Perspective I: Graph Isomorphism Testing
3. Perspective II: Function Approximation
4. Perspective III: Substructure Counting
5. Case studies
   - Message Passing Neural Networks
   - Invariant Graph Networks
6. Conclusions

Question: Can GNNs distinguish any pair of non-isomorphic graphs?

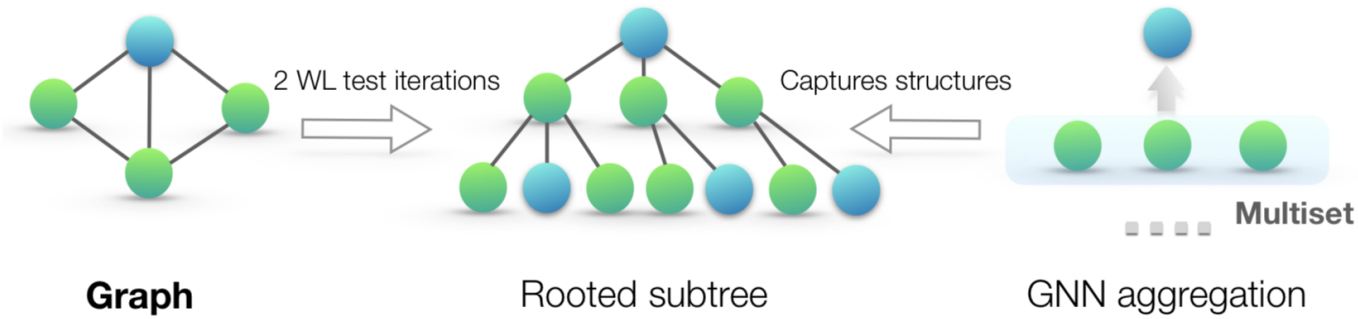
Answer: No, for GNNs based upon message-passing. For example,



Theorem (Xu et al. (2018); Morris et al. (2019))

If two graphs cannot be distinguished by the WL test, then they cannot be distinguished by any GNN based on message-passing.

WL test compares the rooted subtree substructures of the graphs.



[3]

Xu et al. (2018) and Morris et al. (2019) also propose GNN models that are as powerful as the WL test.

[3] Xu et al. (2018)

Problems:

- WL can distinguish almost all pairs of non-isomorphic graphs (Babai et al., 1980).

  Does this mean that the existing GNNs are already powerful enough?

- Is graph isomorphism testing necessarily relevant to real-world tasks on graph datasets?

Table of Contents

1. Introduction
2. Perspective I: Graph Isomorphism Testing
3. Perspective II: Function Approximation
4. Perspective III: Substructure Counting
5. Case studies
   - Message Passing Neural Networks
   - Invariant Graph Networks
6. Conclusions

GNNs Represent Functions on Graphs

A GNN architecture defines a family of functions on graphs.

Training a GNN for a graph-level prediction task
$\approx$ fitting an underlying target function.

Question: Can certain families of graph neural networks approximate "arbitrary" functions on graphs?

Let $\mathcal{G}$ be a finite space of graphs, and $\mathcal{F}$ a set of functions $\mathcal{G} \to \mathbb{R}$.

Definition

We say $\mathcal{F}$ is **universally approximating** if $\forall$ function $g : \mathcal{G} \to \mathbb{R}$ and $\forall \epsilon > 0$, $\exists f \in \mathcal{F}$ such that $\sup_{G \in \mathcal{G}} |g(G) - f(G)| < \epsilon$.

Definition

We say $\mathcal{F}$ is **GISO-discriminating** if $\forall G_1, G_2 \in \mathcal{G}$ that are not isomorphic, $\exists f \in \mathcal{F}$ such that $f(G_1) \neq f(G_2)$.

Proposition (Chen et al. (2019b, 2020))

- *If $\mathcal{F}$ is universally approximating on $\mathcal{G}$, then it is also GISO-discriminating on $\mathcal{G}$;*
- *If $\mathcal{F}$ is GISO-discriminating on $\mathcal{G}$, then $\mathcal{F}^{+1}$ is universally approximating on $\mathcal{G}$.*

$\mathcal{F}^{+1}$ consists of functions in $\mathcal{F}$ augmented with one extra layer.

Corollary

GNNs based on message-passing are not universally approximating.

Remaining question: What functions can and cannot GNNs approximate, and how to understand them intuitively?

We want a criterion for the expressive power of GNNs that is:

- Intuitive and concrete
- Relevant for applications
- Helpful in guiding the search for more powerful GNNs

Some other interesting perspectives: Turing universality (Loukas, 2019); ability to decide several graph properties (Garg et al., 2020)

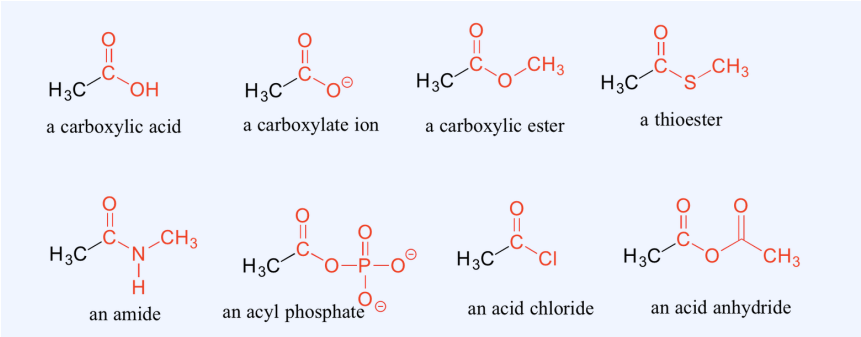
Our answer: **substructure counting**.

Table of Contents

1. Introduction
2. Perspective I: Graph Isomorphism Testing
3. Perspective II: Function Approximation
4. Perspective III: Substructure Counting
5. Case studies
   - Message Passing Neural Networks
   - Invariant Graph Networks
6. Conclusions

Why is Substructure Counting Relevant?

- For example, functional groups in organic chemistry[4]

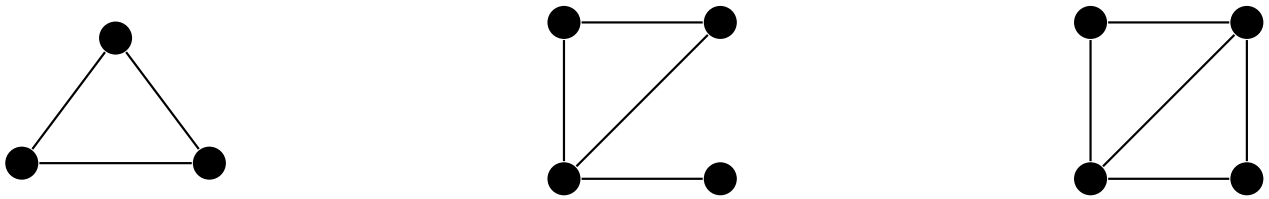


- Computational biology and social network studies
- Generating molecular fingerprints, computing molecular similarities, computing graph kernels...
- Interpreting GNNs' predictions via substructures (Ying et al., 2019)

[4] https://chem.libretexts.org/Courses/Sacramento_City_College

Given two graphs $G_A = (V_A, E_A)$ and $G_B = (V_B, E_B)$:

- G_A is a **subgraph** of G_B if $V_A \subseteq V_B$ and $E_A \subseteq E_B$.
- G_A is an **induced subgraph** of G_B if $V_A \subseteq V_B$ and $E_A = E_B \cap (V_A \times V_A)$.



Can be similarly defined for graphs with node and edge attributes.

Subgraph-count and induced-subgraph-count

Let G be a graph, and $G^{[\mathrm{P}]}$ be a pattern we are interested in counting.

The **subgraph-count** of $G^{[\mathrm{P}]}$ in G, denoted by $\mathrm{C}_S(G; G^{[\mathrm{P}]})$, is the number of *subgraphs* of G that are isomorphic to $G^{[\mathrm{P}]}$.

The **induced-subgraph-count** of $G^{[\mathrm{P}]}$ in G, denoted by $\mathrm{C}_I(G; G^{[\mathrm{P}]})$, is the number of *induced subgraphs* of G that are isomorphic to $G^{[\mathrm{P}]}$.

What does it mean for GNNs to count substructures?

Say $\mathcal{F}$ is a family of GNNs, $G^{[\mathrm{P}]}$ is a given pattern.

- Defined via function approximation

  Can $\mathcal{F}$ approximate $\mathrm{C}_S(\cdot; G^{[\mathrm{P}]})$ or $\mathrm{C}_I(\cdot; G^{[\mathrm{P}]})$ as functions on the graph space arbitrarily well?

- Defined via graph isomorphism testing

  Given any two graphs, $G^{[1]}$ and $G^{[2]}$ with different (induced-)subgraph-counts of a pattern $G^{[\mathrm{P}]}$, can we always find a function in $\mathcal{F}$ that distinguishes them?

Equivalence between the two definitions above. ✓

Table of Contents

1. Introduction
2. Perspective I: Graph Isomorphism Testing
3. Perspective II: Function Approximation
4. Perspective III: Substructure Counting
5. Case studies
   - Message Passing Neural Networks
   - Invariant Graph Networks
6. Conclusions

Table of Contents

1. Introduction
2. Perspective I: Graph Isomorphism Testing
3. Perspective II: Function Approximation
4. Perspective III: Substructure Counting
5. Case studies
   - Message Passing Neural Networks
   - Invariant Graph Networks
6. Conclusions

Initially, set $h_i^{(0)} = x_i$ for each node $i \in V$.

For $t \in \{0, ..., T-1\}$,

$$m_i^{(t+1)} = \sum_{\mathcal{N}(i)} M_t(h_i^{(t)}, h_j^{(t)}, e_{i,j})$$

$$h_i^{(t+1)} = U_t(h_i^{(t)}, m_i^{(t+1)})$$

Finally,

$$\hat{y} = R(\{h_i^{(T)} : i \in V\}),$$

Trainable parameters in each M_t, U_t and R.

Question: For what patterns can MPNNs perform (induced-)subgraph-count?

Answer completely determined by the size of the pattern, assuming that the pattern of interest is connected.

- Pattern with 1 node
  (i.e., to count the number of nodes with a given node feature x^*)

  Can be solved by an MPNN with 0 hidden layer and

$$R(\{h_i^{(0)} : i \in V\}) = \sum_{i \in V} \mathbb{1}(h_i^{(0)} = x^*)$$

- Pattern with 2 nodes
  (i.e. to count the number of edges with a given edge feature e^*)

  Can be solved by an MPNN with 1 hidden layer and

$$M_0(h_i^{(0)}, h_j^{(0)}, e_{i,j}) = \mathbb{1}(e_{i,j} = e^*)$$
$$U_0(h_i^{(0)}, m_i^{(1)}) = m_i^{(1)}$$
$$R(\{h_i^{(1)} : i \in V\}) = \frac{1}{2} \sum_{i \in V} h_i^{(1)}$$

- What about patterns consisting of 3 nodes or more?

Theorem (Chen et al. (2020))

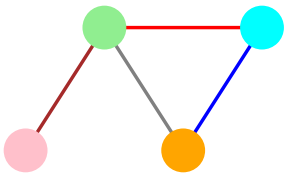
MPNNs **cannot** *perform* **induced-subgraph-count** *of any connected pattern consisting of 3 or more nodes.*

Proof strategy:

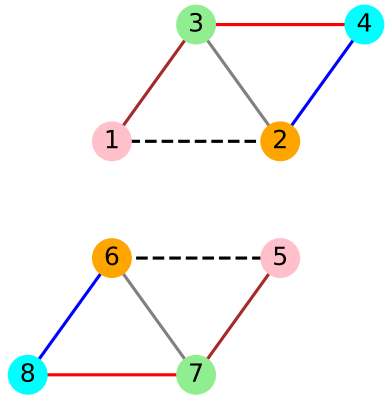
Construct a pair of graphs that have the different induced-subgraph-counts but cannot be distinguished by any MPNN.

Counter-example pair of graphs

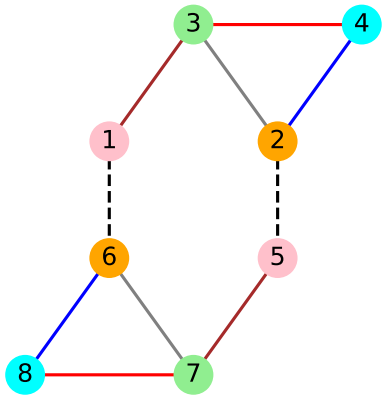
For example,



$G^{[P]}$



$G^{[1]}$



$G^{[2]}$

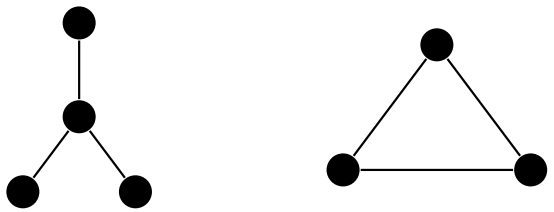
On one hand, $\mathsf{C}_I(G^{[1]}; G^{[P]}) = 0$ whereas $\mathsf{C}_I(G^{[2]}; G^{[P]}) = 2$.

On the other hand, no MPNN can distinguish $G^{[1]}$ from $G^{[2]}$.

For subgraph-count, we have the following positive result:

Theorem (5, Chen et al. (2020))

2-WL and MPNNs **can** *perform* **subgraph-count** *of star-shaped patterns.*



This is an extension of a result in Arvind et al. (2018) on WL to the cases that include node and edge features.

GNNs based on higher-order message-passing have been proposed, achieving k-WL-level of power for $k > 1$ (Morris et al., 2019).

Theorem (7, Chen et al. (2020))

k-WL, at initialization, **can** *perform both* **induced-subgraph-count** *and* **subgraph-count** *of patterns consisting of at most k nodes.*

Theorem (9, Chen et al. (2020))

Running T iterations of k-WL **cannot** *perform* **induced-subgraph-count** *of any path pattern of $(k+1)2^T$ or more nodes.*

Table of Contents

1. Introduction
2. Perspective I: Graph Isomorphism Testing
3. Perspective II: Function Approximation
4. Perspective III: Substructure Counting
5. Case studies
   - Message Passing Neural Networks
   - Invariant Graph Networks
6. Conclusions

Definition (Maron et al. (2018, 2019b))

A kth-order Invariant Graph Network (k-IGN) is a function $F : \mathbb{R}^{n^k \times d_0} \to \mathbb{R}$ that can be decomposed in the following way:

$$F = m \circ h \circ L^{(T)} \circ \sigma \circ \cdots \circ \sigma \circ L^{(1)},$$

where each $L^{(t)}$ is a linear equivariant layer from $\mathbb{R}^{n^k \times d_{t-1}}$ to $\mathbb{R}^{n^k \times d_t}$, σ is a pointwise activation function, h is a linear invariant layer from $\mathbb{R}^{n^k \times d_T}$ to $\mathbb{R}$, and m is an MLP.

Theorem (Maron et al. (2019a))

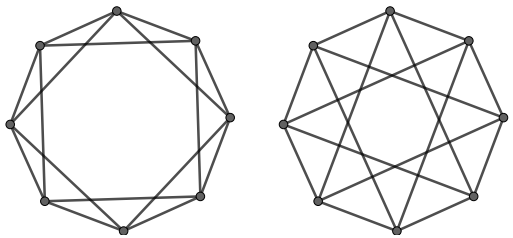
k-IGNs are no less powerful than k-WL.

Theorem (Maron et al. (2019b))

For graphs whose sizes are bounded above, k-IGNs are universally approximating if k is large enough.

Theorem (7, Chen et al. (2019b))

2-IGNs cannot distinguish between non-isomorphic regular graphs with the same degree.



Theorem (11, Chen et al. (2020))

If two graphs cannot be distinguished by 2-WL, then they cannot be distinguished by any 2-IGN.

Corollary

2-IGNs are exactly as powerful as 2-WL.

Corollary

2-IGNs **cannot** *perform* **induced-subgraph-count** *of any connected pattern consisting of 3 or more nodes.*

Corollary

2-IGNs **can** *perform* **subgraph-count** *of star-shaped patterns.*

Corollary

k-IGNs, at initialization, **can** *perform both* **induced-subgraph-count** *and containment-count of patterns consisting of at most k nodes.*

Two tasks:

- induced-subgraph-count of triangles
- subgraph-count of 3-stars

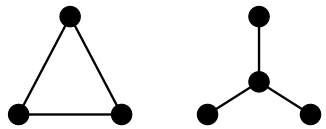


Figure 1: Triangles and 3-stars.

Table 1: Test MSE of different models divided by the variance of ground-truth counts.

	Erdős-Renyi				Random Regular			
	Triangle (M)		3-Star (C)		Triangle (M)		3-Star (C)	
	top 1	top 3	top 1	top 3	top 1	top 3	top 1	top 3
LRP-1-4	1.56E-4	2.49E-4	2.17E-5	5.23E-5	2.47E-4	3.83E-4	1.88E-6	2.81E-6
2-IGN	9.83E-2	9.85E-1	5.40E-4	5.12E-2	2.62E-1	5.96E-1	1.19E-2	3.28E-1
GIN	1.23E-1	1.25E-1	1.62E-4	3.44E-4	4.70E-1	4.74E-1	3.73E-4	4.65E-4
GCN	6.78E-1	8.27E-1	4.36E-1	4.55E-1	1.82	2.05	2.63	2.80
sGNN	9.25E-2	1.13E-1	2.36E-3	7.73E-3	3.92E-1	4.43E-1	2.37E-2	1.41E-1

Observation: To count small (induced)-subgraphs, we only need to look at local neighborhoods.

Definition

An **egonet** of depth l centered at a node i is the induced subgraph consisting of i and all nodes within distance l from it.

Idea: Apply a very expressive (and perhaps expensive) model to each egonet, and then aggregate over all egonets.

Relational Pooling (RP) (Murphy et al., 2019)

One can obtain a *permutation-invariant* function by summing or averaging a *non-permutation-invariant* function over all permutations:

$$f_{\mathrm{RP}}(G) = \sum_{\pi \in S_n} \bar{f}(\pi \circ \boldsymbol{B}(G))$$

This can achieve universal approximation of permutation-invariant functions.

Drawback: Computational complexity is $O(n!)$.

But not as big of an issue if we only apply them to local neighborhoods.

Depth-l LRP (applying RP to every egonet of depth l):

$$f_{\mathrm{LRP}}^{l}(G) = \sum_{i \in V} \sum_{\pi \in S_{n_{i,l}}} \bar{f}\left(\pi \circ \boldsymbol{B}_{i,l}^{[\mathrm{ego}]}\right)$$

Further reducing complexity with size-k cropping and BFS permutations:

$$f_{\mathrm{LRP}}^{l}(G) = \sum_{i \in V} \sum_{\pi \in S_{n_{i,l}}^{\mathrm{BFS}}} \bar{f}\left(C_k(\pi \circ \boldsymbol{B}_{i,l}^{[\mathrm{ego}]})\right)$$

Can also apply LRP iteratively, which yields *Deep LRP*.

Table 2: Results on ogbg-molhiv (measured by ROCAUC)

Model	Training	Validation	**Testing**
Deep LRP-1-4	89.81±2.90	81.31±0.88	76.87±1.80
Deep LRP-1-4 (ES)	87.56±2.11	82.09±1.16	**77.19±1.40**
GIN†	88.64±2.54	82.32±0.90	75.58±1.40
GIN + VN†	92.73±3.80	84.79±0.68	77.07±1.49
GCN†	88.54±2.19	82.04±1.41	76.06±0.97
GCN + VN†	90.07±4.69	83.84±0.91	75.99±1.19
GAT ‡	-	-	72.9±1.8
GraphSAGE ‡	-	-	74.4±0.7

Table 3: Performances on QM9 (measured by MAE)

Target	DTNN	MPNN	123-gnn	Powerful-IGN	Deep LRP-1-4
μ	0.244	0.358	0.476	0.231	0.399
α	0.95	0.89	0.27	0.382	0.337
ϵ_{homo}	0.00388	0.00541	0.00337	0.00276	0.00287
ϵ_{lumo}	0.00512	0.00623	0.00351	0.00287	0.00309
Δ_ϵ	0.0112	0.0066	0.0048	0.00406	**0.00396**
$\langle R^2 \rangle$	17	28.5	22.9	16.07	20.4
ZPVE	0.00172	0.00216	0.00019	0.00064	0.00067
U_0	2.43	2.05	0.0427	0.234	0.590
U	2.43	2	0.111	0.234	0.588
H	2.43	2.02	0.0419	0.229	0.587
G	2.43	2.02	0.0469	0.238	0.591
C_v	0.27	0.42	0.0944	0.184	0.149
Loss	0.1014	0.1108	0.0657	0.0512	0.0641

Table of Contents

1. Introduction
2. Perspective I: Graph Isomorphism Testing
3. Perspective II: Function Approximation
4. Perspective III: Substructure Counting
5. Case studies
   - Message Passing Neural Networks
   - Invariant Graph Networks
6. Conclusions

Three perspectives on the power of GNNs:

- Graph isomorphism testing
- Approximation of permutation-invariant functions
- **Substructure counting**

Case studies on MPNNs and k-IGNs

Proposing LRP for substructure counting and real tasks

Future directions:

- Other GNN architectures
- Explaining real-world tasks with substructure counting (Ying et al., 2019)

References

Arvind, V., Fuhlbrück, F., Köbler, J., and Verbitsky, O. (2018). On weisfeiler-leman invariance: Subgraph counts and related graph properties. arXiv preprint arXiv:1811.04801.

Babai, L., Erdos, P., and Selkow, S. M. (1980). Random graph isomorphism. SIAM Journal on computing, 9(3):628–635.

Chen, Z., Chen, L., Villar, S., and Bruna, J. (2020). Can graph neural networks count substructures? arXiv preprint arXiv:2002.04025.

Chen, Z., Li, L., and Bruna, J. (2019a). Supervised community detection with line graph neural networks. Internation Conference on Learning Representations.

Chen, Z., Villar, S., Chen, L., and Bruna, J. (2019b). On the equivalence between graph isomorphism testing and function approximation with gnns. In Advances in Neural Information Processing Systems, pages 15868–15876.

Garg, V. K., Jegelka, S., and Jaakkola, T. (2020). Generalization and representational limits of graph neural networks.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 1263–1272. JMLR. org.

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In Advances in Neural Information Processing Systems, pages 1024–1034.

References (cont.)

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

Loukas, A. (2019). What graph neural networks cannot learn: depth vs width. arXiv preprint arXiv:1907.03199.

Maron, H., Ben-Hamu, H., Serviansky, H., and Lipman, Y. (2019a). Provably powerful graph networks. In Advances in Neural Information Processing Systems, pages 2153–2164.

Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. (2018). Invariant and equivariant graph networks.

Maron, H., Fetaya, E., Segol, N., and Lipman, Y. (2019b). On the universality of invariant networks. arXiv preprint arXiv:1901.09342.

Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. (2019). Weisfeiler and leman go neural: Higher-order graph neural networks. Association for the Advancement of Artificial Intelligence.

Murphy, R. L., Srinivasan, B., Rao, V., and Ribeiro, B. (2019). Relational pooling for graph representations. arXiv preprint arXiv:1903.02541.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018). How powerful are graph neural networks? arXiv preprint arXiv:1810.00826.

Ying, Z., Bourgeois, D., You, J., Zitnik, M., and Leskovec, J. (2019). Gnnexplainer: Generating explanations for graph neural networks. In Advances in neural information processing systems, pages 9244–9255.

Thanks!



Can graph neural networks count substructures?
Z. Chen, L. Chen, S. Villar, J. Bruna. Preprint, 2020

On the equivalence between graph isomorphism testing and function approximation with GNNs
Z. Chen, S. Villar, L. Chen, J. Bruna. NeurIPS 2019

Supervised community detection with line graph neural networks
Z. Chen, L. Li, J. Bruna. ICLR 2019