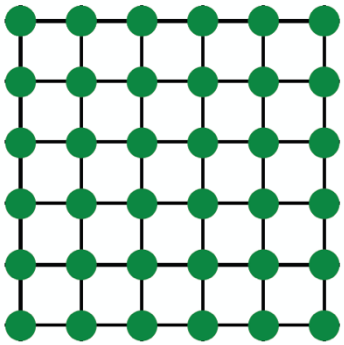
Advancements in Graph Neural Networks:
PGNNs, Pretraining, and OGB

Jure Leskovec

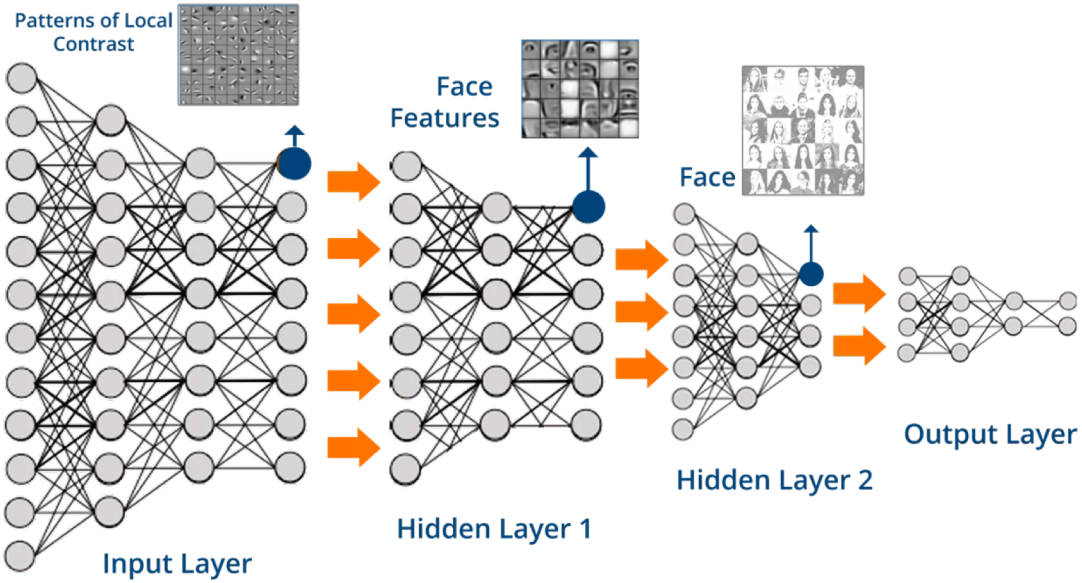
CHAN ZUCKERBERG BIOHUB

Includes joint work with W. Hu, J. You, M. Fey, Y. Dong, B. Liu, M. Catasta, K. Xu, S. Jegelka, M. Zitnik, P. Liang, V. Pande

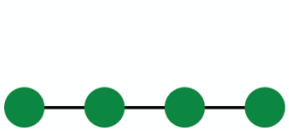
Modern ML Toolbox

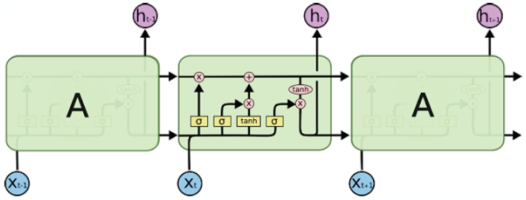


Images



Text/Speech





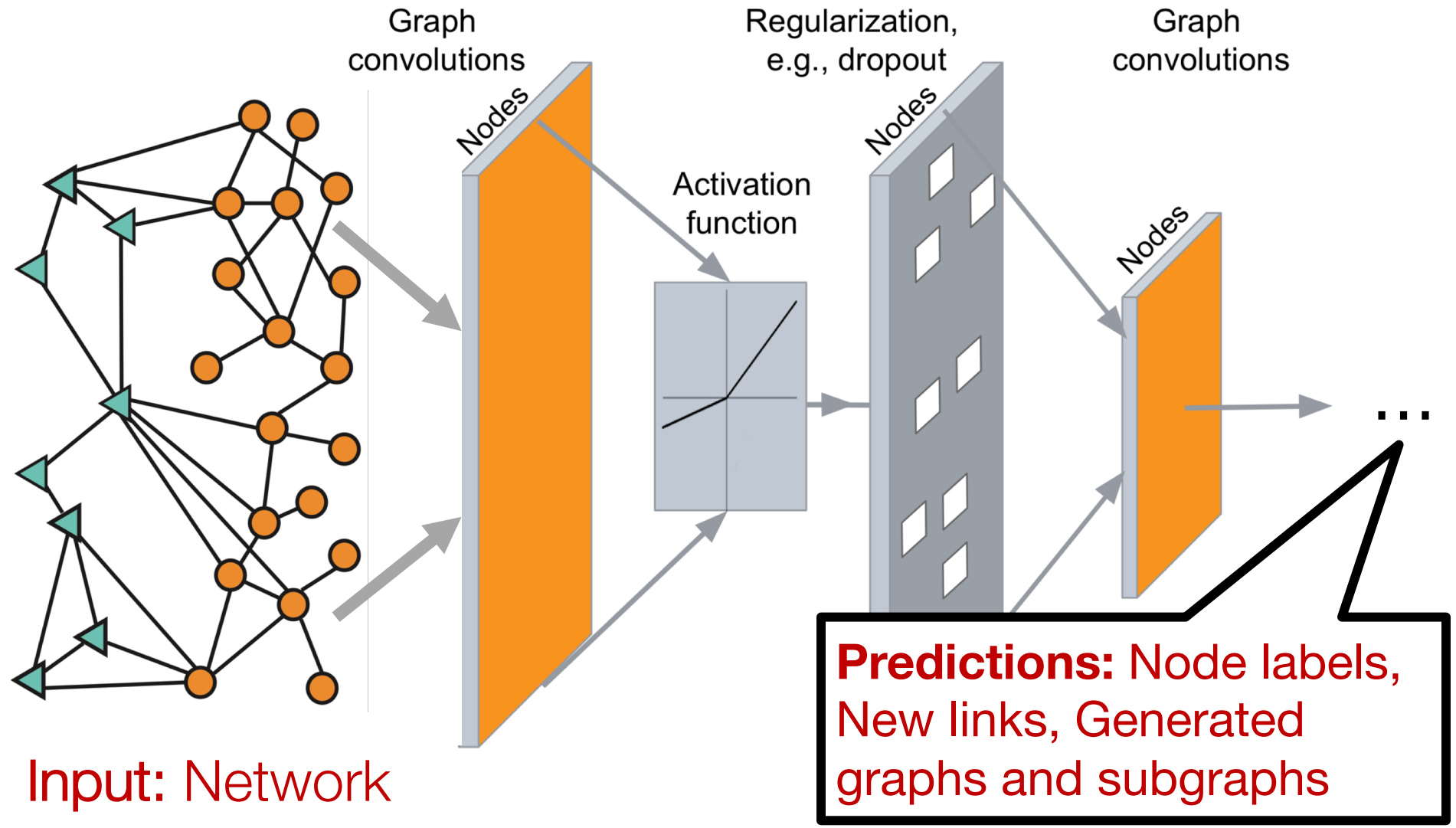
Modern deep learning toolbox is designed for simple sequences & grids

But not everything can be represented as a sequence or a grid

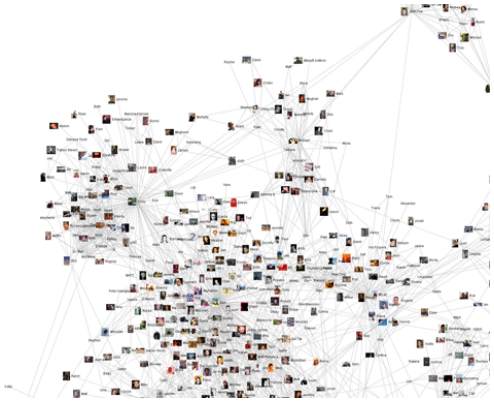
How can we develop neural networks that are much more broadly applicable?

New frontiers beyond classic neural networks that learn on images and sequences

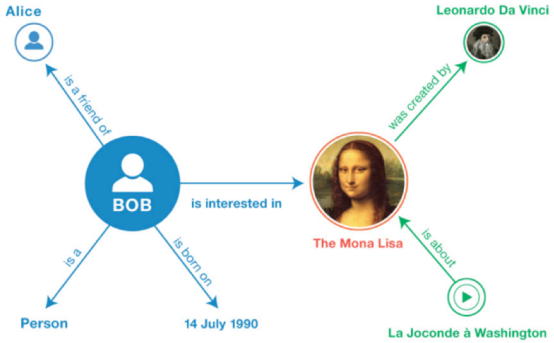
Representation Learning in Graphs



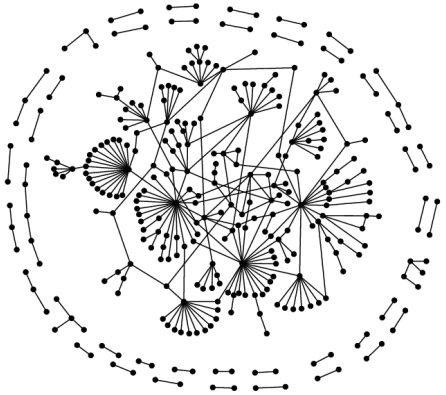
Networks of Interactions



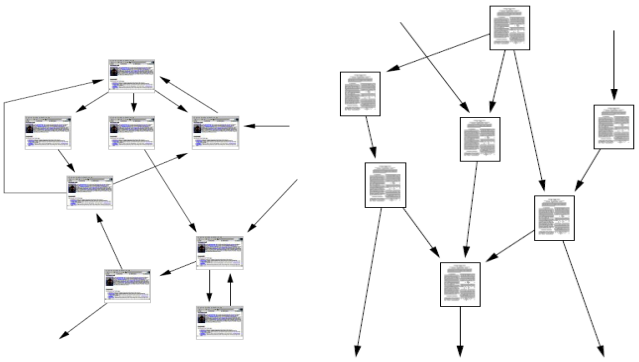
Social networks



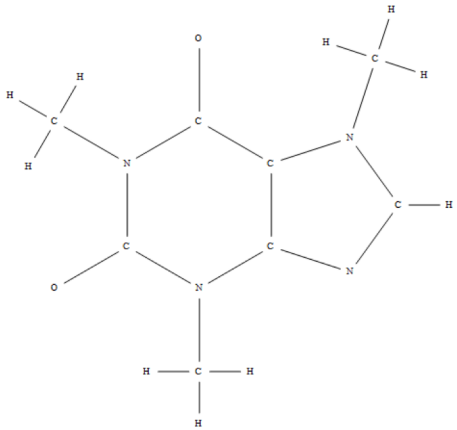
Knowledge graphs



Biological networks



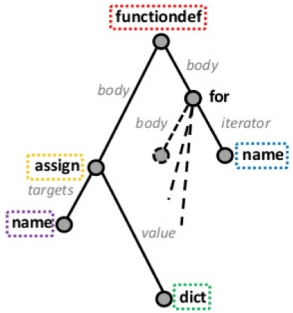
Complex Systems



Molecules

```
def encode(obj):
    """
    Encode a (possibly nested)
    dictionary containing complex values
    into a form that can be serialized
    using JSON.
    """
    e = {}
    for key,value in obj.items():
        if isinstance(value,dict):
            e[key] = encode(value)
        elif isinstance(value,complex):
            e[key] = {'type' : 'complex',
                      'r' : value.real,
                      'i' : value.imag}
    return e
```

```
import ast
tree = ast.parse("    ")
...
```

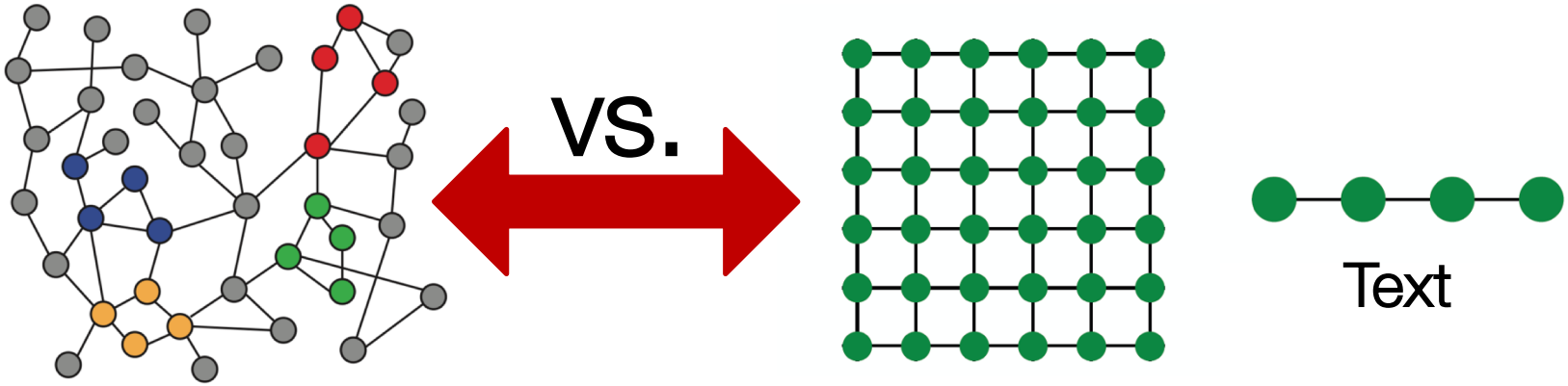


Code

Why is it Hard?

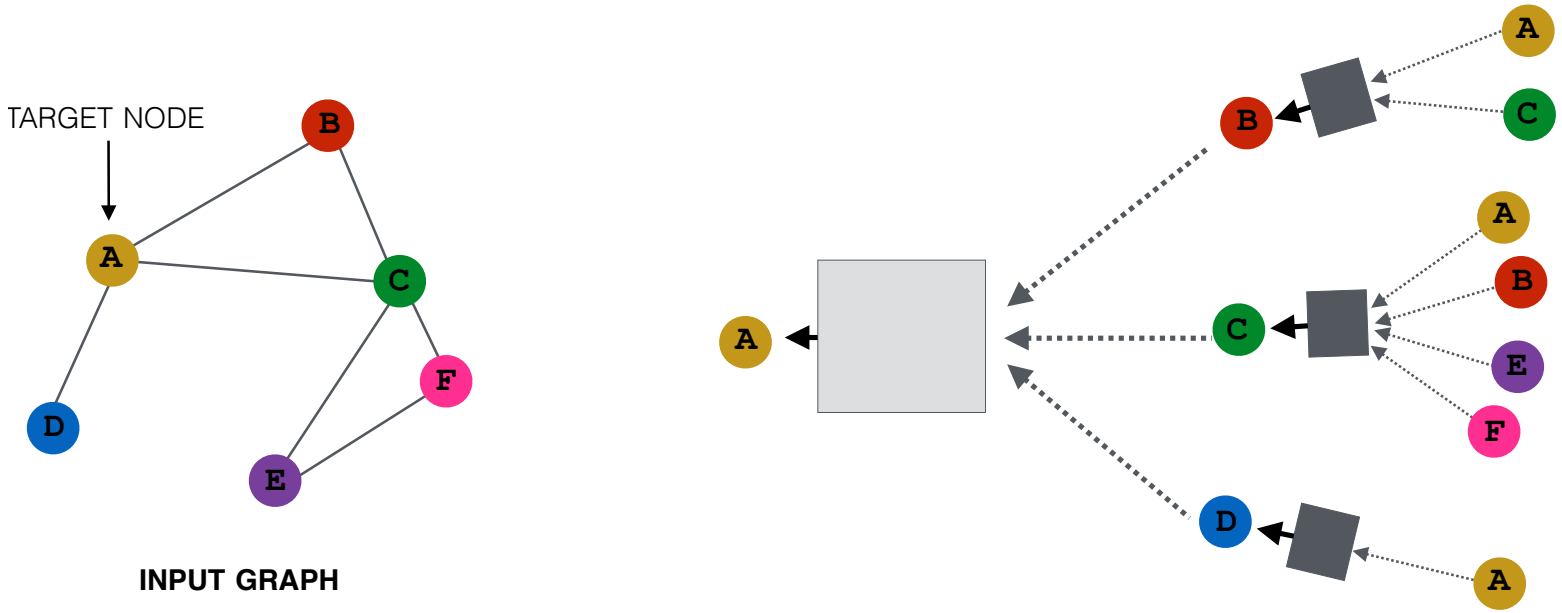
Networks are complex!

- Arbitrary size and complex topological structure (i.e., no spatial locality like grids)



- No fixed node ordering or reference point
- Often dynamic and have multimodal features

Graph Neural Networks

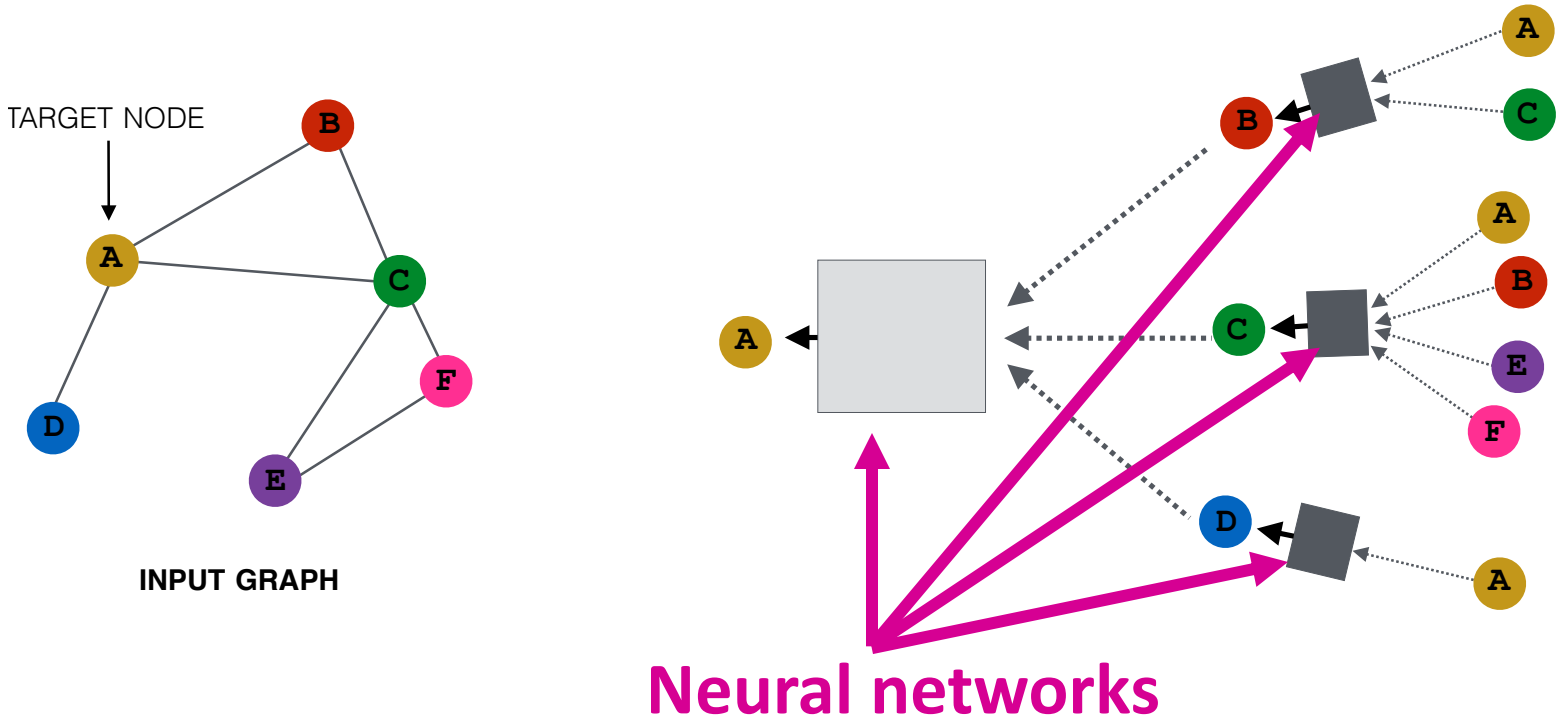


Each node defines a computation graph

- Each edge in this graph is a transformation/aggregation function

Scarselli et al. 2005. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*.

Graph Neural Networks

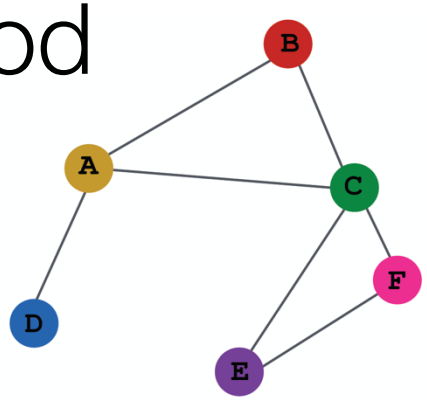


Intuition: Nodes aggregate information from their neighbors using neural networks

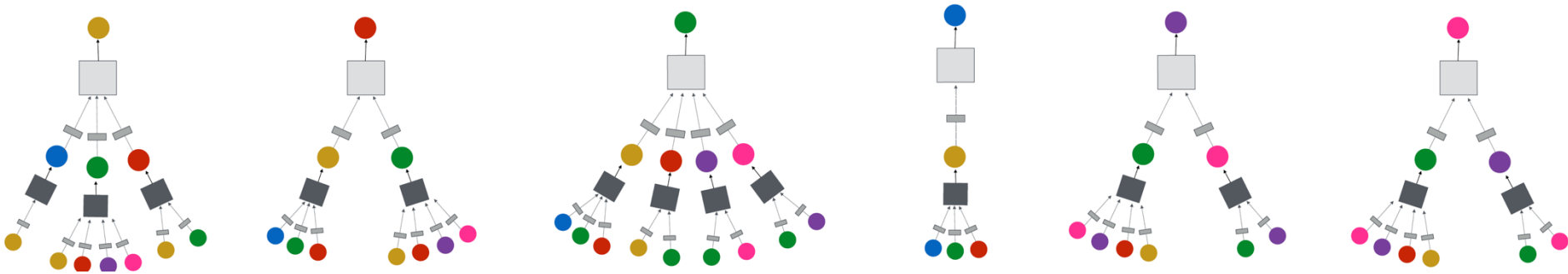
Inductive Representation Learning on Large Graphs. W. Hamilton, R. Ying, J. Leskovec. NIPS, 2017.

Idea: Aggregate Neighbors

Intuition: Network neighborhood defines a computation graph



Every node defines a computation graph based on its neighborhood!



Can be viewed as learning a generic linear combination of graph low-pass and high-pass operators

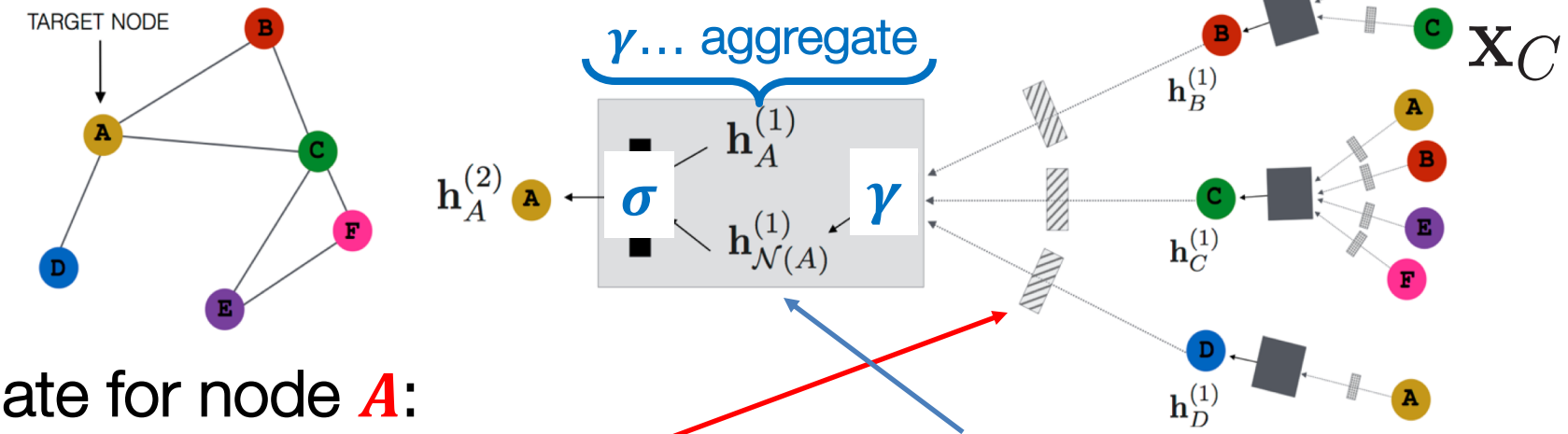
[Bronstein et al., 2017]

Our Approach: GraphSAGE

$$\phi(\gamma_{x \in \mathcal{N}(A)}(h_x))$$

σ… combine

γ… aggregate



Update for node A:

$$h_A^{(k+1)} = \sigma\left(W^{(k)} h_A^{(k)}, \gamma_{x \in \mathcal{N}(A)}\left(\sigma(Q^{(k)} h_x^{(k)})\right)\right)$$

$k+1^{st}$ level embedding of node A

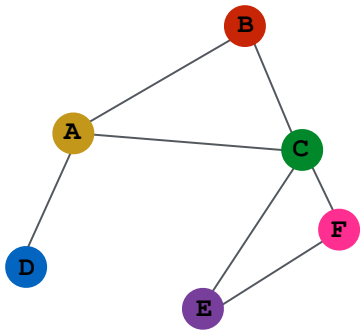
Transform A's own embedding from level k

Transform and aggregate embeddings of neighbors n

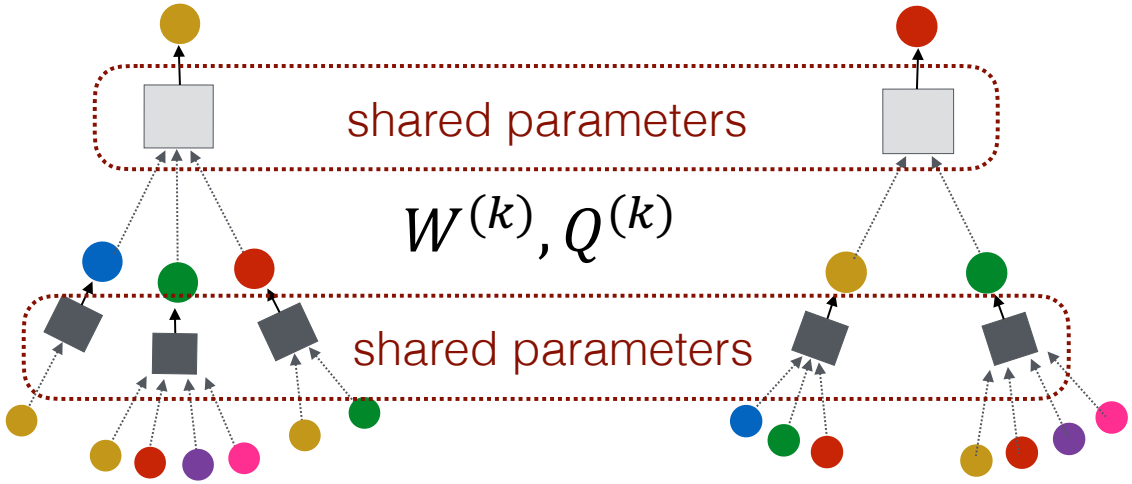
- $h_A^{(0)}$ = features X_A of node A, $\sigma(\cdot)$ is a sigmoid activation function

GraphSAGE: Training

- Aggregation parameters are shared for all nodes
- Number of model parameters is independent of $|V|$
- Can use different loss functions:
  - Classification/Regression: $\mathcal{L}(h_A) = \left||y_A - f(h_A)|\right|^2$
  - Pairwise Loss: $\mathcal{L}(h_A, h_B) = \max(0, 1 - dist(h_A, h_B))$

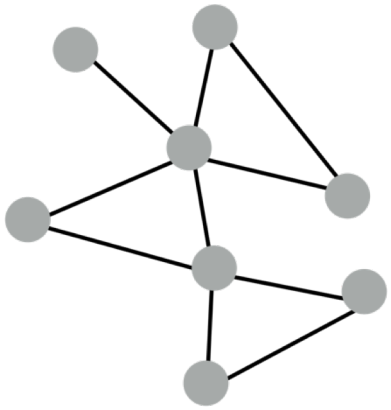


INPUT GRAPH

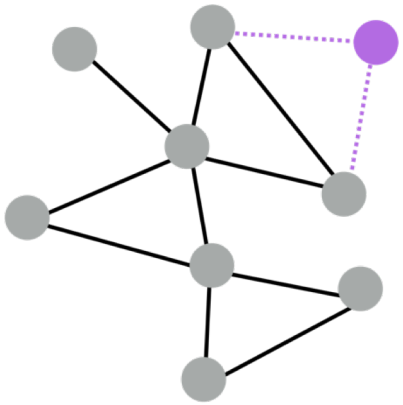


Compute graph for node A **Compute graph for node B**

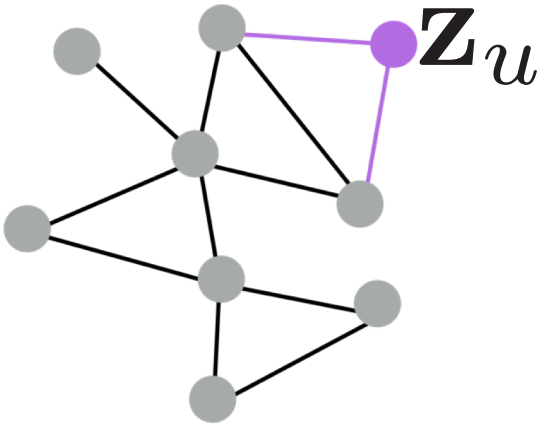
Inductive Capability



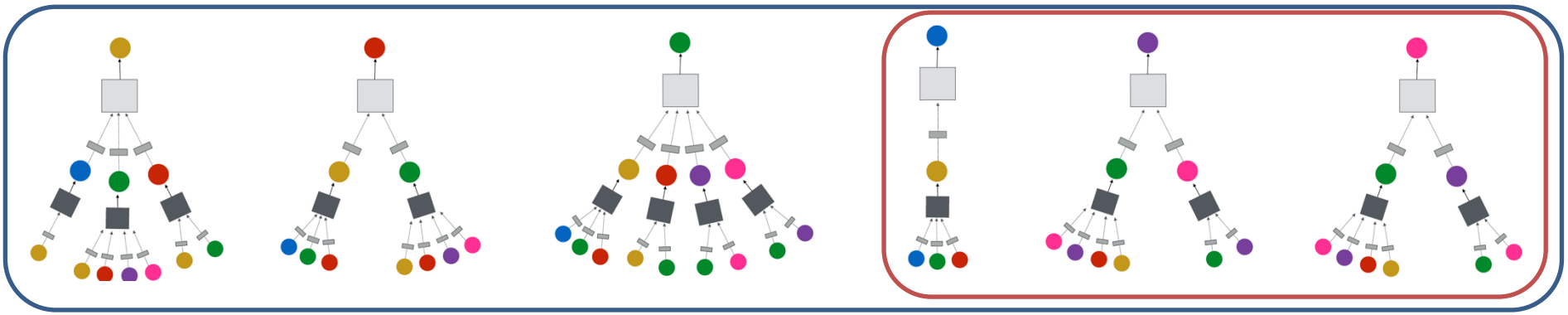
train with a snapshot



new node arrives



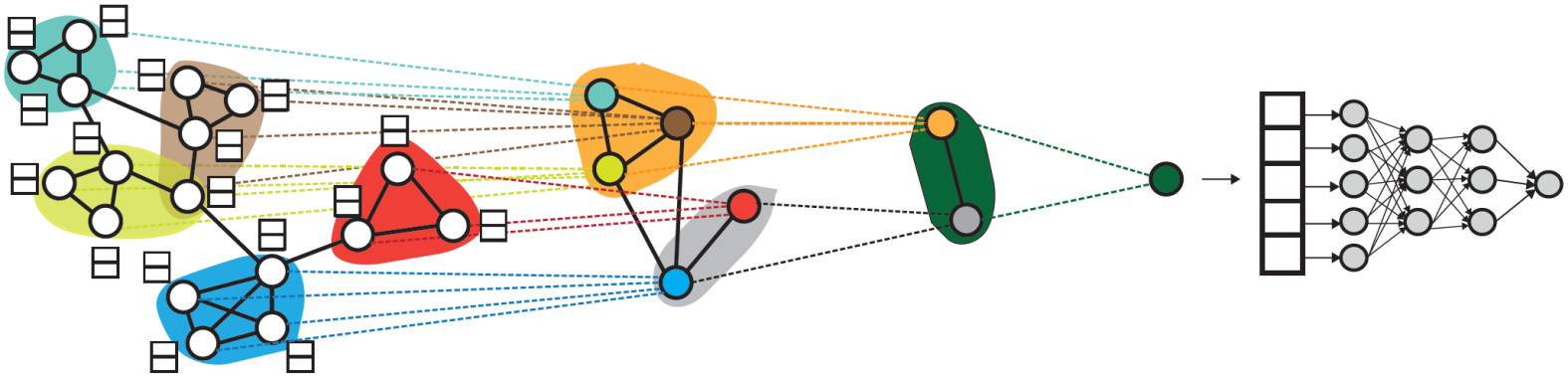
generate embedding for new node



Even for nodes we never trained on!

[NeurIPS '18]

DIFFPOOL: Pooling for GNNs



Don't just embed individual nodes. Embed the entire graph.

Problem: Learn how to hierarchical pool the nodes to embed the entire graph

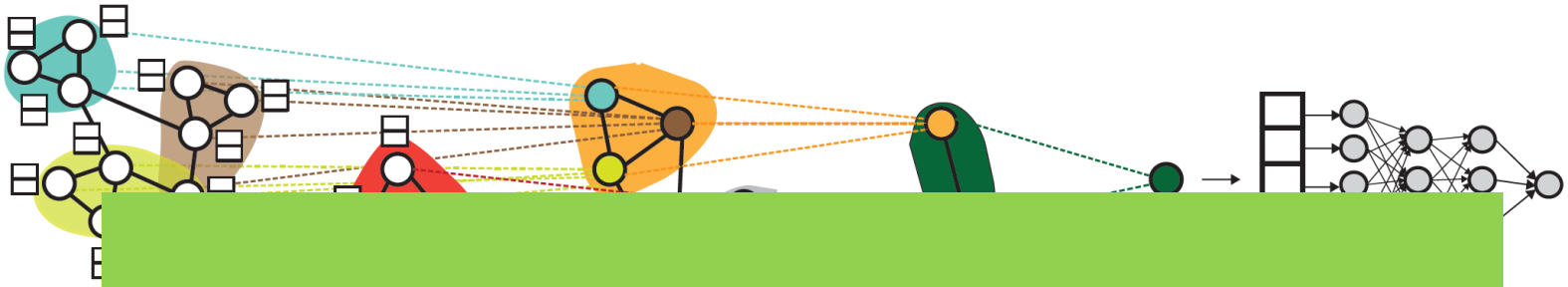
Our solution: **DIFFPOOL**

- Learns hierarchical pooling strategy
- Sets of nodes are pooled hierarchically
- Soft assignment of nodes to next-level nodes

Hierarchical Graph Representation Learning with Differentiable Pooling. R. Ying, et al. NeurIPS 2018.

[NeurIPS '18]

DIFFPOOL: Pooling for GNNs



How expressive are Graph Neural Networks?

Dor … the
enti…
Prol… e
nod…
Our…

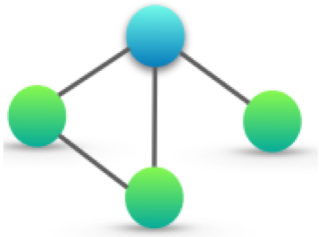
- Learns hierarchical pooling strategy
- Sets of nodes are pooled hierarchically
- Soft assignment of nodes to next-level nodes

Hierarchical Graph Representation Learning with Differentiable Pooling. R. Ying, et al. NeurIPS 2018.

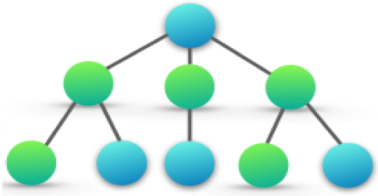
How expressive are GNNs?

Theoretical framework: Characterize GNN's discriminative power:

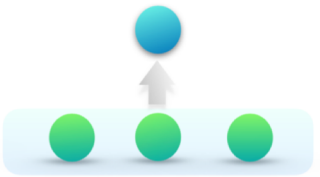
- Characterize upper bound of the discriminative power of GNNs
- Propose a maximally powerful GNN
- Characterize discriminative power of popular GNNs



GNN tree:



Aggregation:



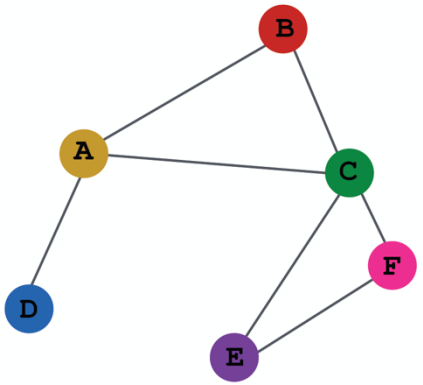
Multiset X

How Powerful are Graph Neural Networks? K. Xu, et al. ICLR 2019.

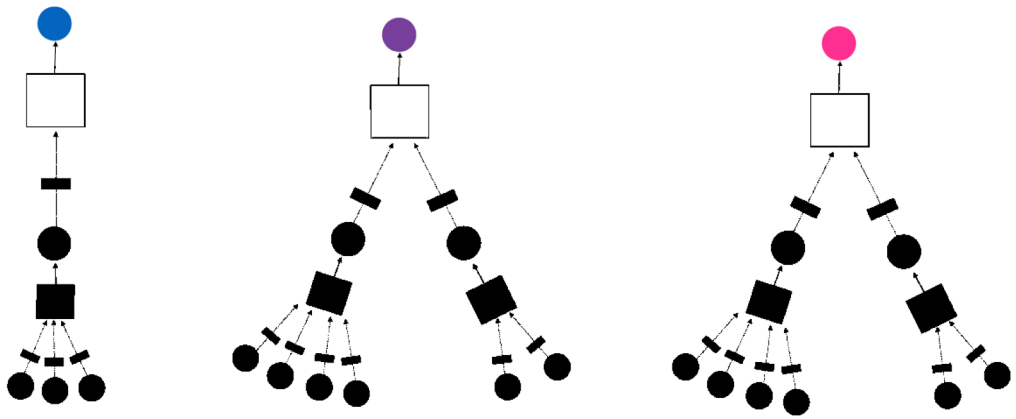
Key Insight: Rooted Subtrees

Assume **no node features**, then single nodes cannot be distinguished but rooted trees can be distinguished:

Graph:



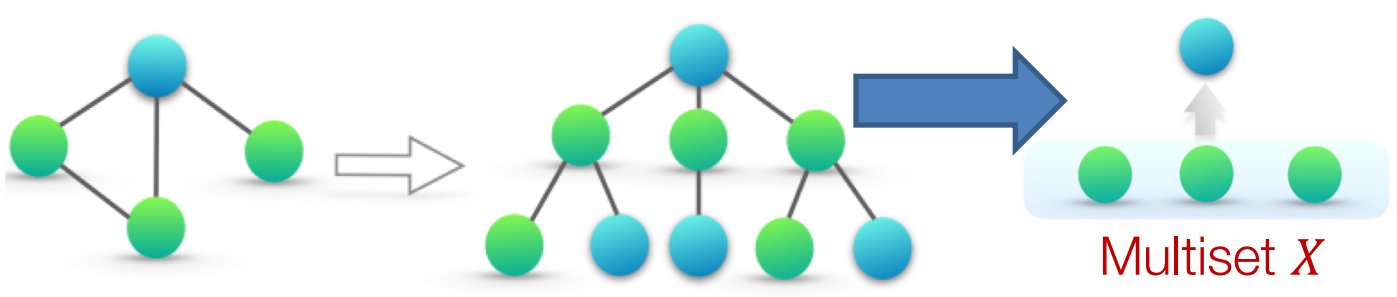
GNN distinguishes:

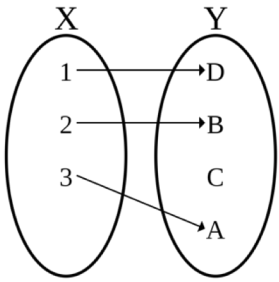


The most powerful GNN is able to distinguish rooted subtrees of different structure

GNN can distinguish blue D and violet E but not violet E and pink F node

Discriminative Power of GNNs





Idea: If GNN aggregation is injective, then different multisets are distinguished and GNN can capture subtree structure

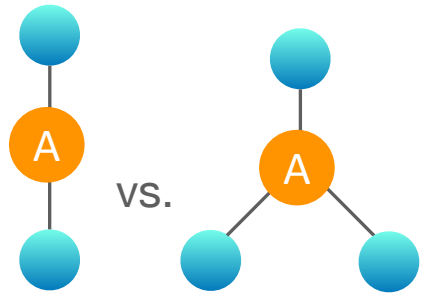
Theorem: Injective multiset function g can be written as: $g(X) = \phi(\sum_{x \in X} f(x))$

Funcs. ϕ, f are based on universal approximation thm.

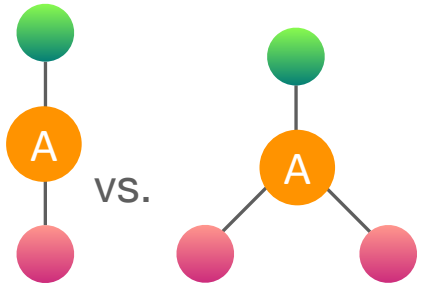
Consequence: Sum aggregator is the most expressive!

Power of Aggregators

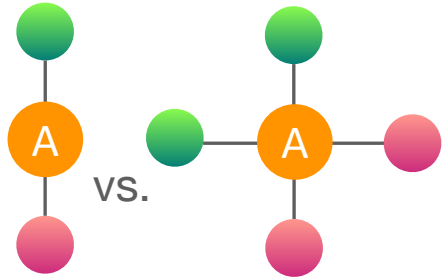
Failure cases for mean and max agg.



(a) Mean and Max both fail

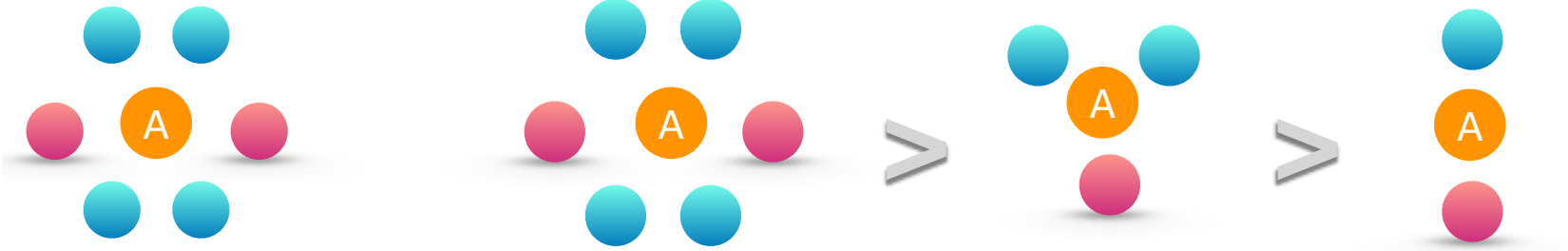


(b) Max fails



(c) Mean and Max both fail

Ranking by discriminative power

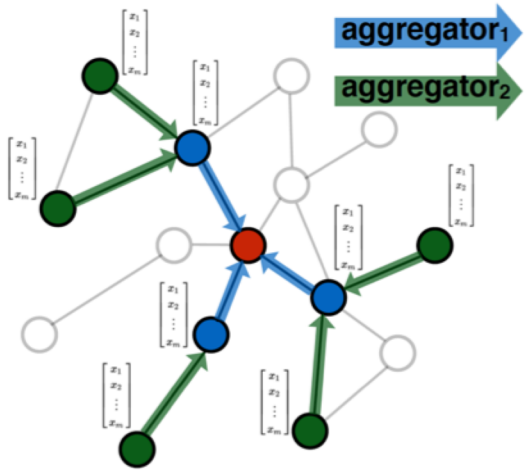


Input sum - multiset mean - distribution max - set

Three Consequences of GNNs

1) The GNN does two things:

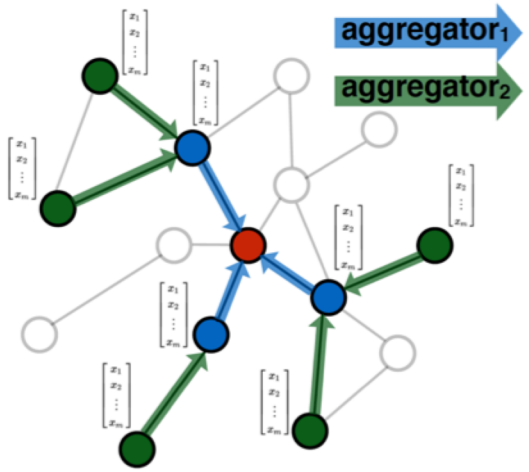
- **1)** Learns how to "borrow" feature information from nearby nodes to enrich the target node



- **2)** Each node can have a different computation graph and the network is also able to capture/learn its structure

Three Consequences of GNNs

2) Computation graphs can be chosen:

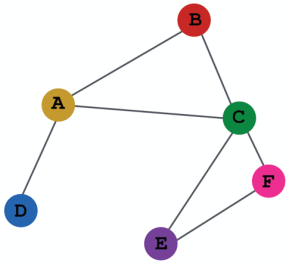


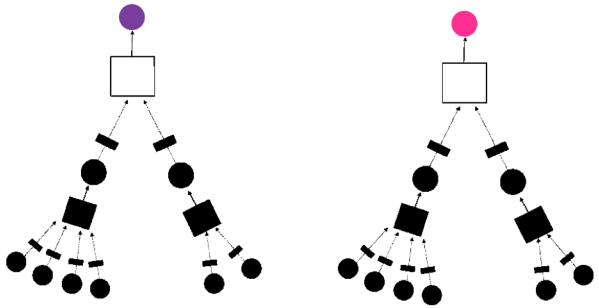
- Aggregation does not need to happen across all neighbors
- Neighbors can be strategically chosen/sampled
- Leads to big gains in practice!

Three Consequences of GNNs

3) We understand GNN failure cases:

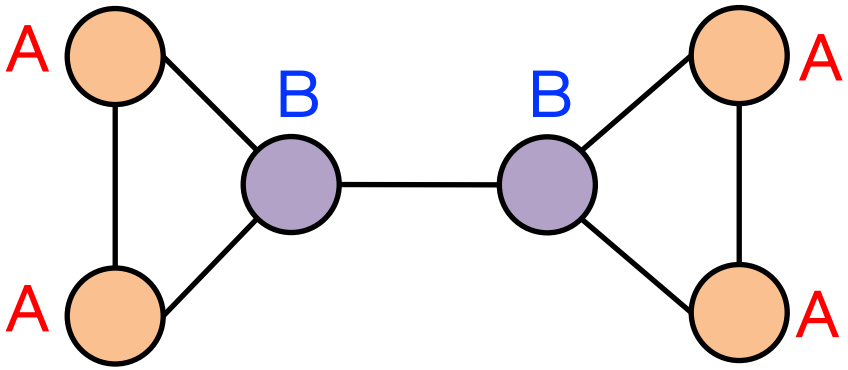
- GNNs fail to distinguish isomorphic nodes
  - Nodes with identical rooted subtrees will be classified in the same class (in the absence of differentiating node features)



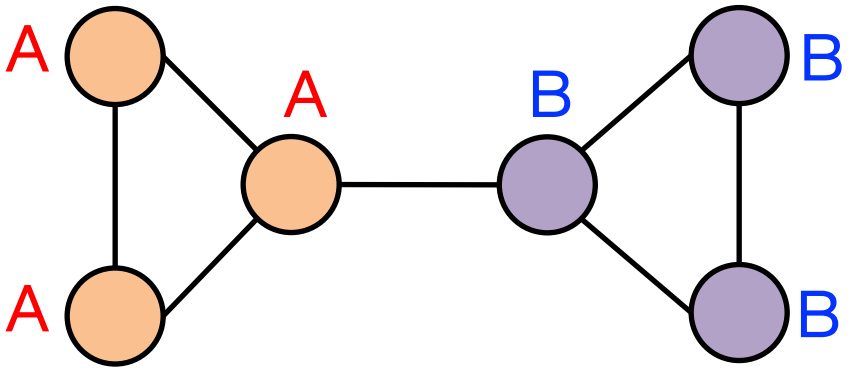


Structure-Aware Task

Position-Aware Task



Vs.

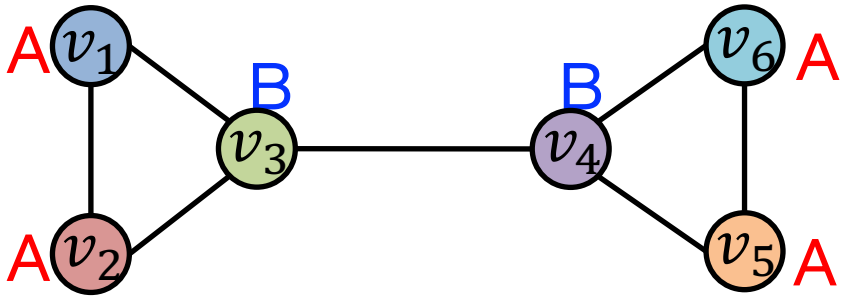


GNNs can predict ☺

GNNs cannot predict ☹

Core Issues with GNNs

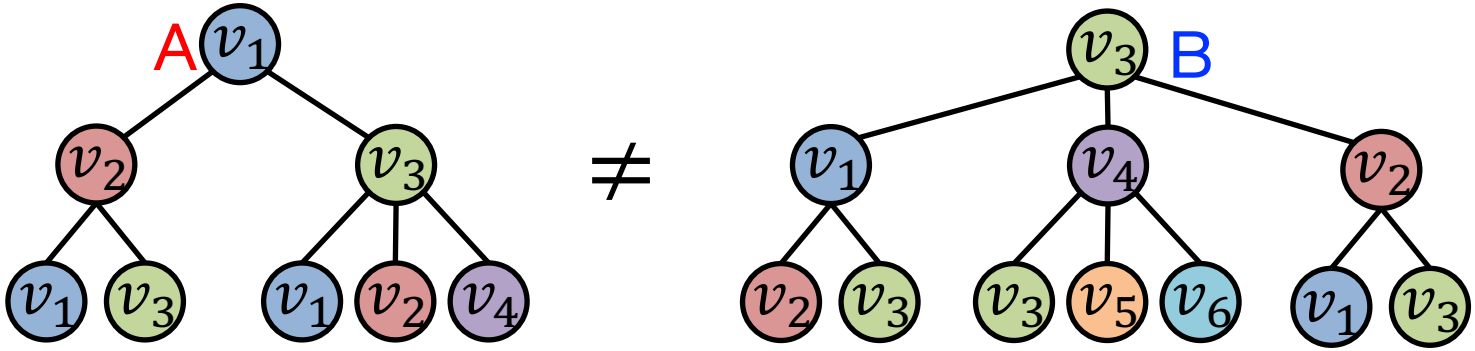
Structure-aware task:



GNNs work ☺

GNN can distinguish nodes v_1 and v_3

v_1 and v_3 have different comp. graphs:



Core Issues with GNNs

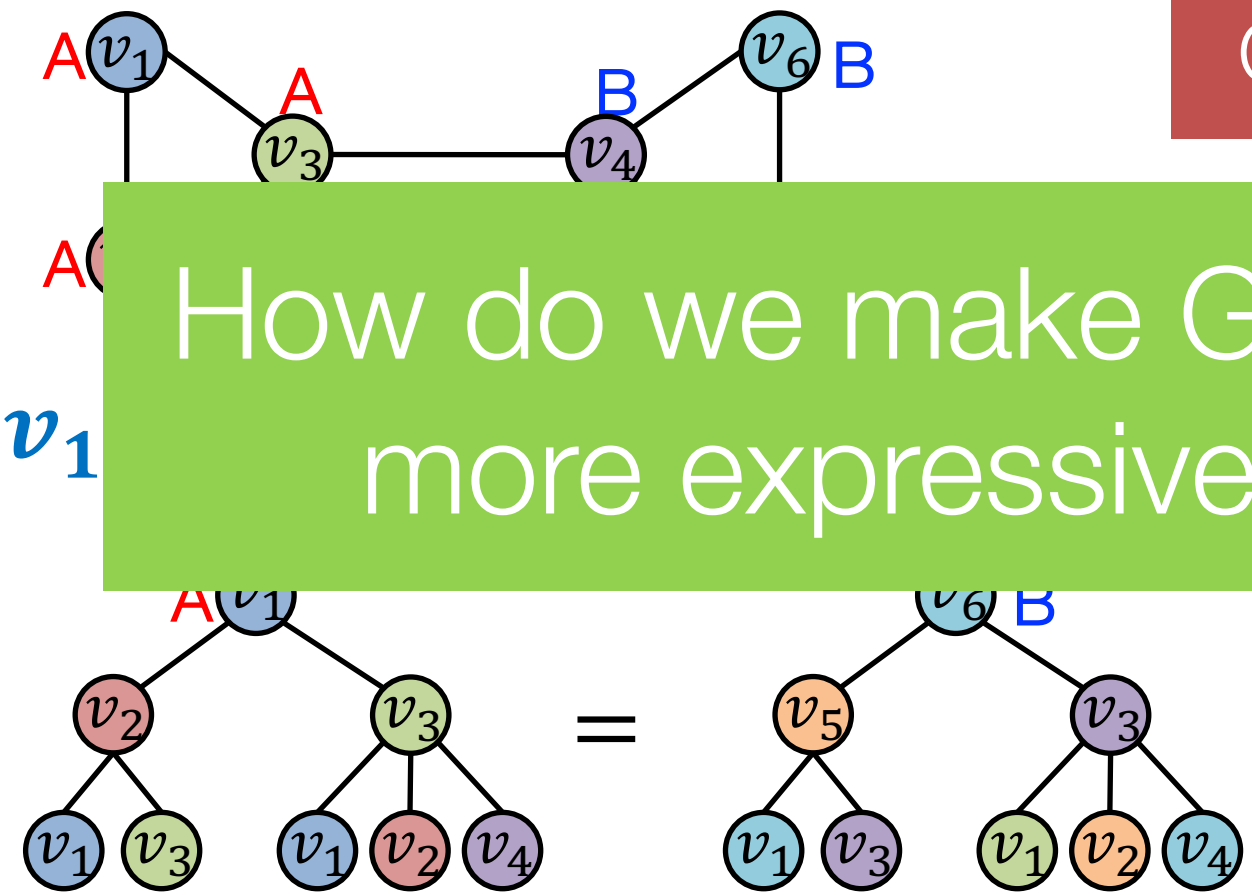
Position-aware task:

GNNs fail ☹

inguish
v_6

s:

How do we make GNNs more expressive?



Our Solution

PGNN: Position-Aware Graph Neural Networks

Position-aware Graph Neural Networks. J. You, R. Ying, J. Leskovec. *ICML*, 2019.
Data, code: https://snap.stanford.edu/pgnn

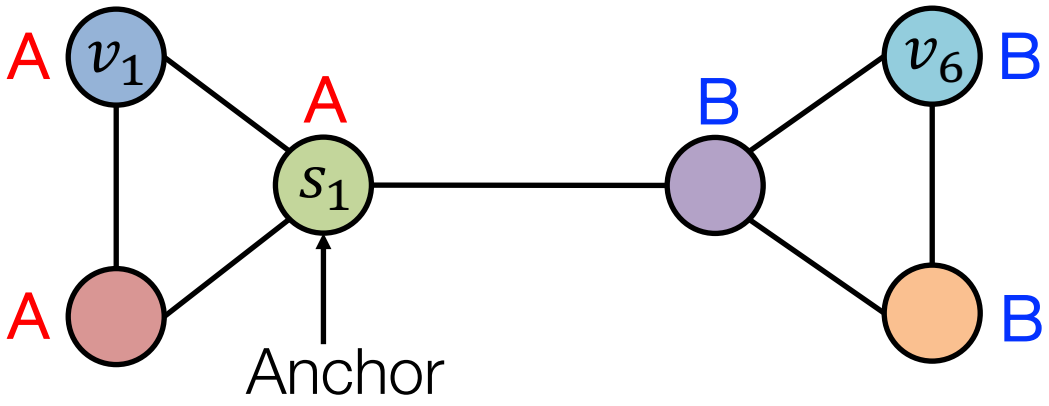
Key Insight: Anchors

Idea: Nodes need to know "where" in the network they are

Solution:

- Anchor: a randomly selected node
- Anchor-set: a randomly selected node subset, anchor is a size-1 anchor-set
- (1) Randomly choose many anchor-sets
- (2) A given node can then use its distance to these anchor-sets to understand its location/position in the network

Power of "Anchor"

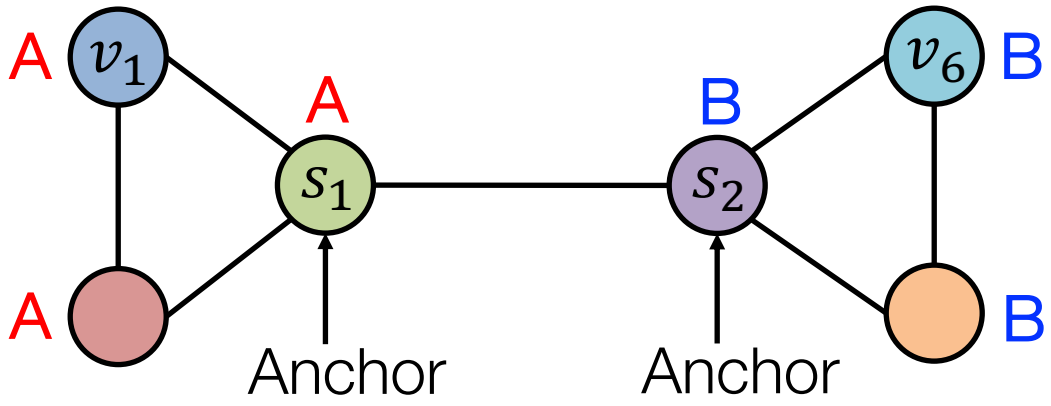


Relative Distances

	s_1
v_1	1
v_6	2

Observation: Represent v_1 and v_6 via their relative distances w.r.t. an anchor s_1, which are different

Power of "Anchors"

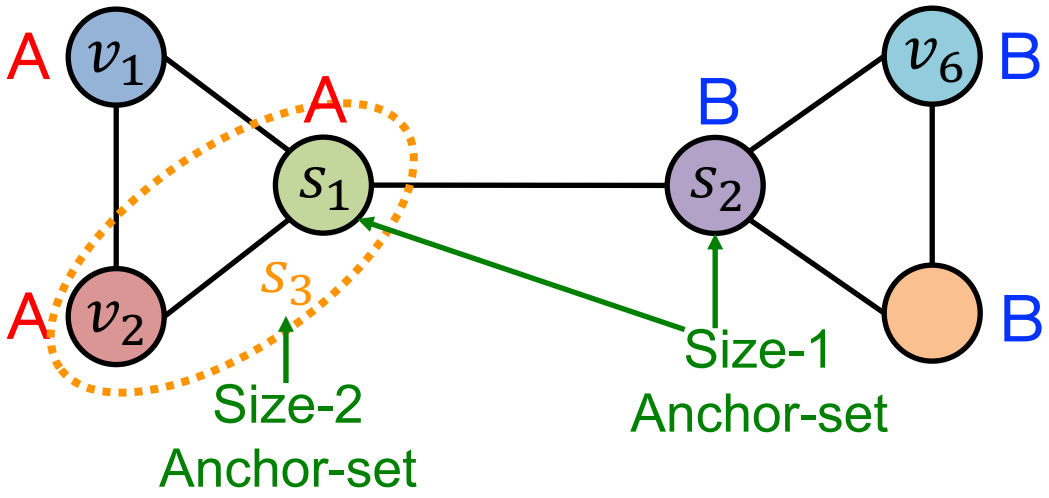


Relative Distances

	s_1	s_2
v_1	1	2
v_6	2	1

Observation: More anchors can better characterize node position

Power of "Anchor-sets"



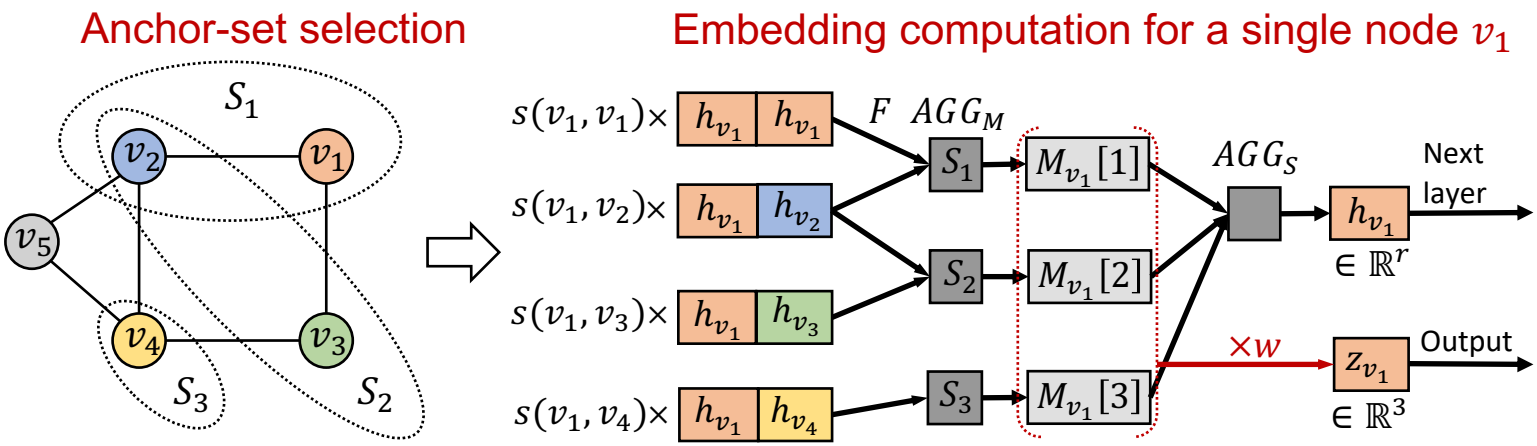
Relative Distances

	s_1	s_2	s_3
v_1	1	2	1
v_6	2	1	2
v_2	1	2	0

Observation: Large anchor-sets are more likely to differentiate symmetric nodes (v_1 and v_2), thus save the number of anchors we need to use.

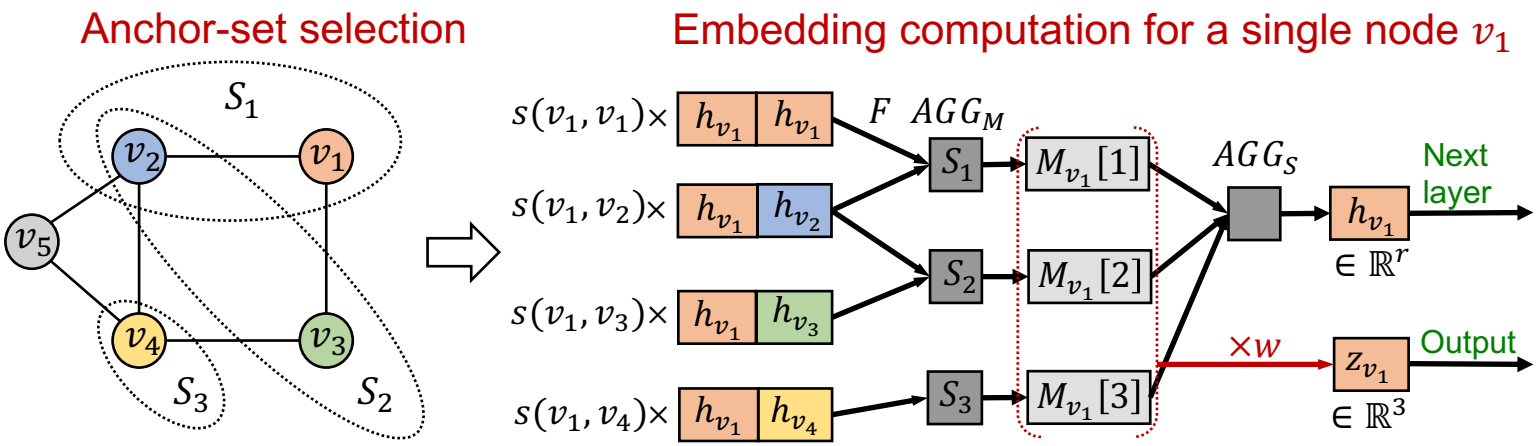
Position-aware GNN Framework

- P-GNN is a family of models
- We demonstrate one possible design, but other models are possible (future work!)



Overview of Position-aware GNN

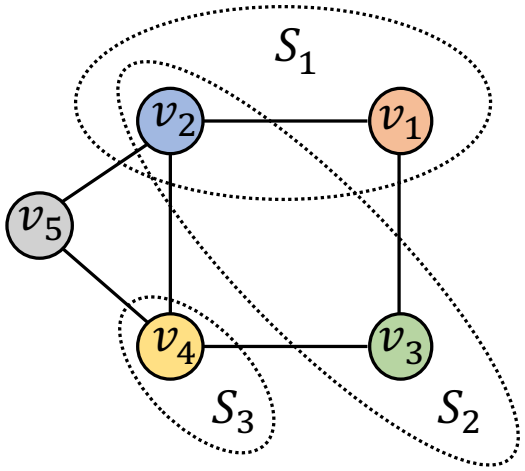
- **(a)** Randomly select anchor-sets
- **(b)** Compute pairwise node distances
- **(c)** Compute anchor-set messages
- **(d)** Transform messages to node embeddings



Position-aware GNN Framework

Step (a): Select anchor-sets [Bourgain, 1985]

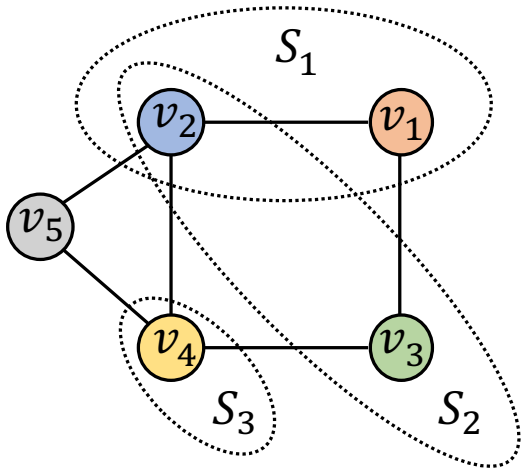
- Randomly choose anchor-sets with sizes from $1, 2, 4, ..., n/2$ ($\log(n)$ number of sizes)
- For each size of anchor-set, repeat $c \cdot \log(n)$ times
- In total, $k = c \cdot \log^2(n)$ anchors



Position-aware GNN Framework

Step (b): Compute pairwise node distances

- **Position-based similarities:** For example, shortest path, personalized PageRank, …
- We use k-hop shortest path distance $d_{sp}^{k}(v_i, v_j)$ for fast computation



$$s(v_i, v_j) = \frac{1}{d_{sp}^{k}(v_i, v_j) + 1}$$

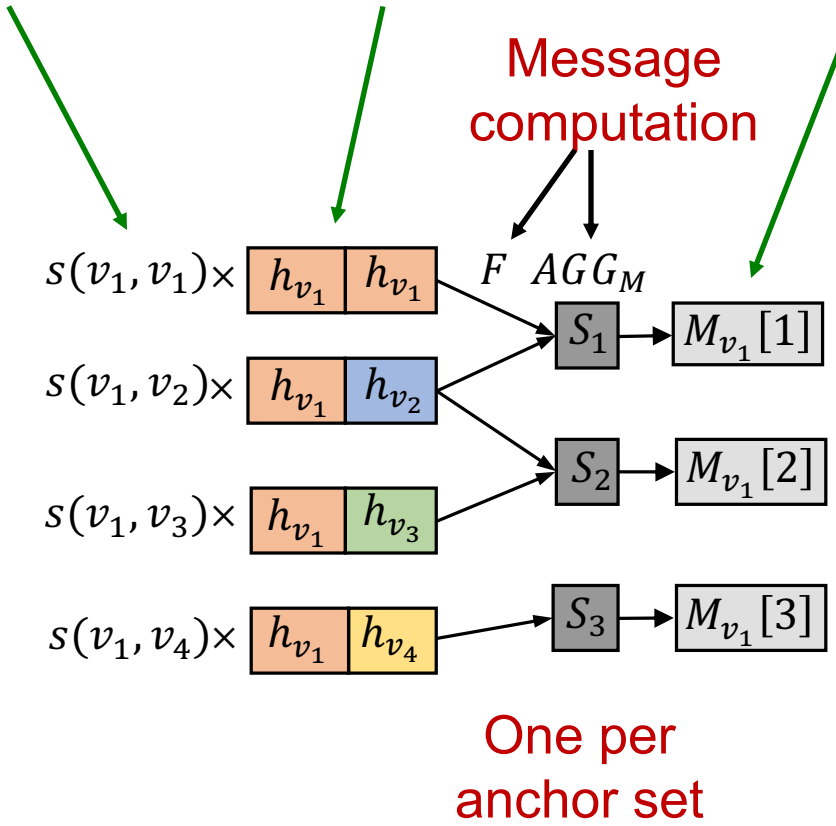
pairwise node distances $s(v_i, v_j)$

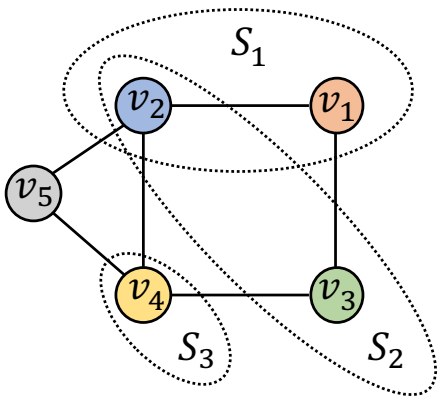
	v_1	v_2	v_3	v_4	v_5
v_1	1	0.5	0.5	0.3	0.3
v_2	0.5	1	0.3	0.5	0.5
v_3	0.5	0.3	1	0.5	0.3
v_4	0.3	0.5	0.5	1	0.5
v_5	0.3	0.5	0.3	0.5	1

Position-aware GNN Framework

Step (c): Compute anchor-set messages

- Position info + Feature info → Anchor-set Messages





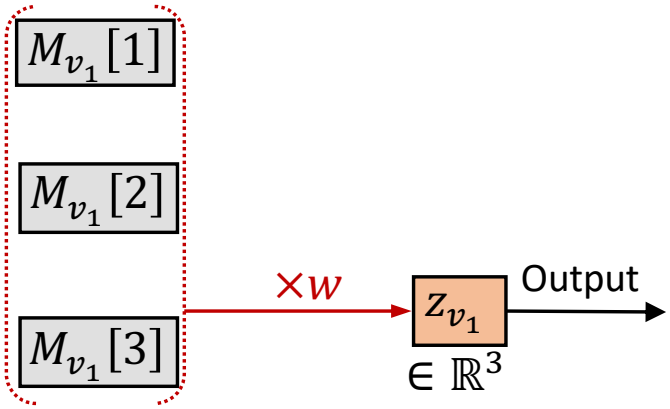
$s(v_i, v_j)$

	v_1	v_2	v_3	v_4	v_5
v_1	1	0.5	0.5	0.3	0.3
v_2	0.5	1	0.3	0.5	0.5
v_3	0.5	0.3	1	0.5	0.3
v_4	0.3	0.5	0.5	1	0.5
v_5	0.3	0.5	0.3	0.5	1

Position-aware GNN Framework

Step (d): From messages to node embeddings

Messages from 3 anchor-sets

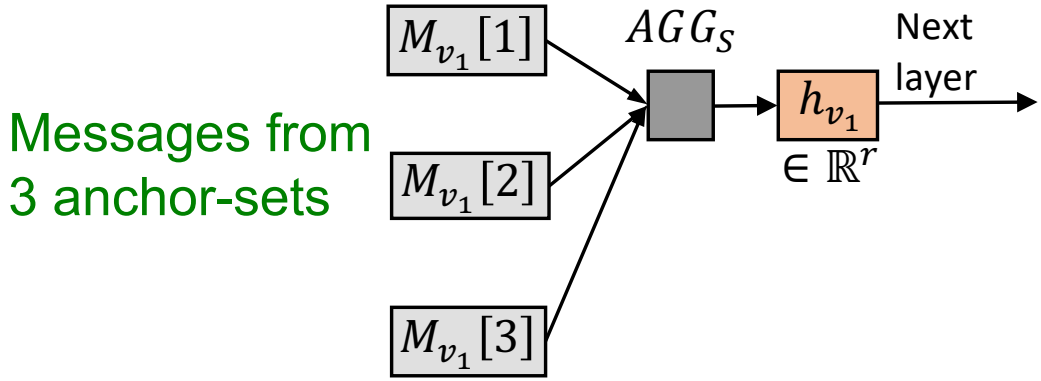


(1) Position-aware node embedding z_{v_1}:

- Output of P-GNN
- 3 anchor-sets → 3 dimensions
- Each dimension is tied to an anchor-set, and is thus position-aware

Position-aware GNN Framework

Step (d): From messages to node embeddings



(2) Node message $\boldsymbol{h_{v_1}}$:

- Fed into next layer of P-GNN
- Aggregate over 3 anchor-sets, keeping the feature information
- Each dimension is independent form anchor-set selection

Tasks and Datasets

- **Link prediction:**
  - Grid: dimension 20 x 20
  - Communities: 20 communities of 20 nodes
  - PPI: 24 graphs of 3000 nodes, node have features
- **Pairwise-node classification:**
  - Node label: Whether two nodes belong to the same community, or same class
  - Communities: 20 communities of 20 nodes
  - Emails: 7 graphs with 6 communities, have features
  - Protein: 113 protein graphs, node have features

Results: Link Prediction

- Link prediction

	Grid	Communities	PPI
GCN	0.456 ± 0.037	0.512 ± 0.008	0.769 ± 0.002
GraphSAGE	0.532 ± 0.050	0.516 ± 0.010	0.803 ± 0.005
GAT	0.566 ± 0.052	0.618 ± 0.025	0.783 ± 0.004
GIN	0.499 ± 0.054	0.692 ± 0.049	0.782 ± 0.010
P-GNN-Fast	0.619 ± 0.080	0.939 ± 0.083	0.719 ± 0.027
P-GNN-Fast-2Layer	0.694 ± 0.066	$\mathbf{0.991} \pm 0.003$	0.805 ± 0.003
P-GNN-Exact	0.879 ± 0.039	0.985 ± 0.005	0.775 ± 0.029
P-GNN-Exact-2Layer	$\mathbf{0.940} \pm 0.027$	0.985 ± 0.008	$\mathbf{0.808} \pm 0.003$

P-GNN up to 66% improvement in ROC AUC

Results: Node Classification

- Pairwise node classification

	Communities	Email	Protein
GAT	0.520 ± 0.025	0.515 ± 0.019	0.515 ± 0.002
GraphSAGE	0.514 ± 0.028	0.511 ± 0.016	0.520 ± 0.003
GAT	0.620 ± 0.022	0.502 ± 0.015	0.528 ± 0.011
GIN	0.620 ± 0.102	0.545 ± 0.012	0.523 ± 0.002
P-GNN-F-1L	0.985 ± 0.008	0.630 ± 0.019	0.510 ± 0.010
P-GNN-F-2L	0.997 ± 0.006	$\mathbf{0.640} \pm 0.037$	$\mathbf{0.729} \pm 0.176$
P-GNN-E-1L	0.991 ± 0.013	0.625 ± 0.058	0.507 ± 0.006
P-GNN-E-2L	$\mathbf{1.0} \pm 0.001$	$\mathbf{0.640} \pm 0.029$	0.631 ± 0.175

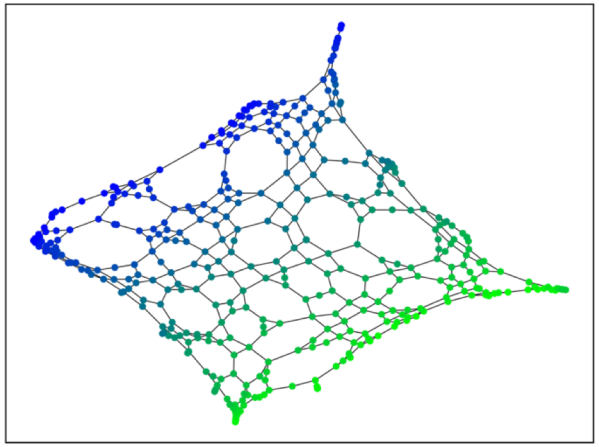
P-GNN up to 61% improvement in ROC AUC

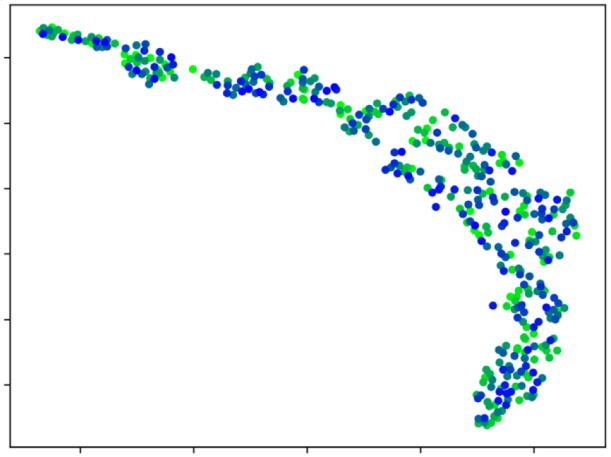
PGNN: Visualizing Embeddings

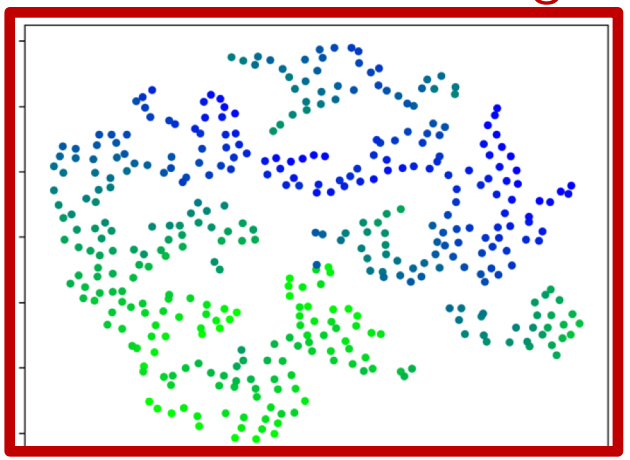
Input graph

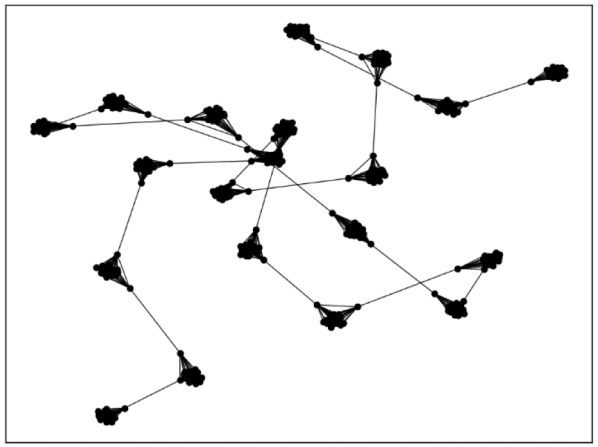
GNN embedding

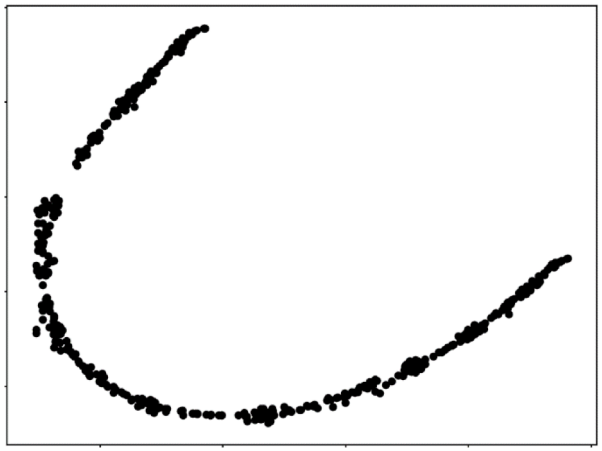
P-GNN embedding

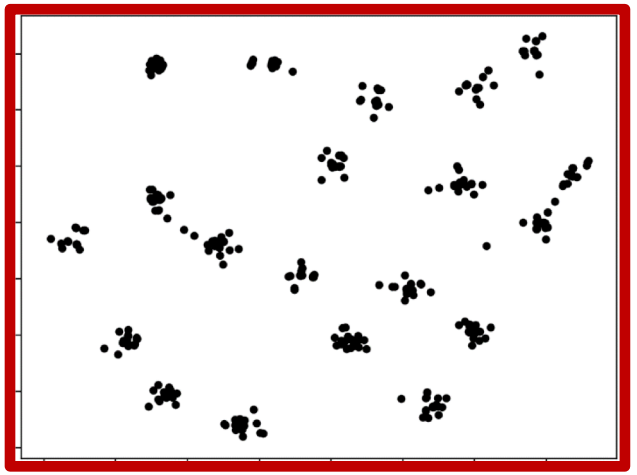












Published as a conference paper at ICLR 2020

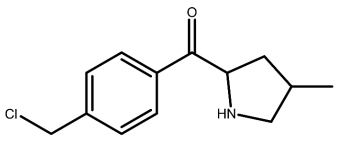
STRATEGIES FOR PRE-TRAINING GRAPH NEURAL NETWORKS

Weihua Hu[1]*, **Bowen Liu**[2]*, **Joseph Gomes**[4], **Marinka Zitnik**[5],
Percy Liang[1], **Vijay Pande**[3], **Jure Leskovec**[1]
[1]Department of Computer Science, [2]Chemistry, [3]Bioengineering, Stanford University,
[4]Department of Chemical and Biochemical Engineering, The University of Iowa,
[5]Department of Biomedical Informatics, Harvard University
{weihuahu,liubowen,pliang,jure}@cs.stanford.edu,
joe-gomes@uiowa.edu, marinka@hms.harvard.edu, pande@stanford.edu

Strategies For Pre-training Graph Neural Networks. W. Hu, B. Liu, J. Gomes, M. Zitnik, P. Liang, V. Pande, J. Leskovec. *ICLR*, 2020.

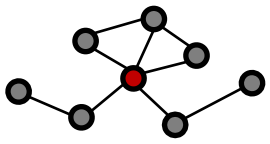
Graph ML in Scientific Domains

- **Chemistry:** Molecular graphs
  - Molecular property prediction

f(

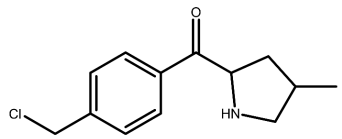
) = toxic?

- **Biology:** Protein Protein Interaction Networks
  - Protein function prediction

f () = biological activity?

Graph ML in Scientific Domains

- **Chemistry:** Molecular graphs
  - Molecular property prediction

f() = toxic?

Our running example

- Biology: Protein Protein Interaction Networks
  - Protein function prediction

f ( ) = biological activity?

Challenges for ML in Scientific Domains

1. **Scarcity of labeled data**
   - Obtaining labels requires expensive lab experiments
     → GNNs overfit to small training datasets

2. **Out-of-distribution prediction**
   - Test examples tend to be very different from training examples
     → GNNs extrapolate poorly

Our Solution: Pre-training GNNs

We design GNN pre-training strategies to systematically investigate:

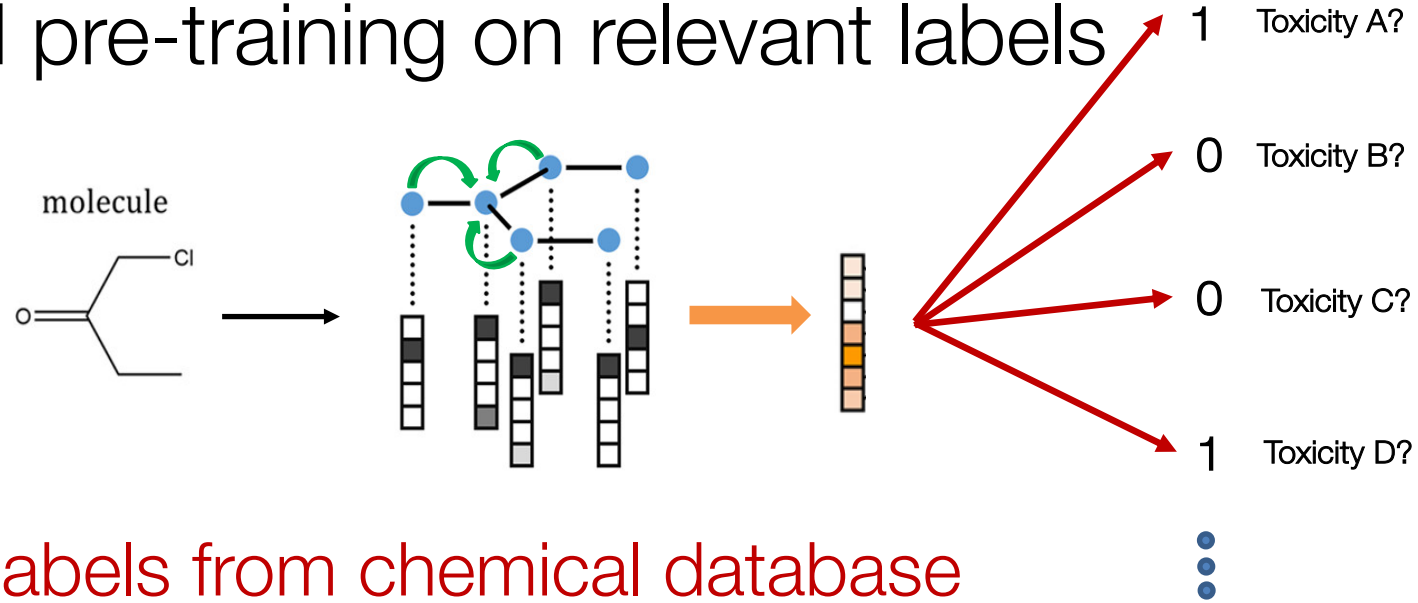
Q1: How effective is pre-training GNNs?

Q2: What are the most effective strategies?

How Effective is Pre-training GNNs?

Naïve strategy:

Supervised pre-training on relevant labels



Pre-training labels from chemical database

- ~450K molecules from ChEMBL
- 1310 diverse binary tasks

ChEMBL

Downstream task:

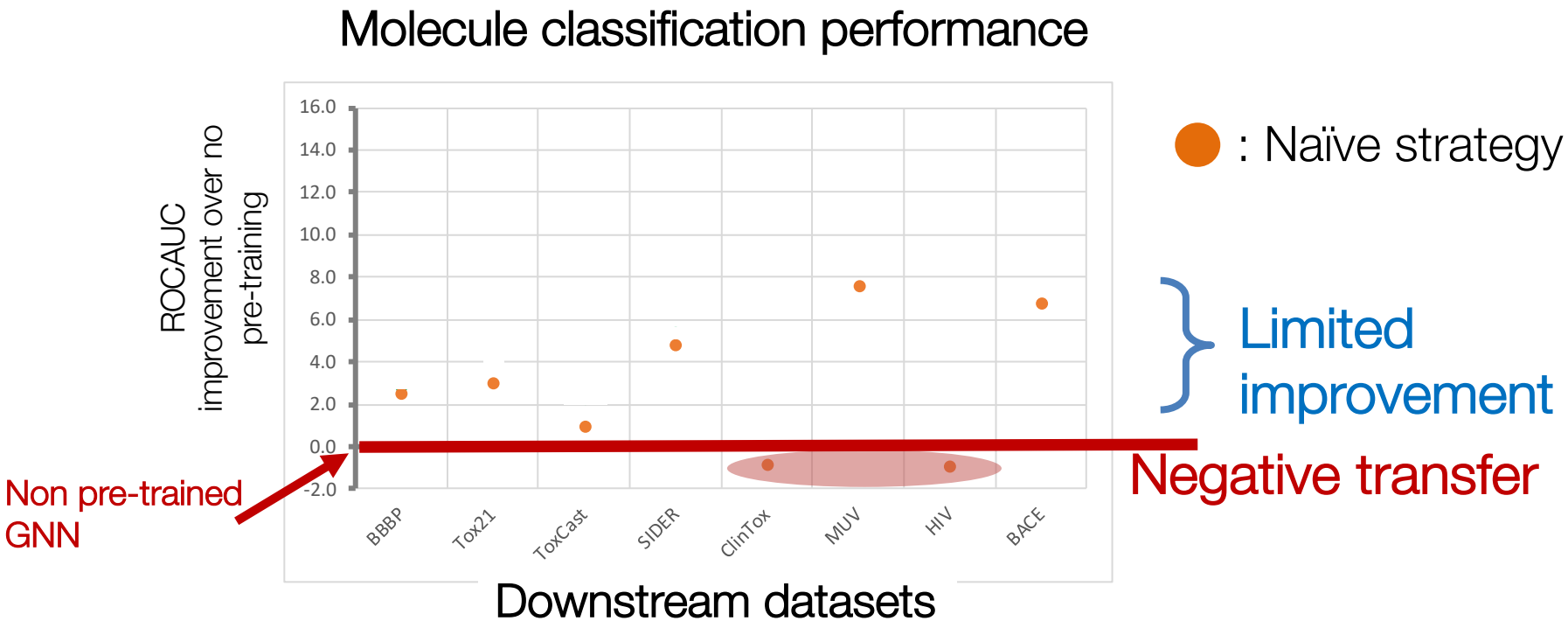
- 8 diverse molecular classification datasets

How Effective is Pre-training GNNs?

Naïve strategy:

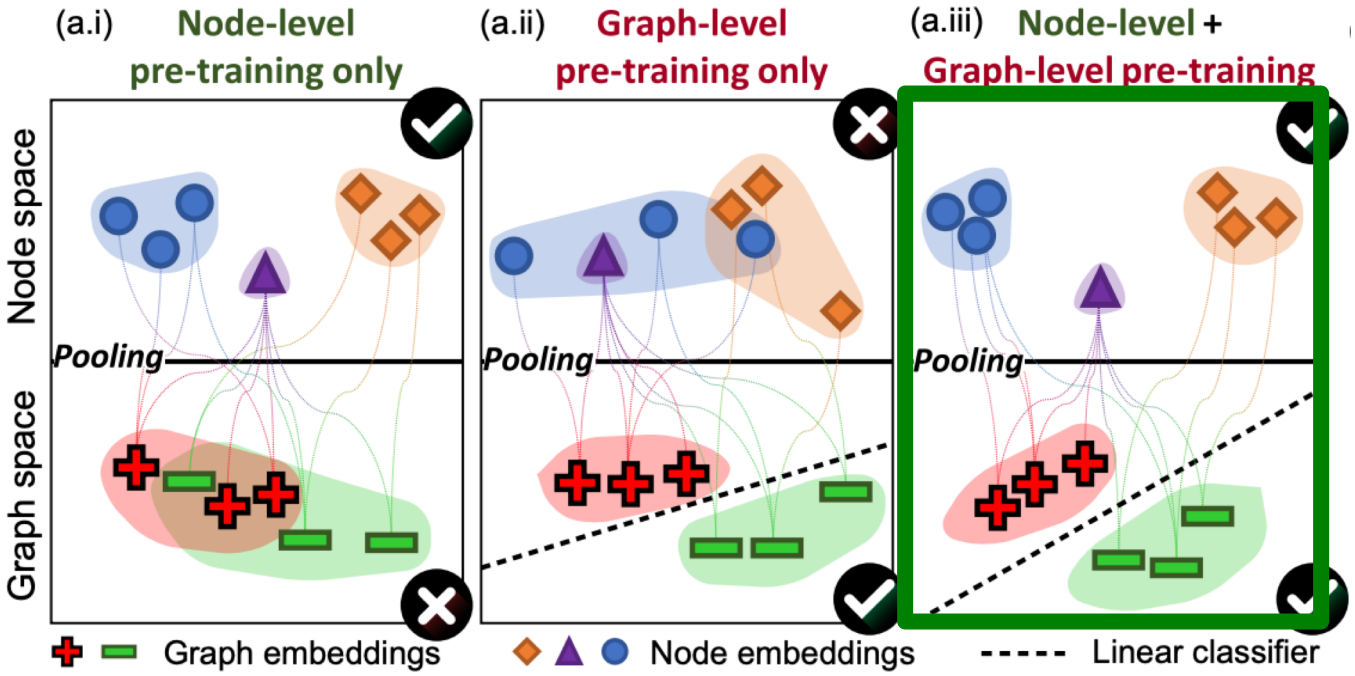
Supervised pre-training on relevant labels

-> Limited improvement. Often leads to **negative transfer!**



What are the Effective Strategies?

Key insight: Pre-train both node and graph embeddings

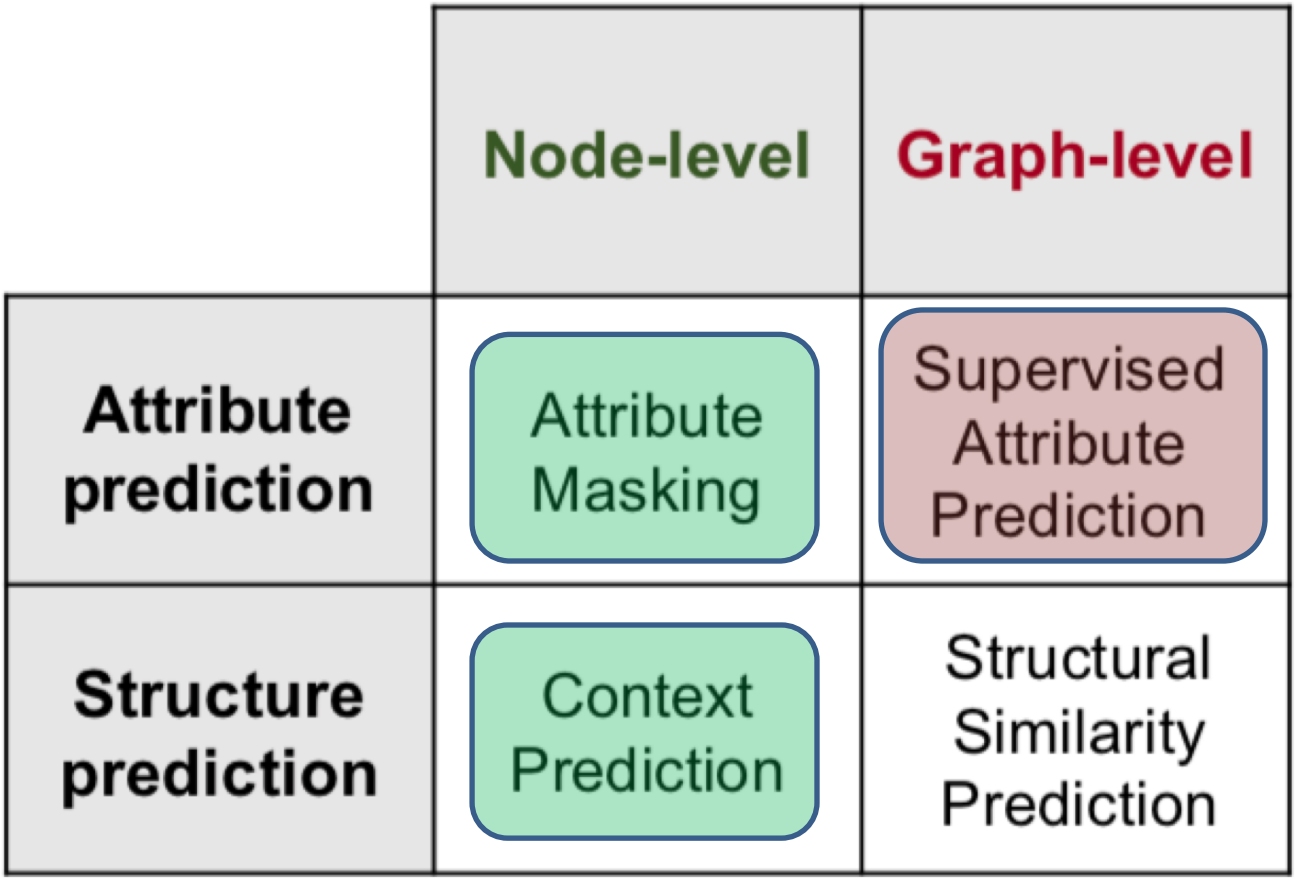


Nodes are separated. But embeddings are not composable (graphs are not separated)!

Graphs are separated. But embeddings of individual nodes are not!

Separation both in node as well as in graph space.

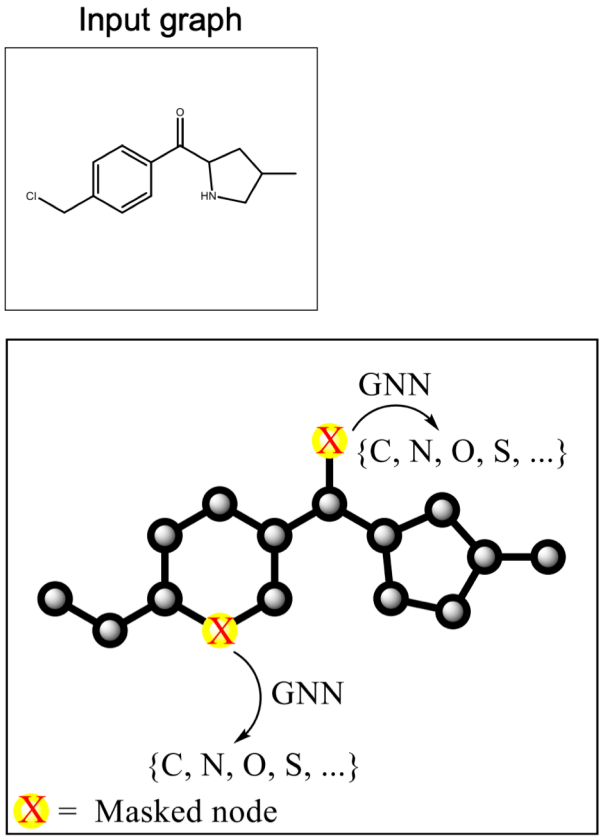
Possible Pre-training Methods



Attribute Masking

1. Mask node attributes: Atom types
2. Let GNNs predict them

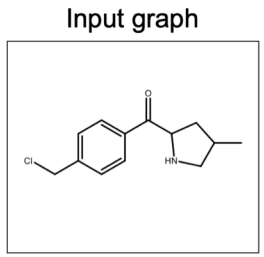
	Node-level	Graph-level
Attribute prediction	Attribute Masking	Supervised Attribute Prediction
Structure prediction	Context Prediction	Structural Similarity Prediction

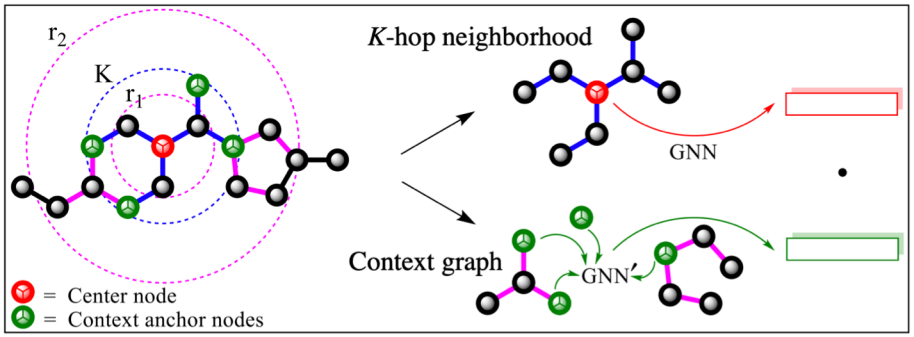


Context Prediction

1. Sample a center node for each molecule
2. Sample neighborhood, and a context graph
3. Let GNN distinguish true (neighborhood, context) pairs from false ones

	Node-level	Graph-level
Attribute prediction	Attribute Masking	Supervised Attribute Prediction
Structure prediction	Context Prediction	Structural Similarity Prediction

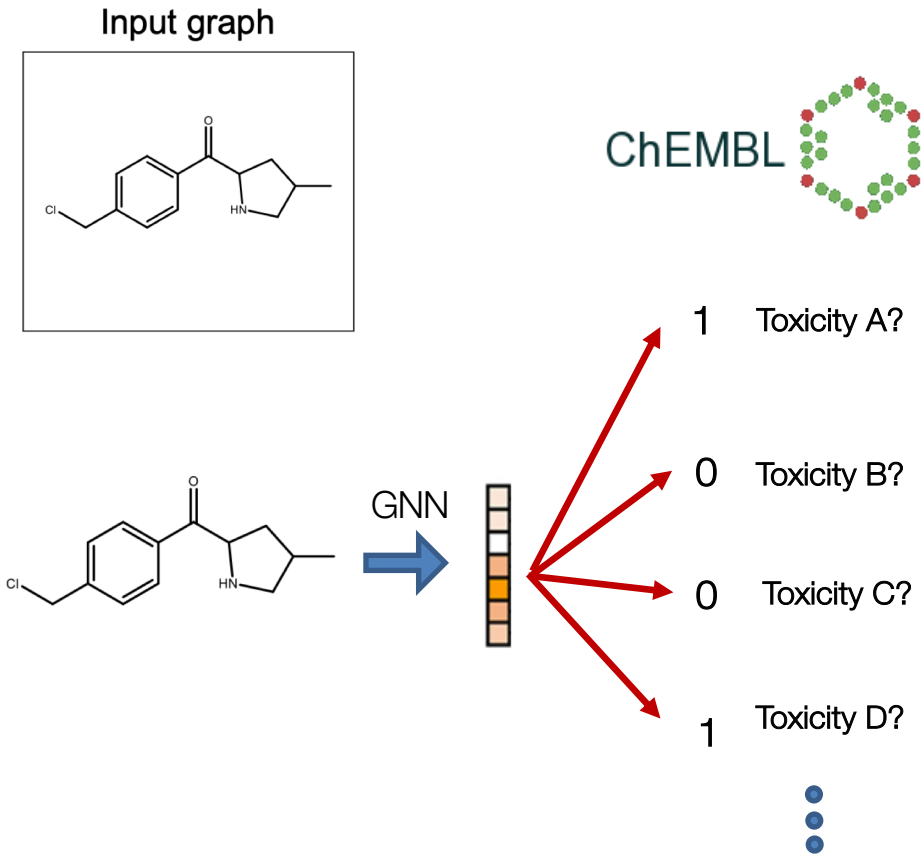




Supervised Attribute Prediction

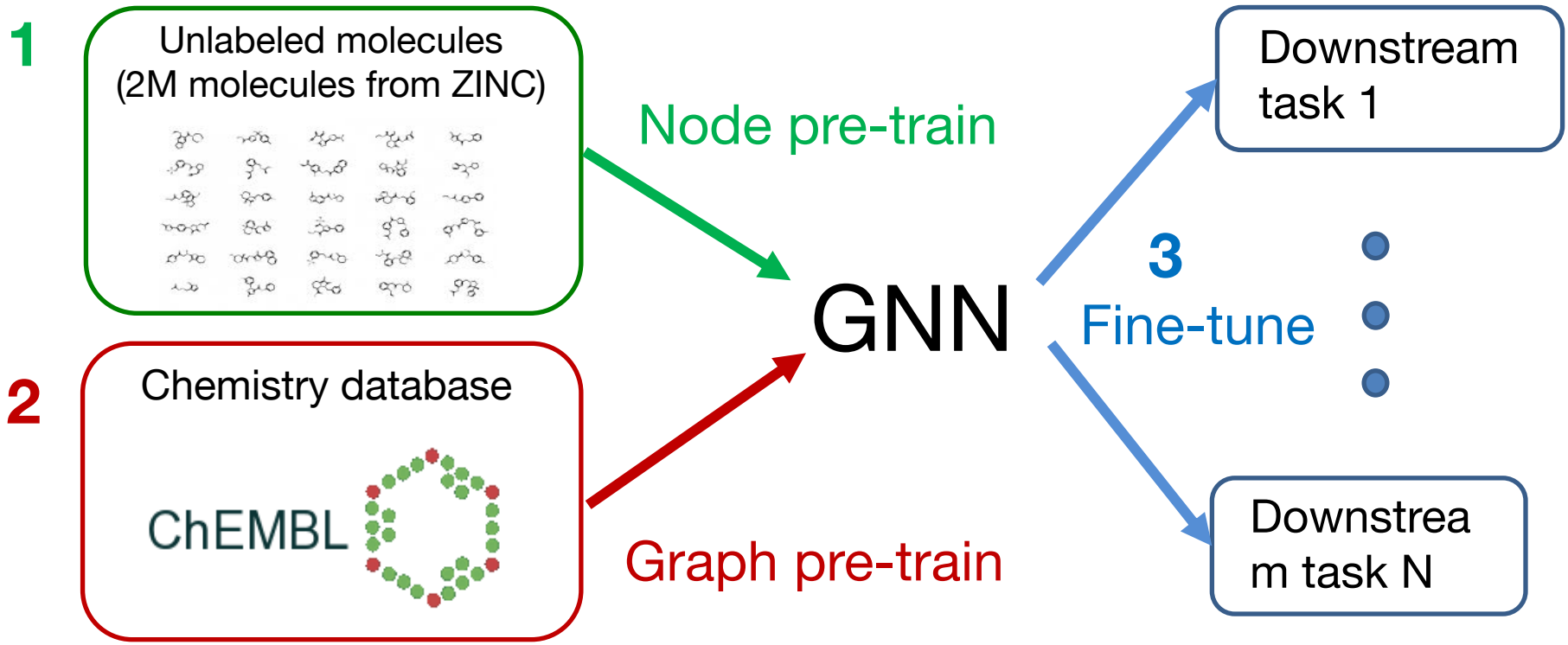
1. Multi-task prediction of many relevant labels

	Node-level	Graph-level
Attribute prediction	Attribute Masking	Supervised Attribute Prediction
Structure prediction	Context Prediction	Structural Similarity Prediction



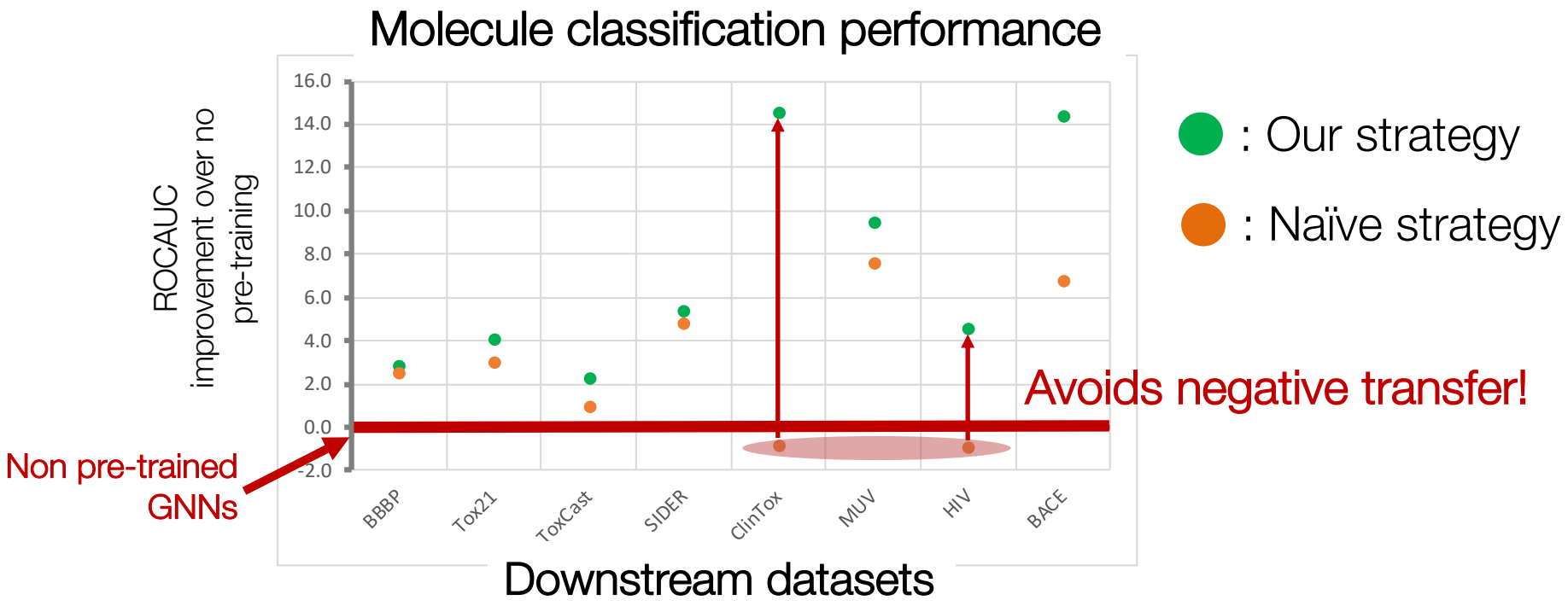
Overall Strategy

1. Node-level pre-training on unlabeled data
2. Graph-level pre-training on labeled data
3. Fine-tune on downstream data



Results of Our Strategy

- **Avoids negative transfer.**
- Significantly improve the performance.



Comparison of GNNs

Q: What are the effect of pre-training on different GNN architectures?

	Chemistry			Biology		
	Non-pre-trained	Pre-trained	Gain	Non-pre-trained	Pre-trained	Gain
GIN	67.0	**74.2**	**+7.2**	64.8 ± 1.0	**74.2 ± 1.5**	**+9.4**
GCN	**68.9**	72.2	+3.4	63.2 ± 1.0	70.9 ± 1.7	+7.7
GraphSAGE	68.3	70.3	+2.0	65.7 ± 1.2	68.5 ± 1.5	+2.8
GAT	66.8	60.3	-6.5	**68.2 ± 1.1**	67.8 ± 3.6	-0.4

A: More expressive models (GIN) benefit the most from pre-training

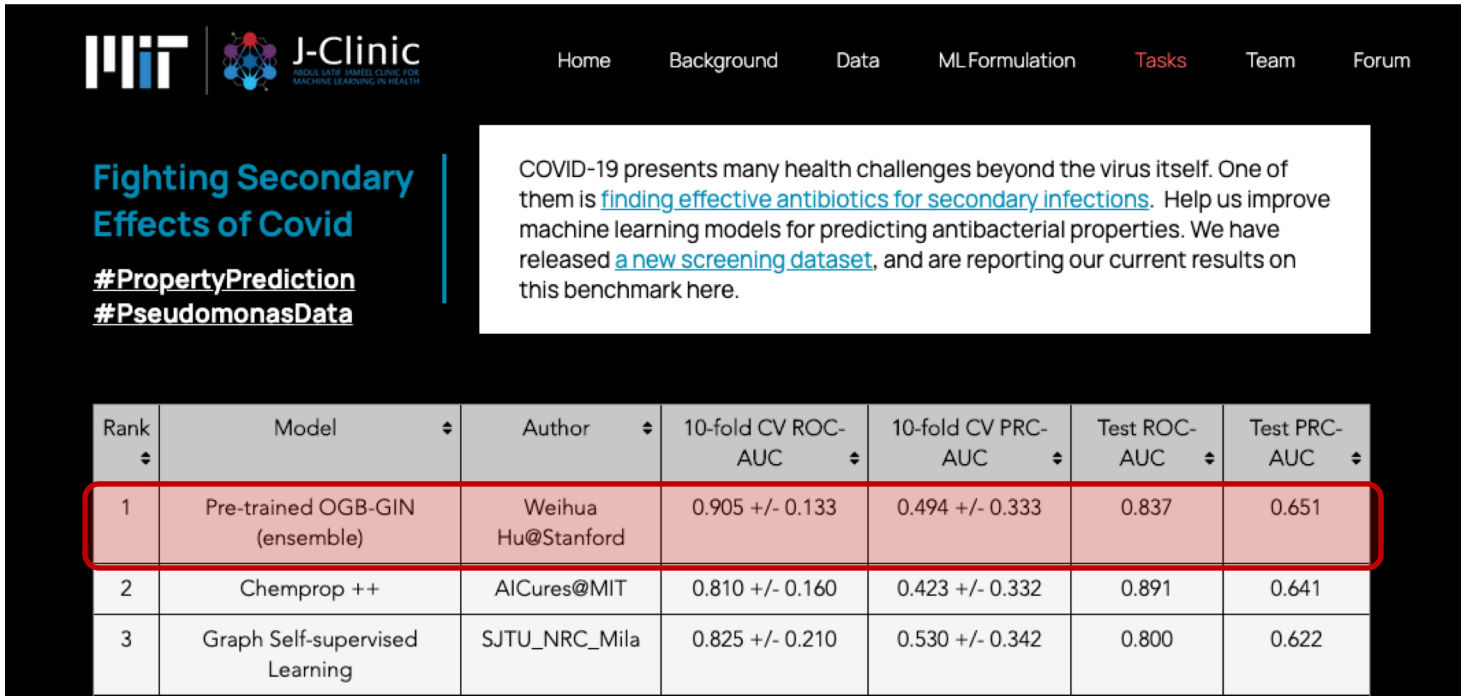
COVID-19 Challenge

- On-going initiative by MIT
- **Open task:** Molecule prediction of antibacterial properties to find antibiotics for COVID-19.
- **Dataset:** 2,335 training molecules, binary class.
- **Evaluation:** Prediction on the **hidden** test set.

https://www.aicures.mit.edu

COVID-19 Challenge

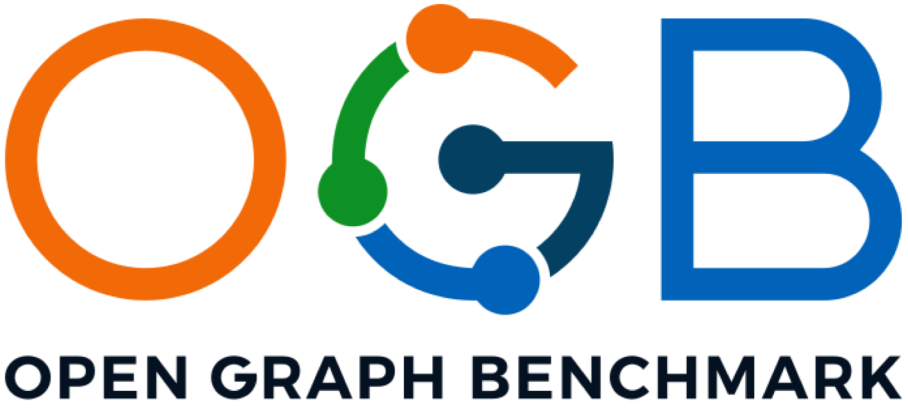
- Our pre-trained GNN ranked 1st place. The 3rd place also used pre-training.
- The model will be eventually used for virtual screening.



Rank	Model	Author	10-fold CV ROC-AUC	10-fold CV PRC-AUC	Test ROC-AUC	Test PRC-AUC
1	Pre-trained OGB-GIN (ensemble)	Weihua Hu@Stanford	0.905 +/- 0.133	0.494 +/- 0.333	0.837	0.651
2	Chemprop ++	AICures@MIT	0.810 +/- 0.160	0.423 +/- 0.332	0.891	0.641
3	Graph Self-supervised Learning	SJTU_NRC_Mila	0.825 +/- 0.210	0.530 +/- 0.342	0.800	0.622

Open Graph Benchmark

- On-going effort for large-scale realistic benchmark datasets for graph ML.



Webpage: https://ogb.stanford.edu/
Github: https://github.com/snap-stanford/ogb

Major release and paper coming in two weeks!

ML with Graphs Today

Datasets commonly used today:

- Node classification
  - CORA: 2,708 nodes, 5,429 edges
  - Citeseer: 3,327 nodes, 4,732 edges
- Graph Classification
  - MUTAG: 188 molecules
- Knowledge Graphs
  - FB15k: 15,000 nodes, 600 edges.

ML with Graphs

To properly track progress and identify issues with current approaches it is critical for our community to...

…develop diverse, challenging, and realistic benchmark datasets for machine learning on graphs

Why a New Benchmark?

1) **Current focus is on small graphs or small sets of graphs from just a handful of domains:**
   - Datasets are too small
   - Datasets do not contain rich node or edge features
   - Hard to reliably and rigorously evaluate algorithms

2) **Lack of common benchmark datasets for comparing different methods:**
   - Every paper design its own, custom train/test split
   - Performance across papers is not comparable

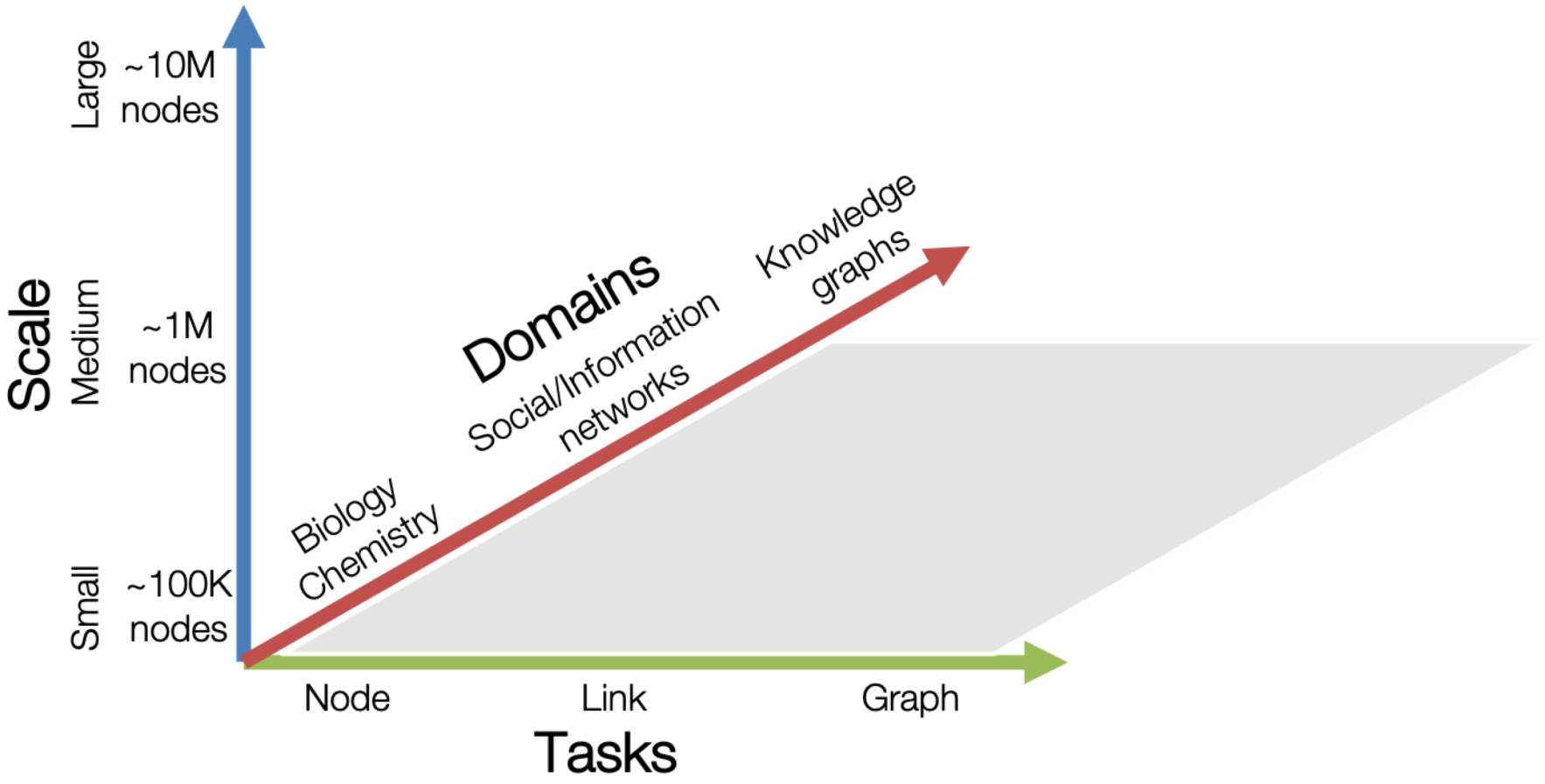
3) **Dataset splits follow conventional random splits:**
   - Unrealistic for real-world applications
   - Accuracies are over-optimistic under conventional splits

The Open Graph Benchmark

OGB is a set of benchmarks for graph ML:

1. **Ready-to-use datasets for key tasks on graphs:**
   - Node classification, link prediction, graph classification
2. **Common codebase to load, construct & represent graphs:**
   - Popular deep frameworks, e.g., DGL, PyTorch Geometric
3. **Common codebase with performance metrics for fast model evaluation and comparison:**
   - Meaningful train/validation/test splits

OGB Datasets are Diverse



Overview of Current OGB Datasets

- Covers diverse ML tasks, domains, and scales.
- More datasets to come to increase the coverage.

Task	**Node property prediction** ogbn-		
Domain	Nature	Society/Information	Knowledge graph
Small		arxiv	
Medium	proteins	products	
Large			

Task	**Link property prediction** ogbl-		
Domain	Nature	Society/Information	Knowledge graph
Small		collab	
Medium	ppa	citation	wikikg
Large			

Task	**Graph property prediction** ogbg-		
Domain	Chemistry	Biology	Computer science
Small	molhiv		
Medium	molpcba	ppi	
Large			

Meaningful Data Splits

Currently: Prediction accuracies on graph benchmarks are saturating. Generalization gap is quite small once we have enough labeled data.

Meaningful data splits focusing on generalization:

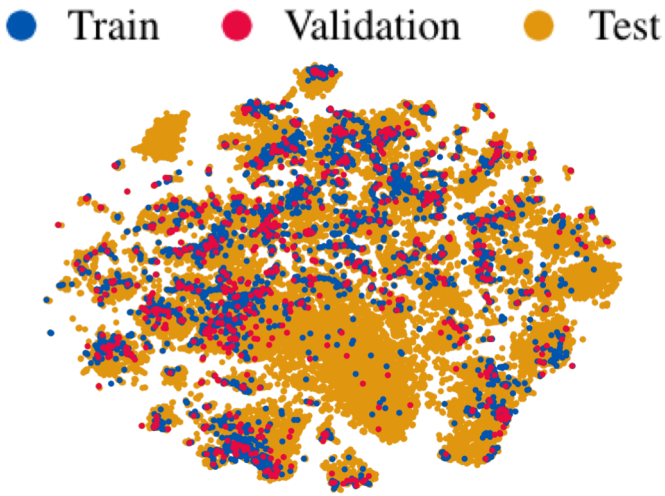
- **Scaffold split:** For molecular graph datasets, OGB:
  - Clusters molecules by scaffold (molecular graph substructure)
  - Gives validation/test sets with structurally different molecules
- **Species split:** For protein interaction datasets, OGB:
  - Uses protein graphs from model species (weed, worm, E. coli, fly, mouse, yeast, zebrafish) as train/validation sets
  - Uses protein graphs from humans as test set
- **Time split:** For the KG completion dataset, OGB:
  - Uses triplets until a certain timestamp as training/validation sets.
  - Uses newly-added triplets as test set.

Case Study: Products Dataset

Out-of-dist. pred. in Amazon product graph.

- Nodes are split according to product sales ranking.

Visualization of the split.



Performance on OGB and random splits.

Method	Train Acc (%)	Test Acc (%)
GraphSAGE (OGB split)	92.98 ± 0.16	78.03 ± 0.22
GraphSAGE (rand split)	90.47 ± 0.28	87.74 ± 0.06

- Non-random split is much more challenging than random split.

Q: How to close the generalization gap in out-of-distribution prediction over graphs?

Case Study: Products Dataset

Scalability challenge in Amazon product graph.

- 2.5M nodes.
- Full-batch GraphSAGE is **not scalable** (requires 40GB GPU memory.)
- Recent scalable GNNs are **worse than full-batch GNN.**

Method	Accuracy (%)
GraphSAGE **(full-batch)**	78.03 ± 0.22
Cluster GraphSAGE (KDD 2018)	75.18 ± 0.41
GraphSAINT (ICLR 2020)	77.29 ± 0.19

Q: How to design scalable GNNs that are as accurate as full-batch GNN?

Open Graph Benchmark

Resource for graph ML problems

- We provide pip-installable Python OGB package with loaders and evaluators.

We envision OGB to be:

- Common, community-driven platform for graph ML research
- Teaching resource

https://ogb.stanford.edu/ https://github.com/snap-stanford/ogb

Open Graph Benchmark

https://ogb.stanford.edu

ogb@cs.stanford.edu

Core development team

Weihua Hu, B. Liu, J. Dong, M. Fey, M. Zitnik, J. Leskovec

Steering committee

Regina Barzilay, Peter Battaglia, Yoshua Bengio, Michael Bronstein, Stephan Günnemann, Will Hamilton, Tommi Jaakkola, Stefanie Jegelka, Maximilian Nickel, Chris Re, Le Song, Jian Tang, Max Welling, Rich Zemel

Postdoc positions in :

(1) ML on Graphs; (2) NLP and knowledge graphs; (3) ML for biomedicine

Apply at http://snap.stanford.edu

PhD Students

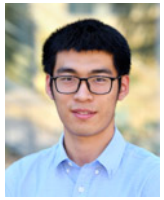

Alexandra Porter


Camilo Ruiz

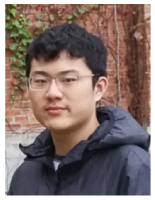

Claire Donnat

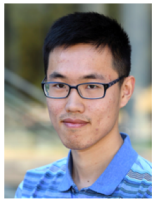

Emma Pierson


Weihua Hu


Jiaxuan You


Bowen Liu

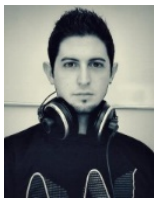

Hongyu Ren

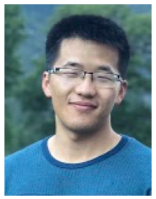

Rex Ying

Post-Doctoral Fellows


Baharan Mirzasoleiman


Marinka Zitnik

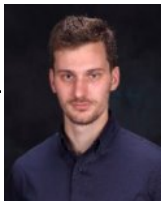

Michele Catasta


Pan Li


Shantao Li


Maria Brbic

Research Staff


Adrijan Bradaschia


Rok Sosic

Industry Partnerships

Funding

Collaborators

Dan Jurafsky, Linguistics, Stanford University
David Grusky, Sociology, Stanford University
Stephen Boyd, Electrical Engineering, Stanford University
David Gleich, Computer Science, Purdue University
VS Subrahmanian, Computer Science, University of Maryland
Marinka Zitnik, Medicine, Harvard University
Russ Altman, Medicine, Stanford University
Jochen Profit, Medicine, Stanford University
Eric Horvitz, Microsoft Research
Jon Kleinberg, Computer Science, Cornell University
Sendhill Mullainathan, Economics, Harvard University
Scott Delp, Bioengineering, Stanford University
James Zou, Medicine, Stanford University

STANFORD INFOLAB