Pre-Training Graph Neural Networks

CS224W: Machine Learning with Graphs

Jure Leskovec, Stanford University

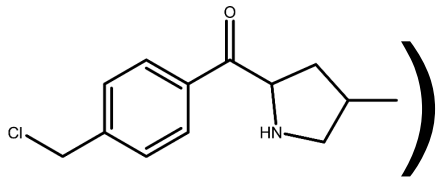
http://cs224w.stanford.edu



Graph ML in Scientific Domains

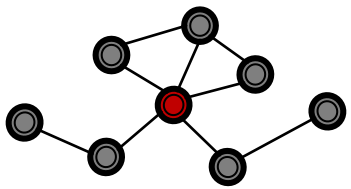
- **Chemistry**: Molecular graphs
  - Molecular property prediction

Our running example today

f(

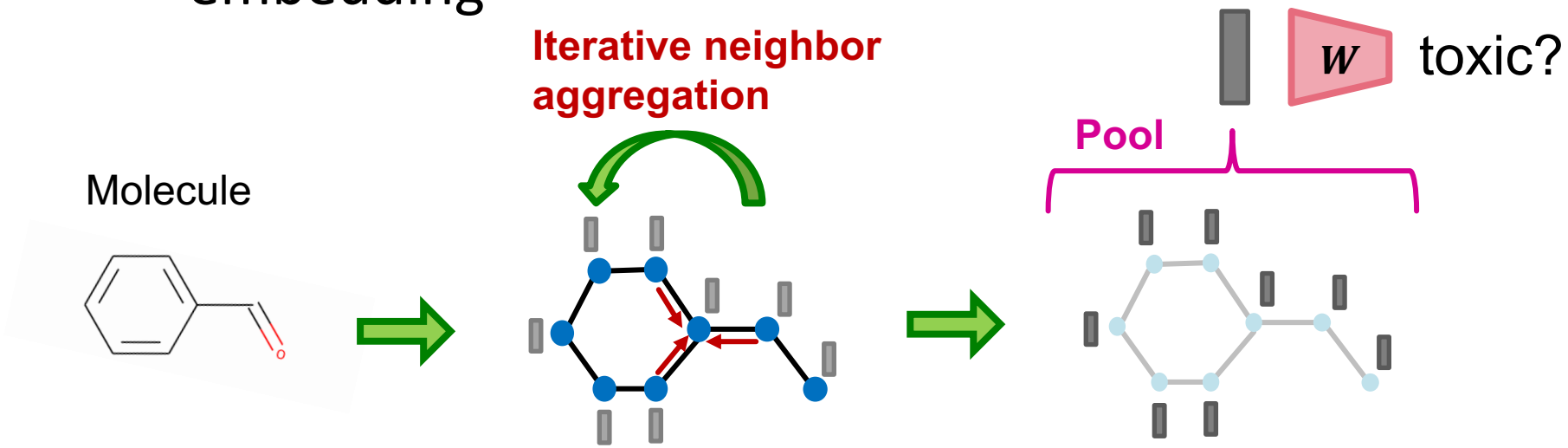
) = toxic?

- **Biology**: Protein-protein association graphs
  - Protein function prediction

f() = biological activity?

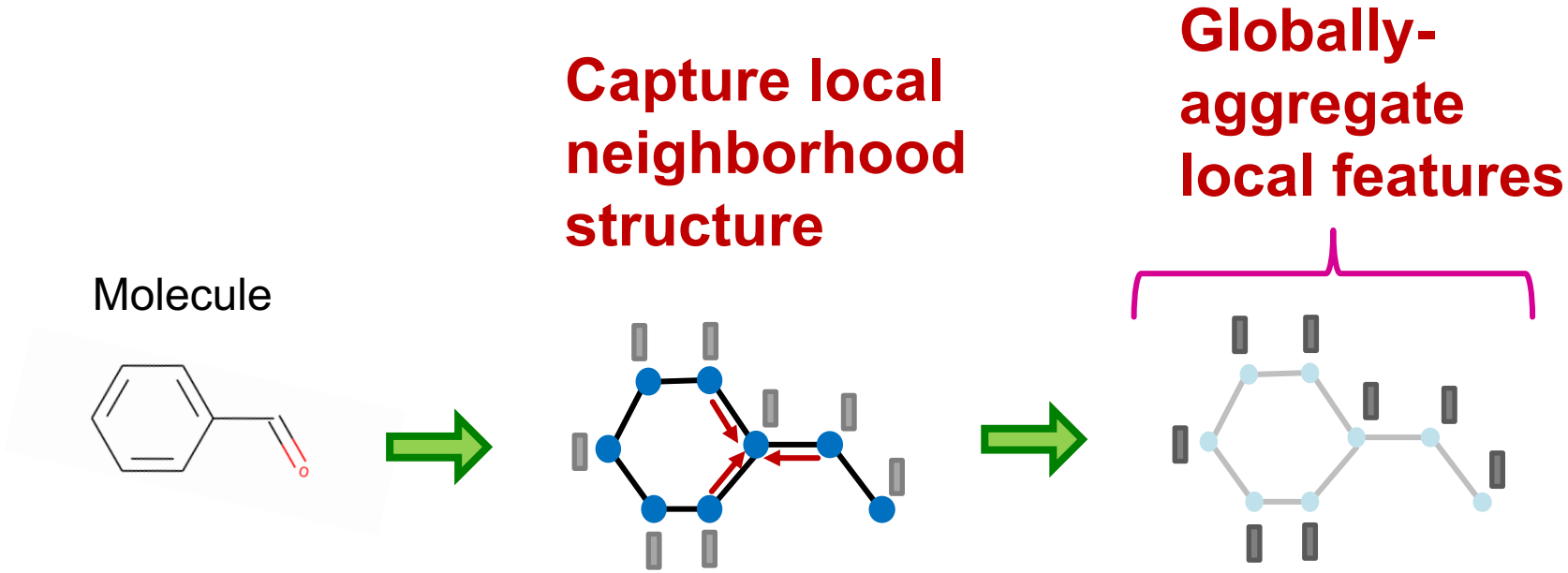
GNNs for Graph Classification

- GNNs obtain an embedding of an entire graph by following two steps
  - **Iteratively aggregate neighboring information** to obtain node embeddings
  - **Pool node embeddings** to obtain a graph embedding



GNNs for Graph Classification

- Node embeddings capture **local neighborhood structure**
- The embedding of an entire graph is a **global aggregation** of such node embeddings



Challenges of Applying ML

- Two fundamental challenges in applying ML to scientific domains

1. Scarcity of labeled data

- Obtaining labels requires expensive lab experiments

  → GNNs overfit to small training data

2. Out-of-distribution prediction

- Test examples tend to be very different from training examples

  → GNNs extrapolate poorly

Challenges for Deep Learning (1)

- **Deep learning models have a lot parameters to train (e.g., in the order of millions).**
- **#(Labeled training data) << #(Parameters)**
- **Deep learning models are extremely prone to overfitting on small labeled data.**

Challenges for Deep Learning (2)

- **Deep learning models extrapolate poorly**
  - Models often make predictions based on spurious correlations in a dataset [Sagawa et al. ICML 2020]
  - Ex) Image classification between "polar bear" and "brown bear"
  - During training:
    - Most "polar bears" have the snow background
    - Most "normal bears" have the grass background
    - **Model can learn to make prediction based on the image background, rather than the animal itself.**
  - At test time, what if we see "polar bear" on the grass?





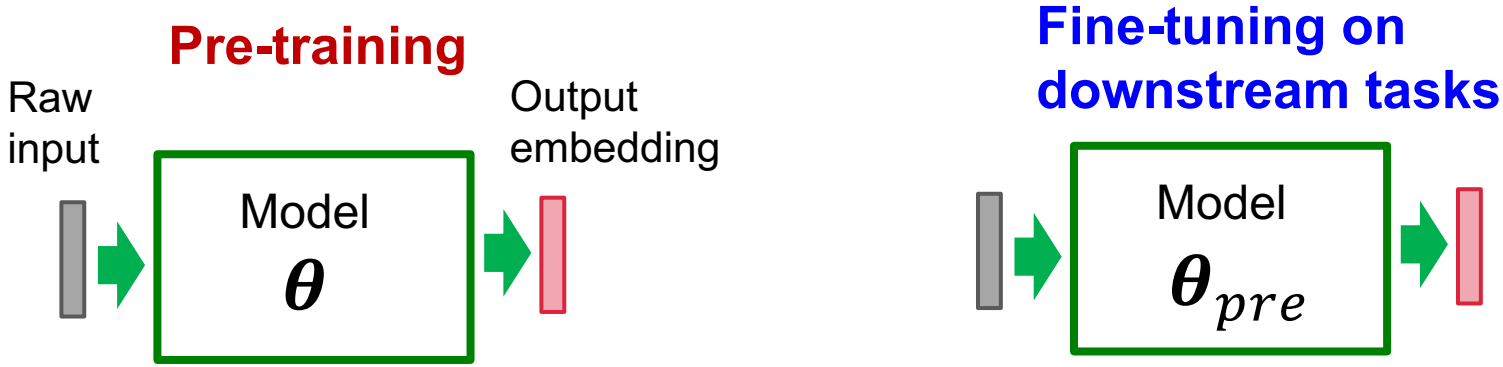
Adapted from Wikipedia

Injecting Domain Knowledge

- **Goal: Improve model's out-of-distribution prediction performance even with limited data.**
- **Key idea**: Inject domain knowledge into a model before training on scarcely-labeled tasks!
  - The model already knows the domain knowledge before training on data
  - So that the model can
    - Generalize well without much task-specific labeled data
    - Extract essential (non-spurious) pattern that allows better extrapolation.

Effective Solution: Pre-Training

- We **pre-train a model relevant tasks**, where data is abundant.
  - After pre-training, the model parameters already contain domain knowledge.
- **For downstream tasks (what we care about, typically with small #labeled data)**
  - **We start from the pre-trained parameters and fine-tuning them.**



Pre-Training is Successful

- **Pre-training has been hugely successful in computer vision and natural language processing.**
  - Pre-training improves label-efficiency.
  - Pre-training improves out-of-distribution performance [Hendrycks et al. ICML 2019]
- Pre-training is a powerful solution to the two ML challenges in scientific applications
  - Scarce labels
  - Out-of-distribution prediction

Pre-training GNNs

Hu et al. ICLR 2020

- Let's consider pre-training GNNs!
- We design GNN pre-training strategies and systematically investigate

Q1. How effective is pre-training GNNs?

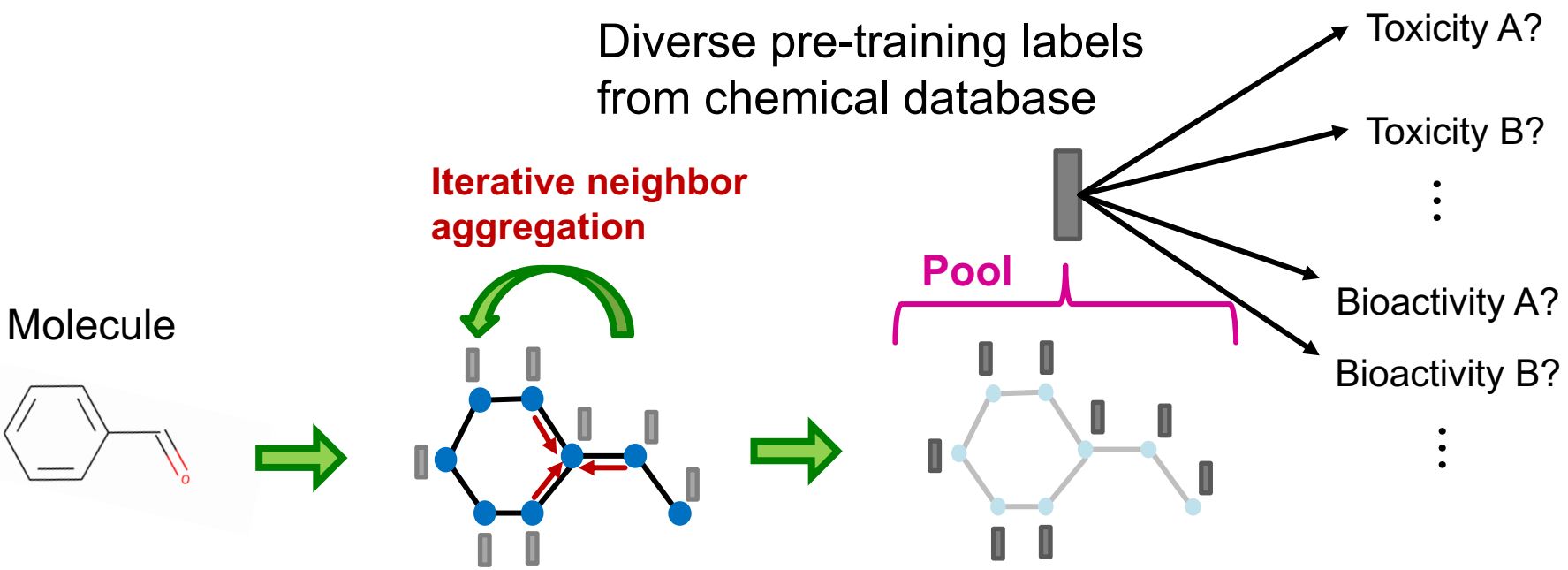
Q2. What is the effective pre-training strategy?

How Effective is Pre-training GNNs?

Let's think about molecular property prediction for drug discovery.

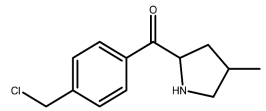
- **Naïve strategy**

Multi-task supervised pre-training on relevant labels.



Experimental Setting

- **Molecule classification**
  - **Task**: Binary classification. ROC-AUC as metric

$$f(\text{  }) = \{0,1\}$$

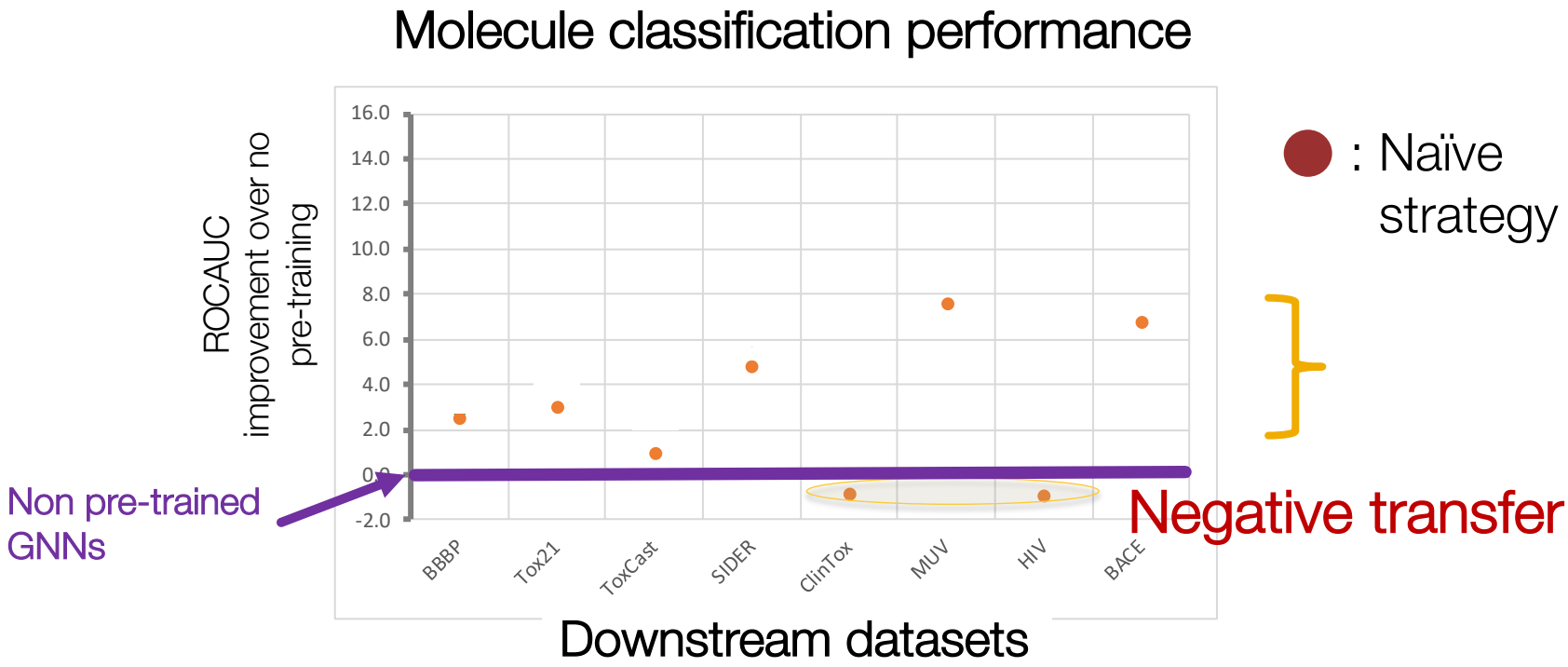
  - **Supervised pre-training data**
    - 1310 diverse binary bioassays annotated over ~450K molecules
  - **Downstream task** (what we care about!)
    - 8 molecular classification datasets (relatively-small, 1K—100K molecules)
  - **Data split**: Scaffold (test molecules are out-of-distribution)

How Effective is Pre-training GNNs?

- **Naïve strategy**:

Multi-task supervised pre-training on relevant labels.

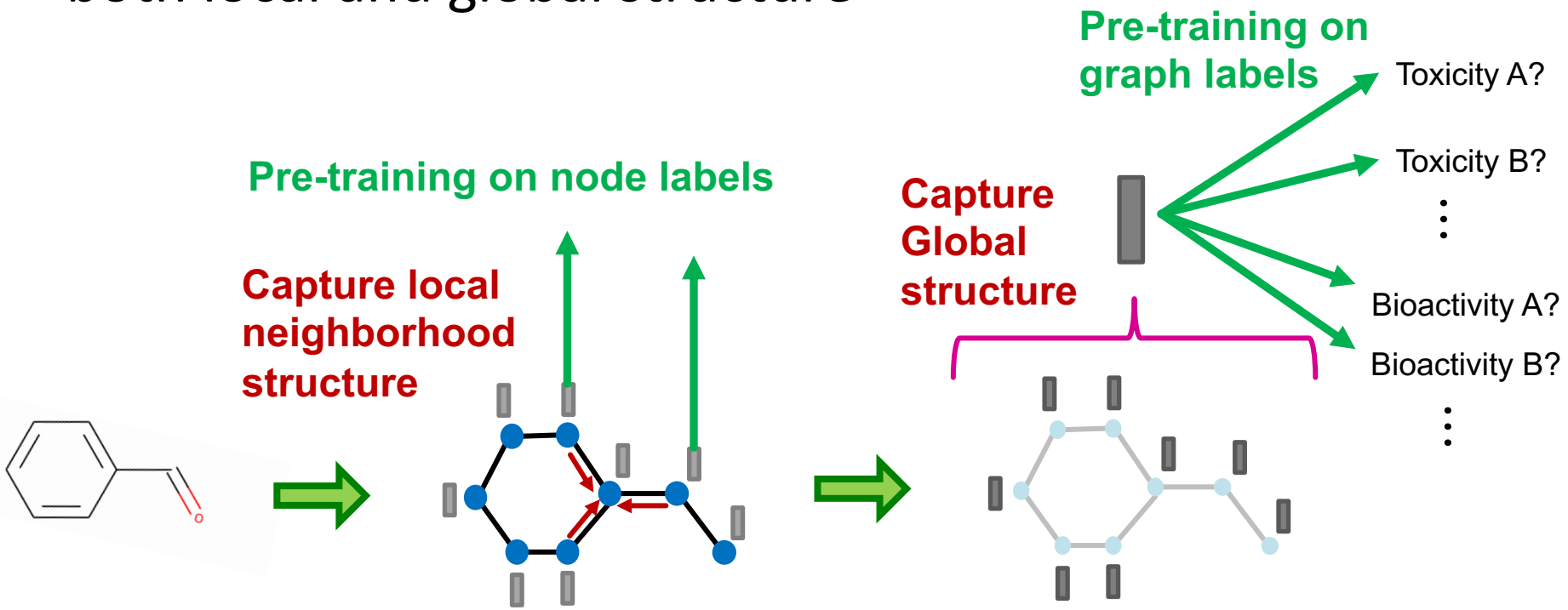
→ Limited performance improvement on downstream tasks. Often leads to negative transfer



What is the Effective Strategy?

- **Key idea**: **Pre-train both node and graph embeddings.**

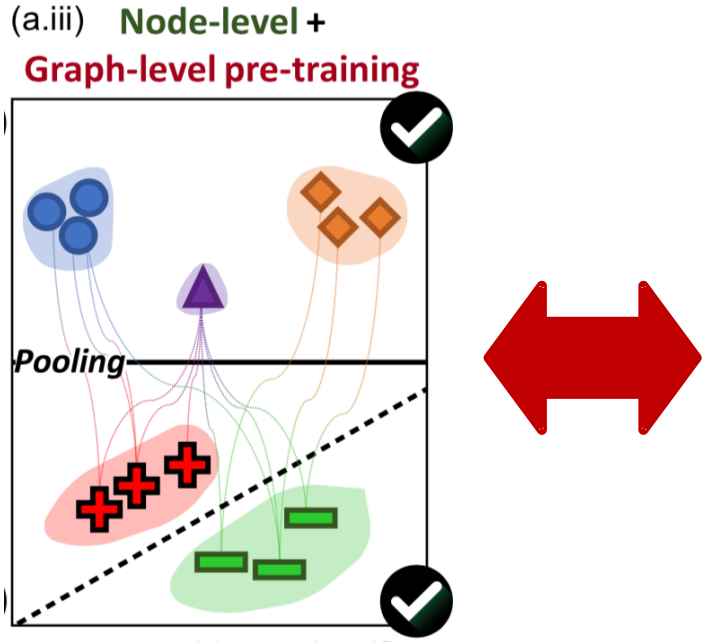
→ GNN can capture domain-specific knowledge of both local and global structure



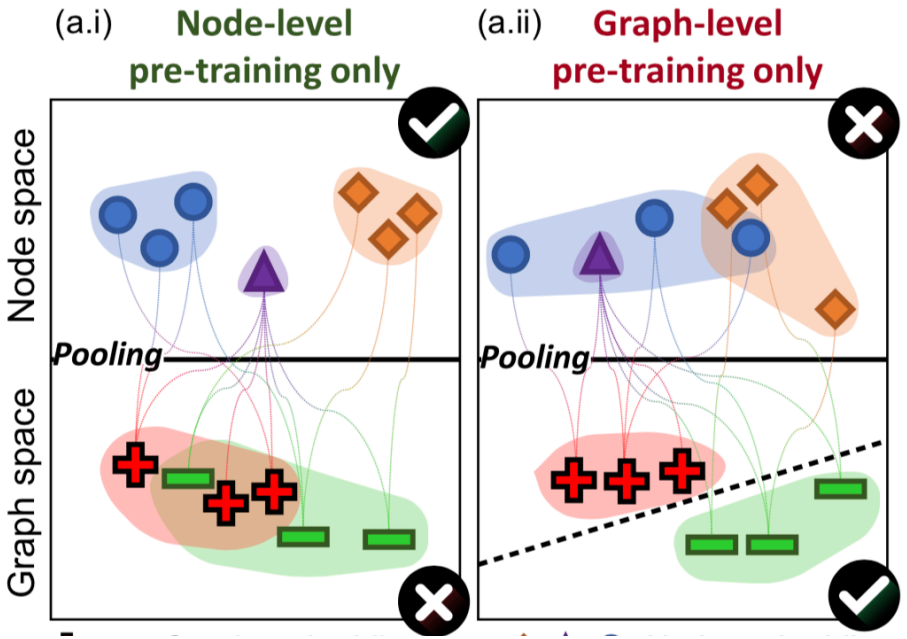
What is the Effective Strategy?

- **Key idea**: **Pre-train both node and graph embeddings.**

Embedding illustration



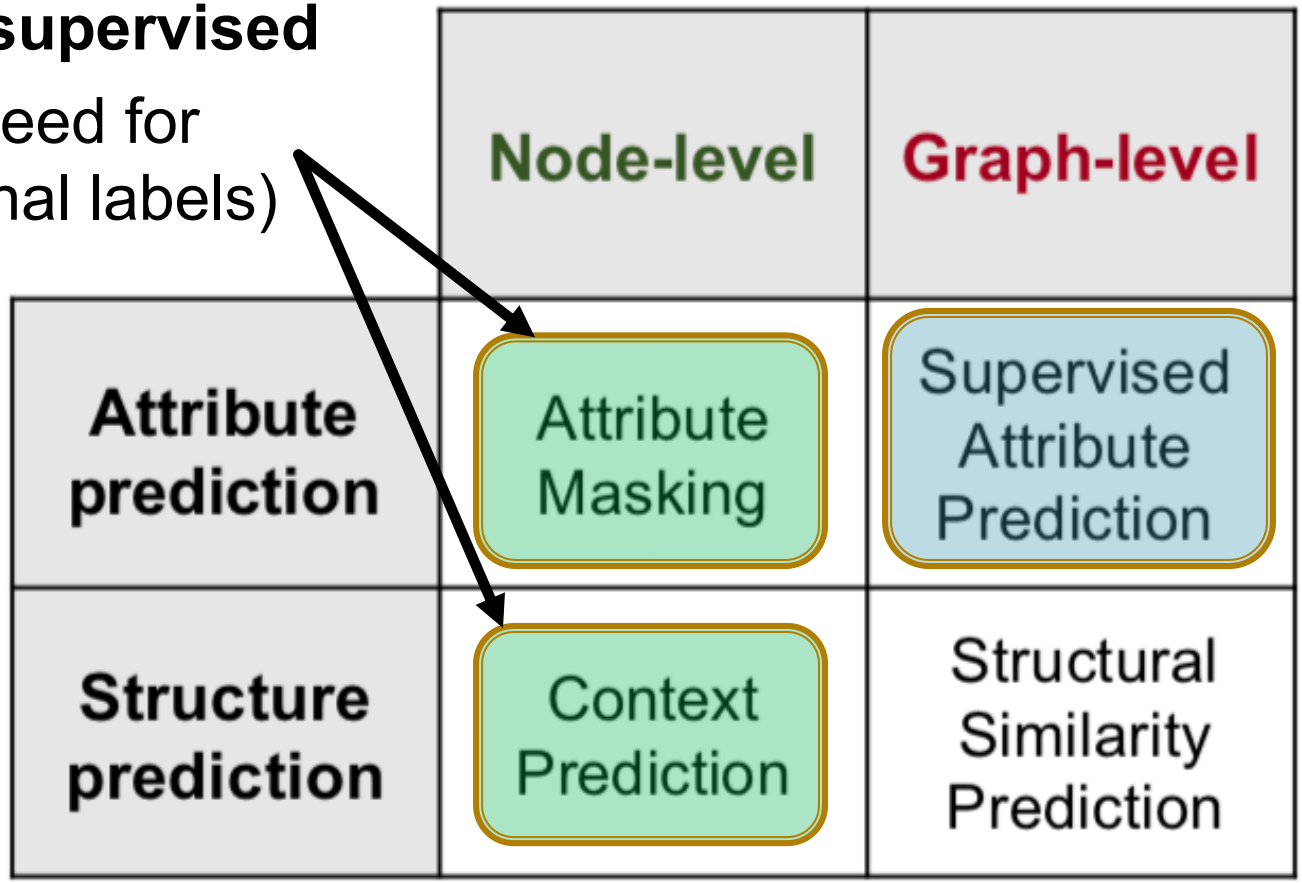
Naïve strategies



Proposed Pre-Training Methods

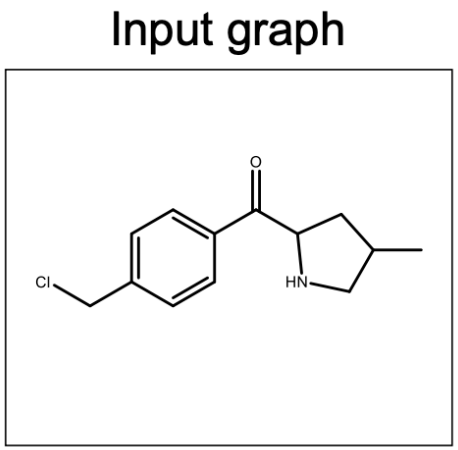
Self-supervised

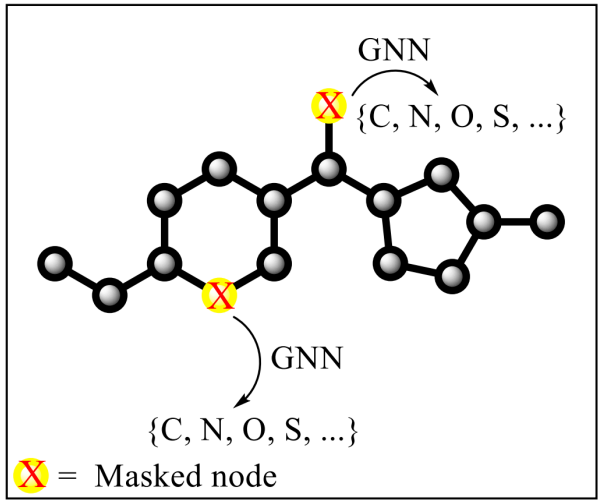
(No need for external labels)



Attribute Masking: Algorithm

- Mask node attributes
- Use GNNs to generate node embeddings.
- Use the embeddings to predict masked attributes.

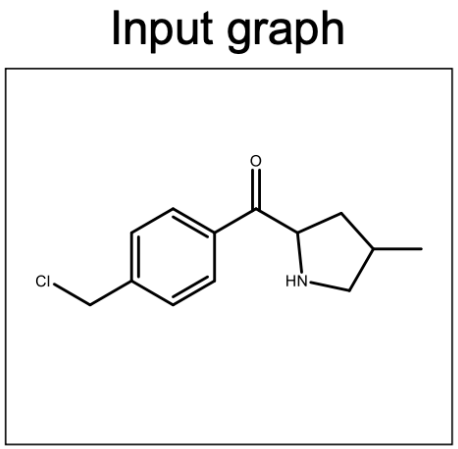


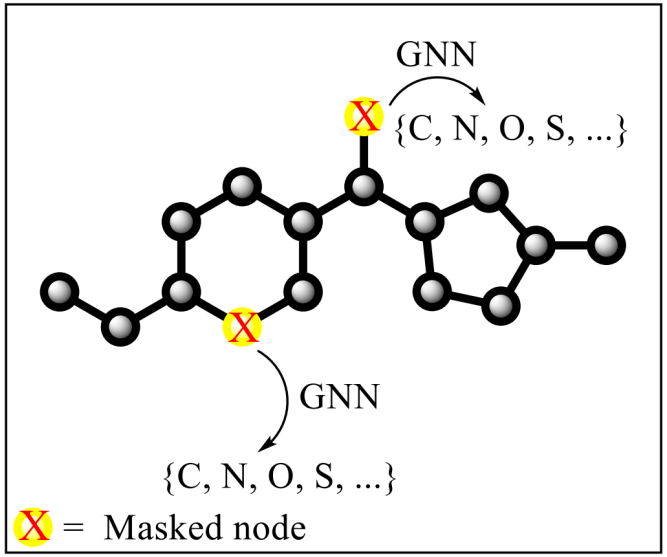


Attribute Masking: Intuition

Intuition

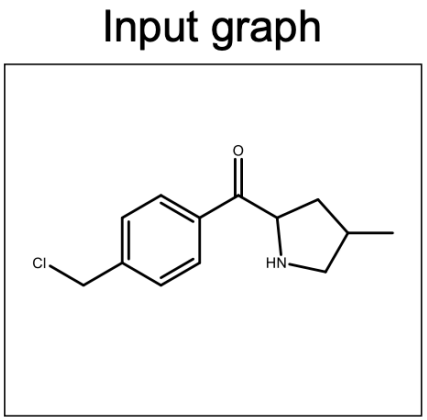
- Through solving the masked attribute prediction task, a GNN is forced to learn domain knowledge, .e.g., chemical rules.

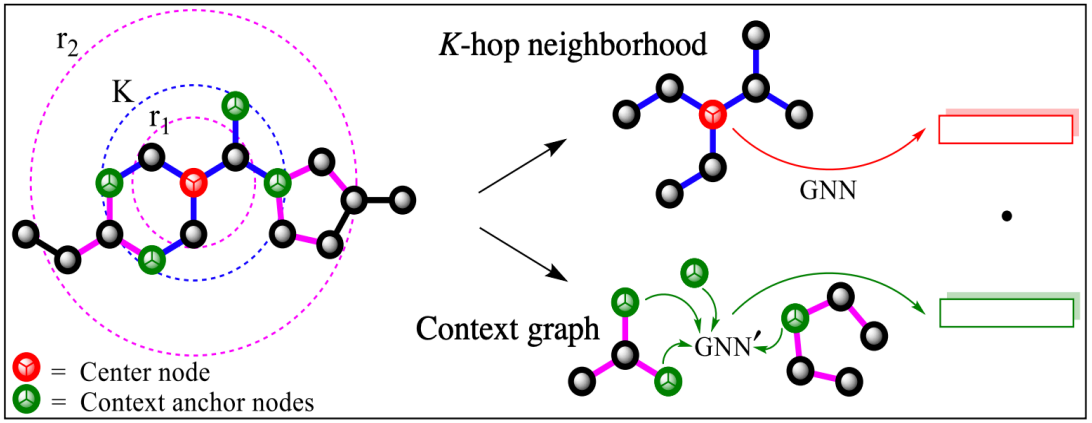




Context Prediction: Algorithm

- For each graph, sample one center node.
- Extract neighborhood and context graphs.
- Use GNNs to encode neighborhood and context graphs into vectors.
- Maximize/minimize the inner product between true/false (neighborhood, context) pairs.



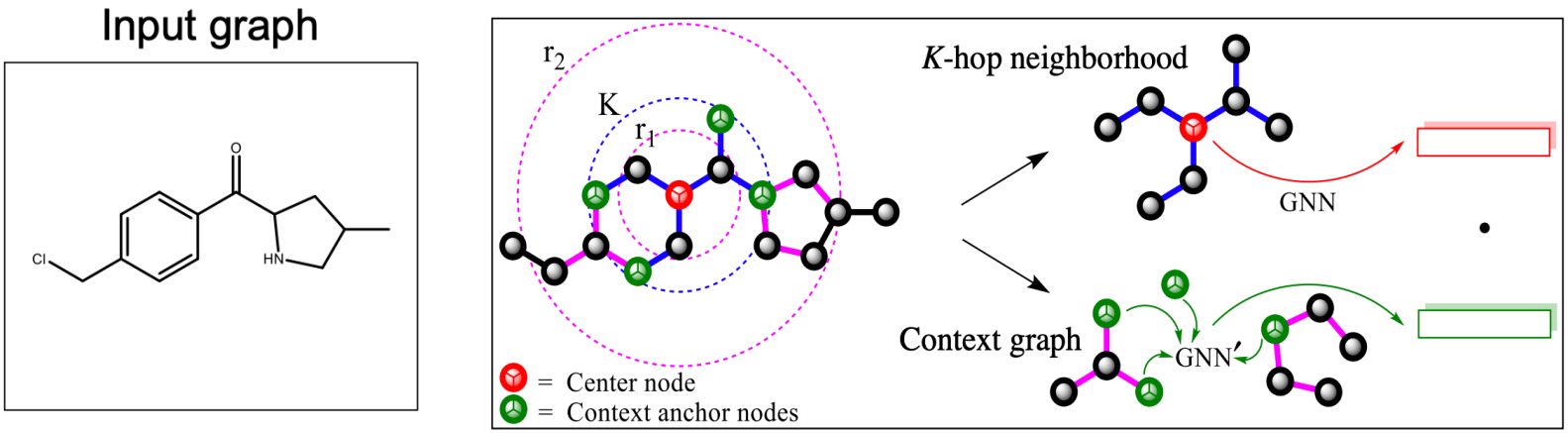


Context Prediction: Intuition

- **Intuition**

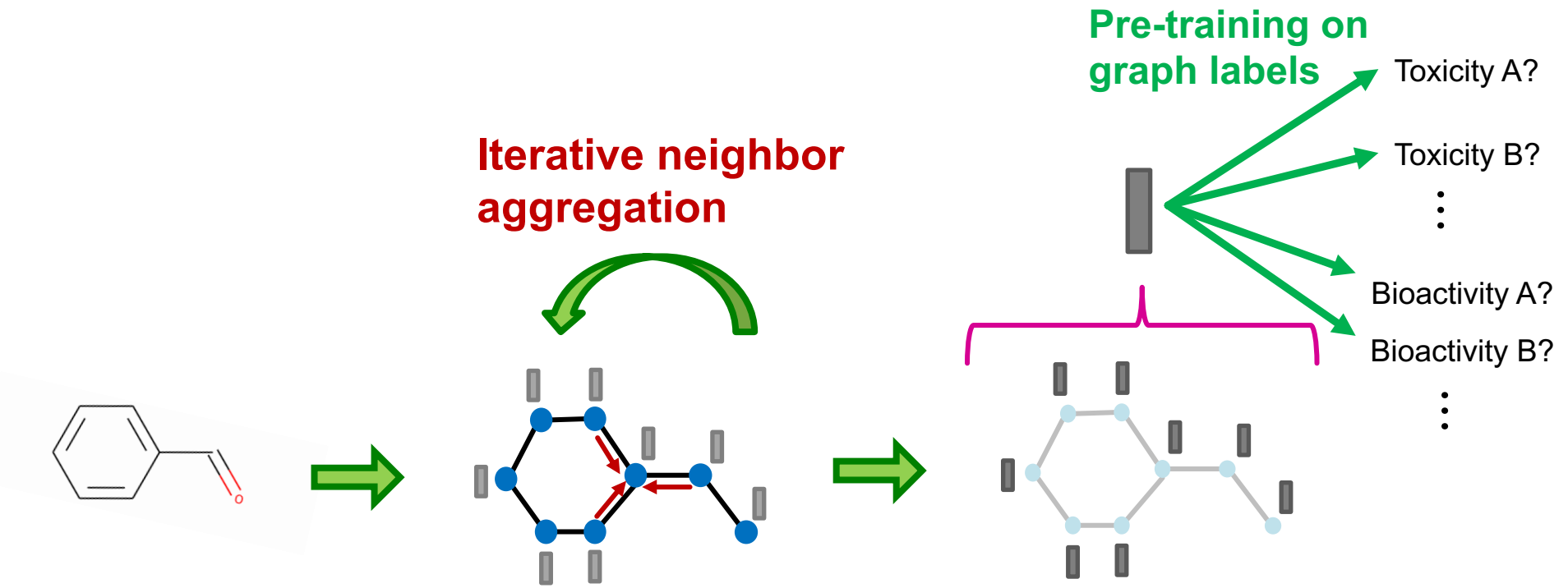
Subgraphs that are surrounded by similar contexts are semantically similar.

  - **In natural language processing, this is called distributional hypothesis, and is exploited in the word2vec model** [Mikolov et al. NIPS 2013].



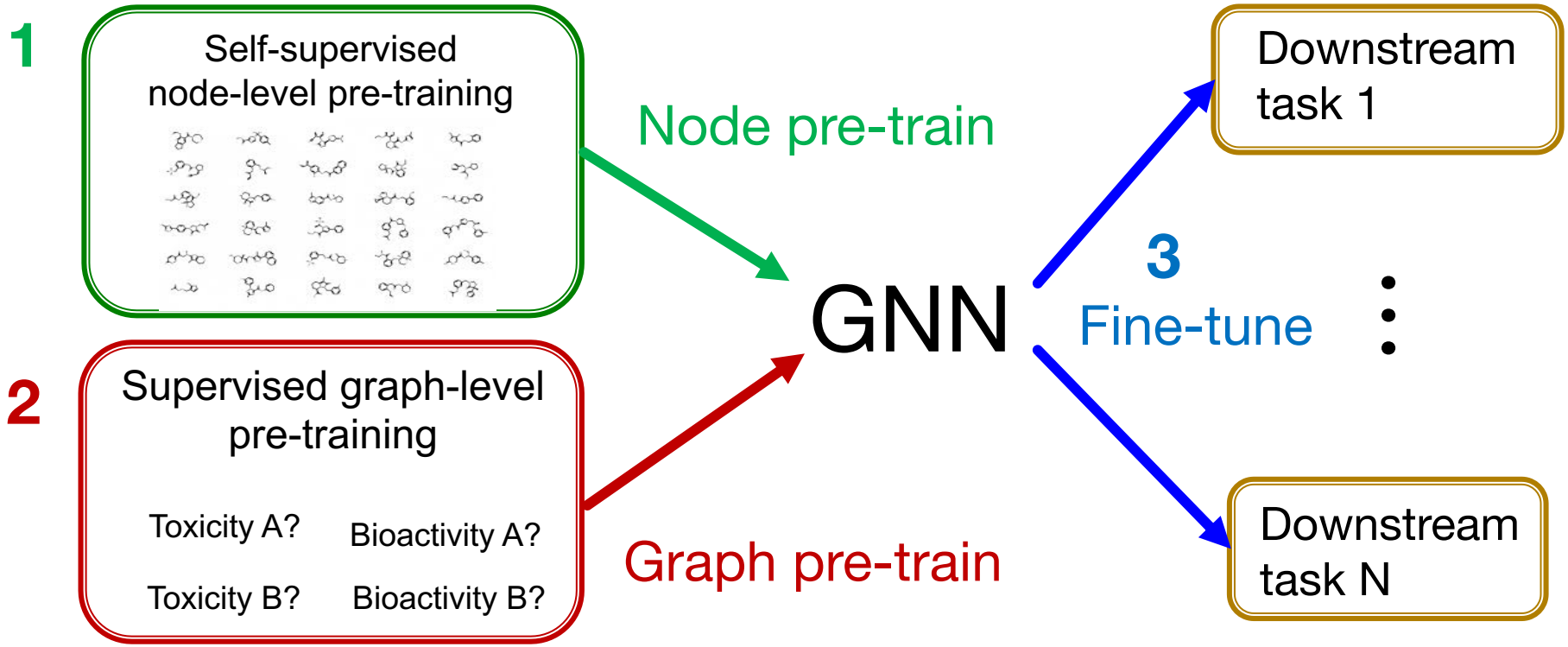
Supervised Attribute Prediction

- Multi-task supervised training on many relevant labels.



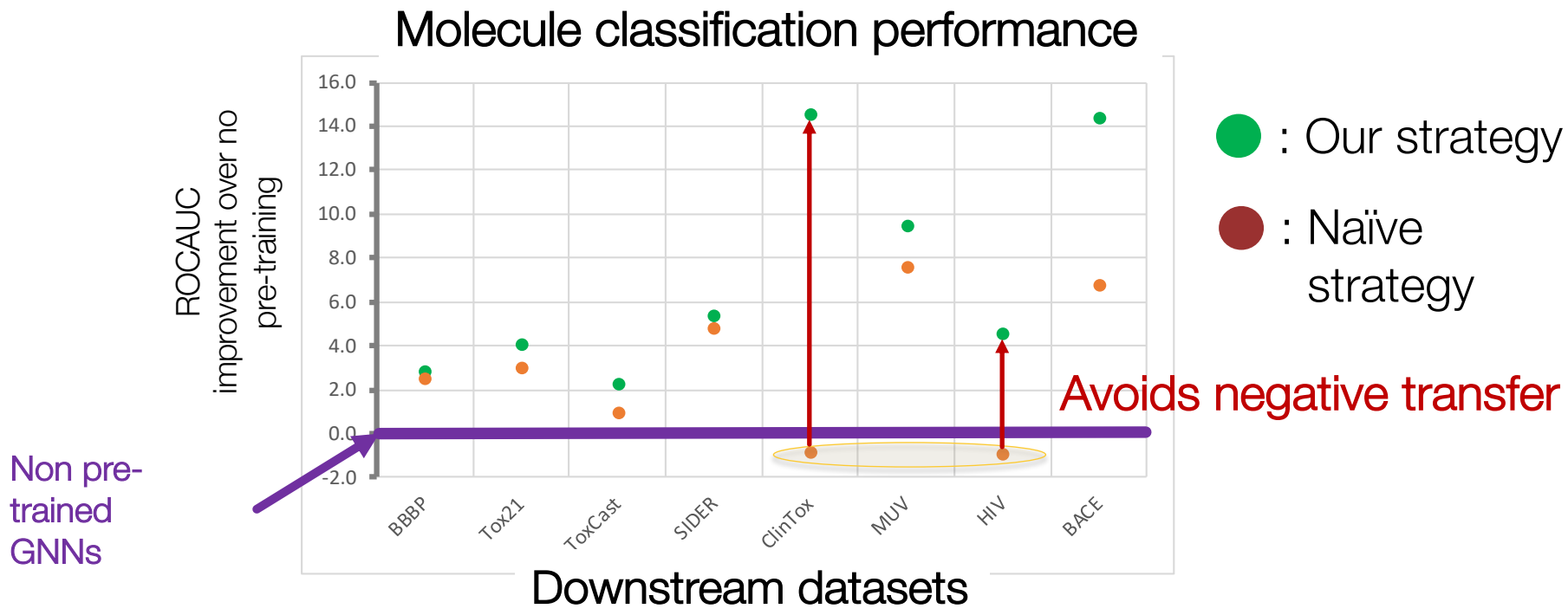
Overall Strategy

1. Node-level pre-training
2. Graph-level pre-training
3. Fine-tuning on downstream tasks



Results of Our Strategy

- Avoids negative transfer.
- Significantly improve the performance.



Comparison of GNN models

- When different GNN models are pre-trained, **the most expressive model (GIN) benefits the most from pre-training.**
- **Intuition: Expressive model can learn to capture more domain knowledge than less expressive models.**

	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

Pre-Training GNNs: Summary

- GNNs have important applications in scientific applications, but they present challenges of
  - **Label scarcity**
  - **Out-of-distribution prediction**
- **Pre-training** is promising to tackle the challenges.
- However, naïve pre-training strategy gives sub-optimal performance and **even leads to negative transfer.**
- **Our strategy: Pre-train both node and graph embeddings** → Leads to significant performance gain on downstream tasks.

Hyperbolic Graph Embeddings

CS224W: Machine Learning with Graphs

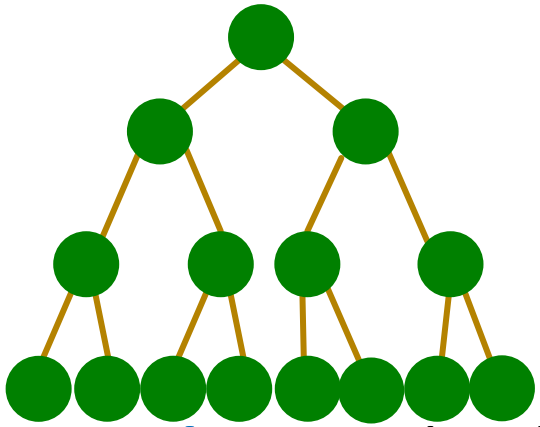
Jure Leskovec, Stanford University

http://cs224w.stanford.edu



Hyperbolic Graph Embeddings

- In previous lectures, we focus on graph representation learning in **Euclidean embedding spaces** $\mathbb{R}^n$
- However, Euclidean embeddings cannot always capture complex graph structures
- Example: Tree structure

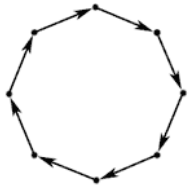


Often Exponential increase in number of nodes as depth increases

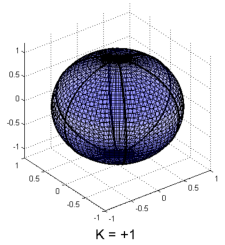
- **Tree-like graphs**: graphs that are similar to a tree, but can contain very few cycles

Graph Embedding Geometry

- Graphs with many large cycles
  - Best with **spherical** space embeddings

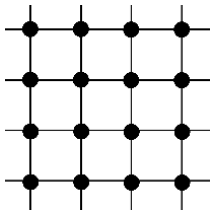


Embed

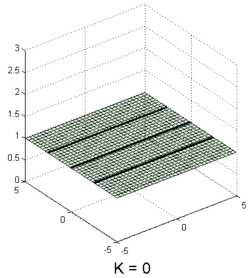


+ curvature

- Grid-like graphs
  - Best with **Euclidean** space embeddings



Embed

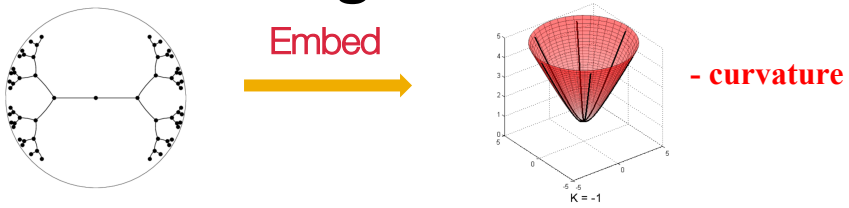


o curvature

- **Hierarchical, tree-like** graphs?

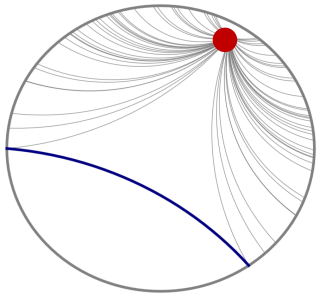
Hyperbolic Embedding Space

- Modeling **Hierarchical, tree-like** graphs
- Hyperbolic space: an alternative embedding space to model embeddings for trees with tree-like structure
- **Advantage**: hyperbolic embedding properties (explained later) allow it to naturally model trees
- **Challenges**: existing deep learning and optimization toolkits can only be applied to hyperbolic embeddings



Hyperbolic Space Model

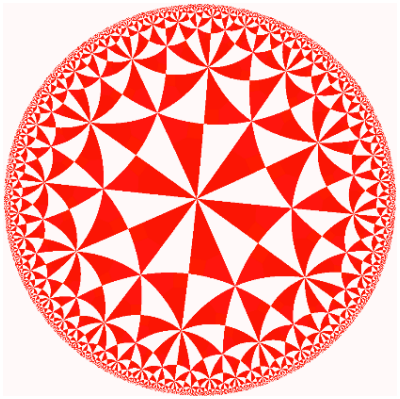
- **Hierarchical, tree-like** graphs
- Hyperbolic geometry differs from Euclidean geometry by the **parallel postulate (5th axiom)**
- In Euclidean geometry:
  In a plane, given a line and a point not on it, at most one line parallel to the given line can be drawn through the point.
- However, in hyperbolic geometry, there are infinite number of lines parallel to the given line through the point



Set of geodesic lines from the red point to boundary of the Poincare ball that are parallel to the blue line

Hyperbolic Space Model

- Hyperbolic space cannot be naturally represented in Euclidean space due to its special properties
- We commonly use 2 geometry models to **represent or visualize hyperbolic spaces**
- They are equivalent: there exists mappings between these spaces
- Poincaré Model
  - Radius proportional to $\sqrt{K}$ (K: inverse of curvature)
  - Open ball (exclude boundary)
  - Each triangle in the figure has the **same** area



Poincaré model: intuitive visualization

Hyperbolic Space Model

- Hyperboloid Model (Lorentz Model)
  - Upper sheet of 2-sheet hyperboloid
  - Subset of Euclidean space
- Hyperboloid Model is numerically stable
- In comparison, Poincaré Model needs to represent exponentially many embedding points close to the boundary of the Poincare ball



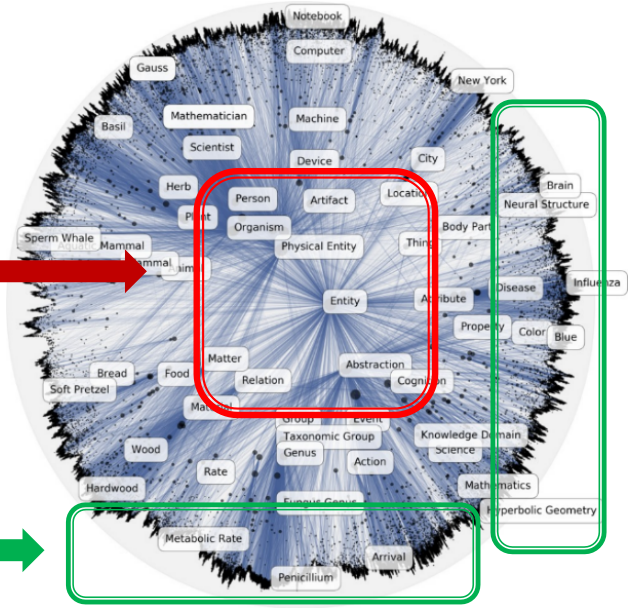
Hyperboloid: numerically more stable, simpler formula

Task

- Graph representation learning on **hierarchical** graphs
  - Link Prediction
  - Node Classification

Nodes in high hierarchy

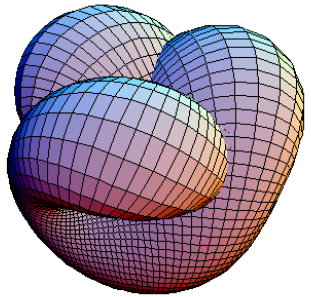
Nodes in low hierarchy



Poincaré embedding (Nickel et. al.)

Hyperbolic Geometry (1)

- Manifold: high-dimensional surface
- Riemannian Manifold
  - Equipped with
    - *Inner product* $\langle .,. \rangle$: metric space
    - *Tangent space* $\mathcal{T}_x$: an $\mathbb{R}^n$ that approximates the manifold at any point x
  - Both functions vary smoothly (differentiable) on the manifold



Hyperbolic Geometry (2)

- Geodesic: shortest path in manifold
  - Analogous to straight lines in $\mathbb{R}^n$
- Hyperbolic space is a Riemannian manifold with constant negative curvature $-\frac{1}{K}$, where $(K > 0)$
  - Becomes Euclidean when $K \to \infty$
  - The more negative is the curvature, the more curved the space is

How does curvature affect the embedding space?

Hyperbolic Geometry Models

- We use $-\frac{1}{K}$ to denote the **curvature**
- $\sqrt{K}$ is also the **radius** of the Poincaré ball
- Hyperboloid model as a Riemannian manifold:
  - Constant **Minkowski inner product**

$$\langle .,. \rangle_{\mathcal{L}} : \mathbb{R}^{d+1} \times \mathbb{R}^{d+1} \to \mathbb{R}$$



$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathcal{L}} = -x_0 y_0 + x_1 y_1 + \ldots + x_d y_d = -\frac{1}{K} \; (K > 0)$$

Time-like

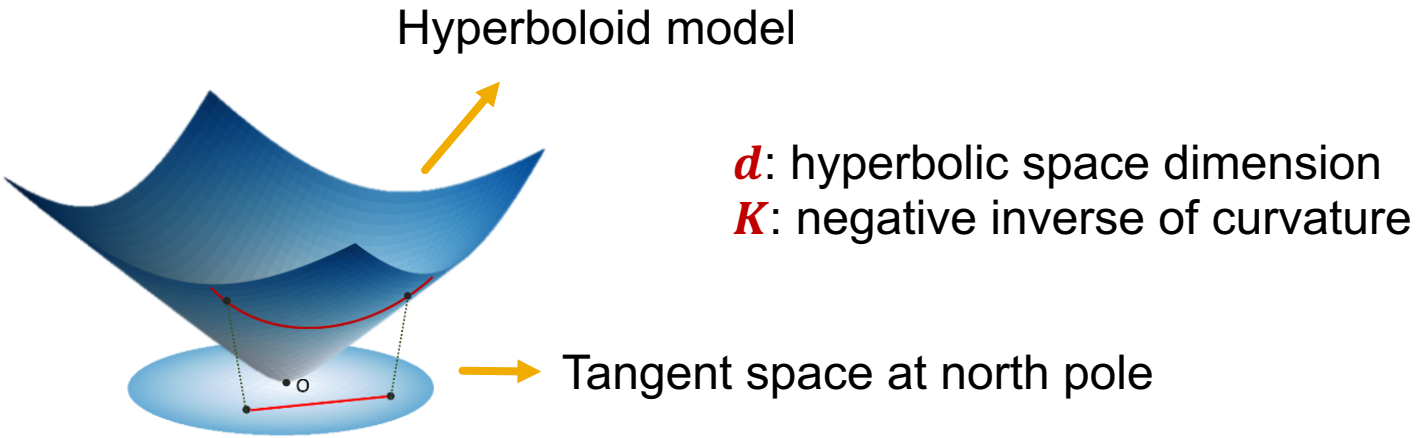
Space-like

- **Distance** between two hyperbolic points x, y:

$$d_{\mathcal{L}}^{K}(\mathbf{x}, \mathbf{y}) = \sqrt{K}\,\mathrm{arcosh}(-\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} / K)$$

Tangent Space

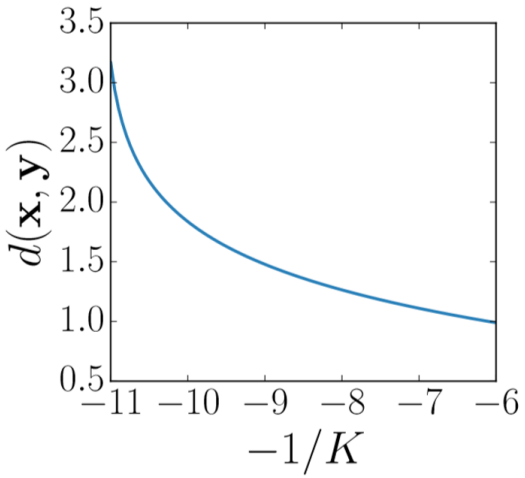
- **Tangent space** expression under hyperboloid model
  - $\mathcal{T}_x \mathbb{H}^{d,K} = \{\boldsymbol{v} \in \mathbb{R}^{d+1} : \langle \boldsymbol{v}, \boldsymbol{x} \rangle_{\mathcal{L}} = 0\}$



$\boldsymbol{d}$: hyperbolic space dimension
$\boldsymbol{K}$: negative inverse of curvature

Geodesic Distance (1)

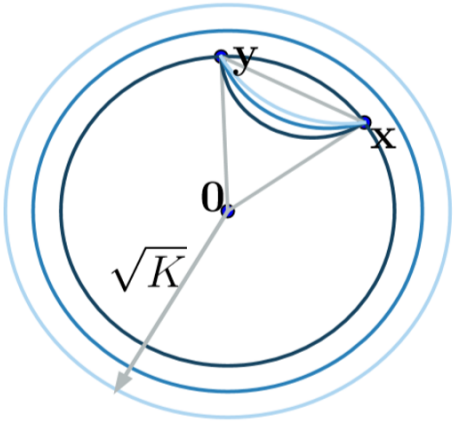
- Geodesic distance in hyperbolic space is analogous to the shortest path distance in Euclidean space
- **"Straight line"** in hyperbolic space
- The more negative the curvature:
  - the more geodesics bends inward
  - distance between **x** and **y** also increases:



Distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ increases as curvature becomes more negative

Geodesic Distance (2)

- Geodesic distance between two hyperbolic points will bend inward more as curvature becomes more negative
- **Larger curvature** results in **smaller radius** of the Poincare ball
- Distance between the same coordinates (x, y) increases as the radius gets smaller



Dark blue: high curvature boundary and geodesics
Light blue: low curvature boundary and geodesics

Mapping to and from Tangent Space

- Exponential map: from tangent space (Euclidean) to manifold
- Logarithmic map: inverse operation of exponential map
- Norm $\|v\|_{\mathcal{L}} = \langle v, v \rangle_{\mathcal{L}}$

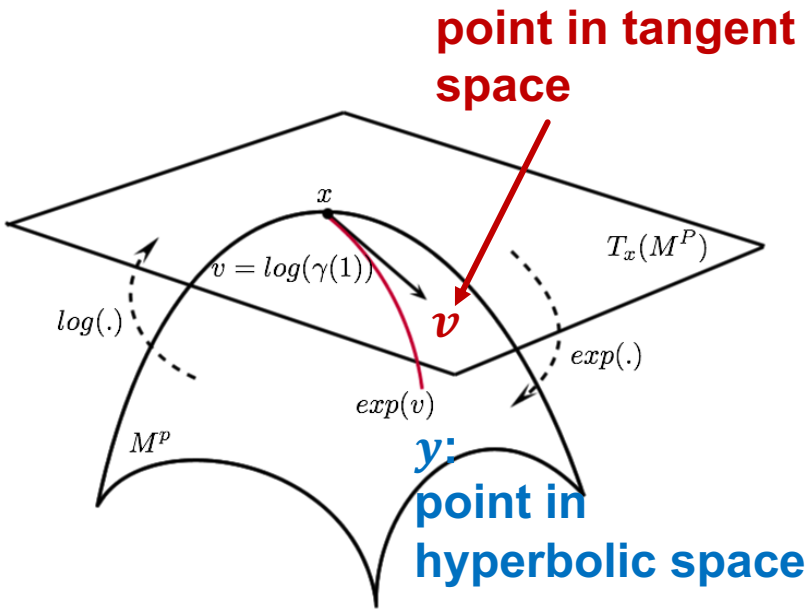
Hyperbolic trigonometry

Exp map

$$\exp_{\mathbf{x}}^{K}(\mathbf{v}) = \cosh\left(\frac{||\mathbf{v}||_{\mathcal{L}}}{\sqrt{K}}\right)\mathbf{x} + \sqrt{K}\sinh\left(\frac{||\mathbf{v}||_{\mathcal{L}}}{\sqrt{K}}\right)\frac{\mathbf{v}}{||\mathbf{v}||_{\mathcal{L}}}$$

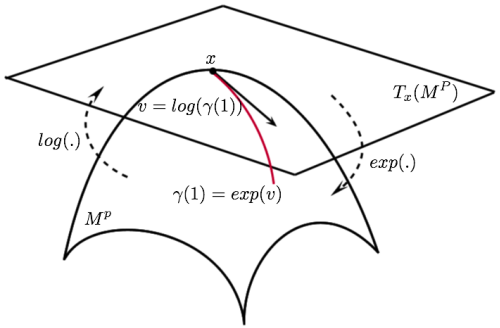
Log map

$$\log_{\mathbf{x}}^{K}(\mathbf{y}) = d_{\mathcal{L}}^{K}(\mathbf{x}, \mathbf{y})\frac{\mathbf{y} + \frac{1}{K}\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}\mathbf{x}}{||\mathbf{y} + \frac{1}{K}\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}\mathbf{x}||_{\mathcal{L}}}$$



Hyperbolic GNN (1)

- **Challenges**
  - Input node features are usually **Euclidean**
  - Perform hyperbolic aggregation for message passing
  - Choose hyperbolic spaces with the **right amount of curvature** at every layer of the GNN

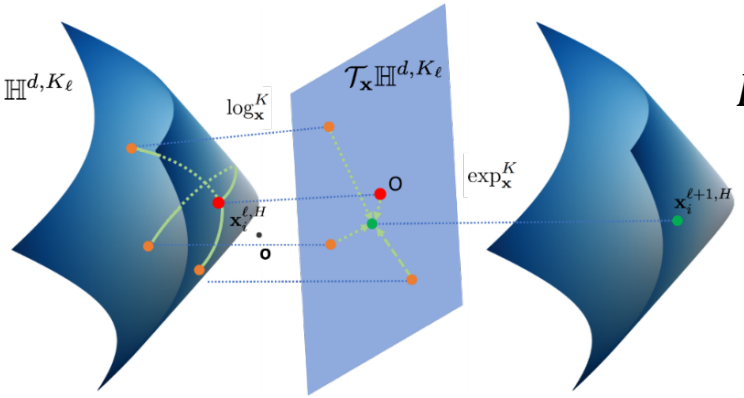


Hyperbolic Graph Convolutional Neural Networks, NeurIPS 2019

Hyperbolic GNN (2)

- $\boldsymbol{h}_i^{l,H} = \mathrm{Msg}\left(\boldsymbol{x}_i^{l-1,H}\right)$ **Message**
- $\boldsymbol{y}_i^{l,H} = \mathrm{AGG}^{K_{l-1}}\left(\boldsymbol{h}^{l,H}\right)_i$ **Aggregation**
- $\boldsymbol{x}_i^{l,H} = \mathrm{Update}^{K_{l-1},K_l}\left(\boldsymbol{y}_i^{l,H}\right)$ **Update**

At every layer:



Message **Aggregation** **Update**

K_l: curvature at layer l

Hyperbolic Graph Convolutional Neural Networks, NeurIPS 2019

Hyperbolic GNN (3)

- Input Transformation: $\boldsymbol{x}^{0,H} := \exp_o^K((0, \boldsymbol{x}^{0,E}))$
- **Message**: $\boldsymbol{h}_i^{l,H} = \left(W^l \otimes^{K_{l-1}} \boldsymbol{x}_i^{l-1,H}\right) \oplus^{K_{l-1}} \boldsymbol{b}^l$
  - Hyperbolic linear: $W \otimes^K \boldsymbol{x}^{l,H} := \exp_o^K(W \log_o^K(\boldsymbol{x}^{l,H}))$
  - Mobius addition: $\boldsymbol{x}^H \oplus^K \boldsymbol{b} := \exp_{x^H}^K(P_{o \to \boldsymbol{x}^H}^K(\boldsymbol{b}))$

Parallel Transport

$$P_{\mathbf{x}\to\mathbf{y}}(\mathbf{v}) = \mathbf{v} - \frac{\langle \log_{\mathbf{x}}(\mathbf{y}), \mathbf{v}\rangle_{\mathcal{L}}}{d_{\mathcal{L}}^1(\mathbf{x},\mathbf{y})^2}(\log_{\mathbf{x}}(\mathbf{y}) + \log_{\mathbf{y}}(\mathbf{x}))$$

- **Aggregation**: $AGG(x^H)_i := \exp_{x_i^H}^K\left(\sum_{j\in\mathcal{N}(i)} w_{ij} \log_{x_i^H}^K(\boldsymbol{x}_j^H)\right)$

Attention weights

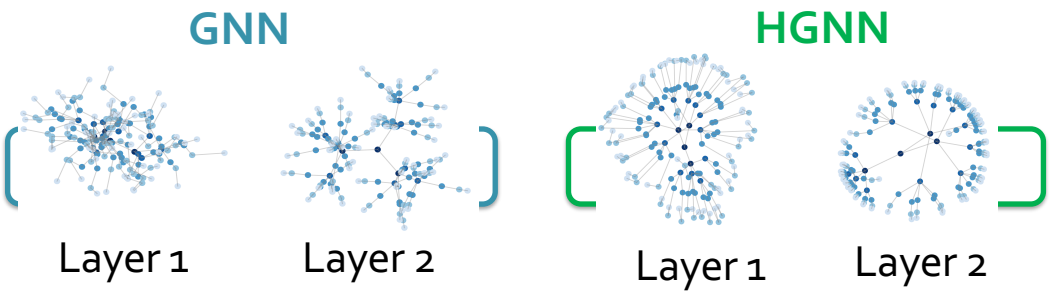
- **Update**: $\text{Update}^{K_{l-1},K_l}(\boldsymbol{x}^H) := \exp_o^{K_l}\left(\sigma\left(\log_o^{K_{l-1}}(\boldsymbol{x}^H)\right)\right)$

Trainable curvature

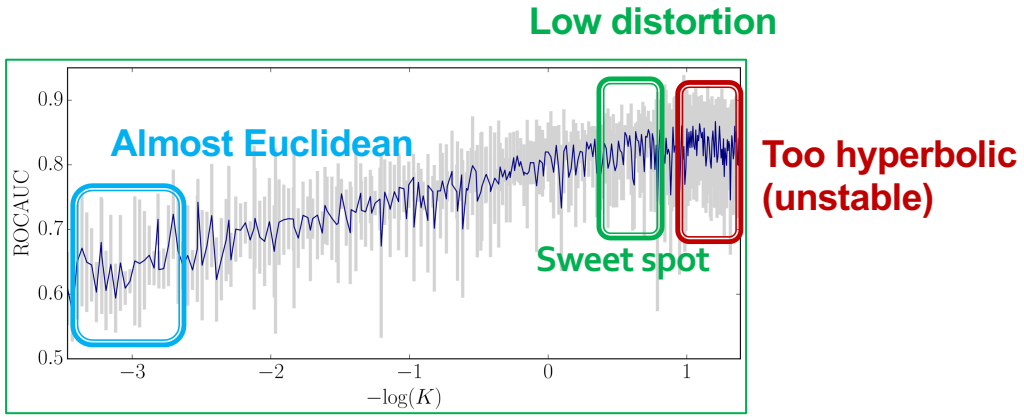
Hyperbolic GNN (4)

Node Classification (tree)

Disease spreding graph:
1044 nodes, 2 classes



- **Predict**: whether a person is infected
- **Groundtruth**: FIR disease spreading model
- **Curvature** is important!



Hyperbolic GNN Results

- Performance on a variety of datasets
- We use **hyperbolicity δ** to measure whether a graph dataset is tree-like and hierarchical
- See *Gromov et al. 1987* for its formula, and an implementation can be found here
- Lower value of hyperbolicity: graph is more tree-like

Inductive task

Non-hierarchical dataset

	Dataset / **Hyperbolicity δ**	DISEASE $\delta = 0$		DISEASE-M $\delta = 0$		AIRPORT $\delta = 1$		PUBMED $\delta = 3.5$		CORA $\delta = 11$	
	Method	LP	NC	LP	NC	LP	NC	LP	NC	LP	NC
Shallow	EUC	59.8 ± 2.0	32.5 ± 1.1	-	-	92.0 ± 0.0	60.9 ± 3.4	83.3 ± 0.1	48.2 ± 0.7	82.5 ± 0.3	23.8 ± 0.7
Shallow	HYP [28]	63.5 ± 0.6	45.5 ± 3.3	-	-	94.5 ± 0.0	70.2 ± 0.1	87.5 ± 0.1	68.5 ± 0.3	87.6 ± 0.2	22.0 ± 1.5
Shallow	EUC-MIXED	49.6 ± 1.1	35.2 ± 3.4	-	-	91.5 ± 0.1	68.3 ± 2.3	86.0 ± 1.3	63.0 ± 0.3	84.4 ± 0.2	46.1 ± 0.4
Shallow	HYP-MIXED	55.1 ± 1.3	56.9 ± 1.5	-	-	93.3 ± 0.0	69.6 ± 0.1	83.8 ± 0.3	73.9 ± 0.2	85.6 ± 0.5	45.9 ± 0.3
NN	MLP	72.6 ± 0.6	28.8 ± 2.5	55.3 ± 0.5	55.9 ± 0.3	89.8 ± 0.5	68.6 ± 0.6	84.1 ± 0.9	72.4 ± 0.2	83.1 ± 0.5	51.5 ± 1.0
NN	HNN[10]	75.1 ± 0.3	41.0 ± 1.8	60.9 ± 0.4	56.2 ± 0.3	90.8 ± 0.2	80.5 ± 0.5	94.9 ± 0.1	69.8 ± 0.4	89.0 ± 0.1	54.6 ± 0.4
GNN	GCN[20]	64.7 ±0.5	69.7 ± 0.4	66.0 ± 0.8	59.4 ± 3.4	89.3 ± 0.4	81.4 ± 0.6	91.1 ± 0.5	78.1 ± 0.2	90.4 ± 0.2	81.3 ± 0.3
GNN	GAT [39]	69.8 ±0.3	70.4 ± 0.4	69.5 ± 0.4	62.5 ± 0.7	90.5 ± 0.3	81.5 ± 0.3	91.2 ± 0.1	79.0 ± 0.3	**93.7** ± 0.1	**83.0** ± 0.7
GNN	SAGE [14]	65.9 ± 0.3	69.1 ± 0.6	67.4 ± 0.5	61.3 ± 0.4	90.4 ± 0.5	82.1 ± 0.5	86.2 ± 1.0	77.4 ± 2.2	85.5 ± 0.6	77.9 ± 2.4
GNN	SGC [42]	65.1 ± 0.2	69.5 ± 0.2	66.2 ± 0.2	60.5 ± 0.3	89.8 ± 0.3	80.6 ± 0.1	94.1 ± 0.0	78.9 ± 0.0	91.5 ± 0.1	81.0 ± 0.1
Ours	HGCN	**90.8** ± 0.3	**74.5** ± 0.9	**78.1** ± 0.4	**72.2** ± 0.5	**96.4** ± 0.1	**90.6** ± 0.2	**96.3** ± 0.0	**80.3** ± 0.3	92.9 ± 0.1	79.9 ± 0.2
Ours	(%) ERR RED	-63.1%	-13.8%	-28.2%	-25.9%	-60.9%	-47.5%	-27.5%	-6.2%	+12.7%	+18.2%

Hyperbolic GNN: Summary

- We introduce **hyperbolic geometry** as the embedding space to embed hierarchical, tree-like graphs
- 2 models for representing hyperbolic space: **Poincaré ball** model and **hyperboloid** model
- Use **exponential and logarithmic map** to map points to and from tangent spaces where neural network operations can be performed
- **Learnable curvature** allows the model to learn a tradeoff between performance and stability

Design Space of Graph Neural Networks

CS224W: Machine Learning with Graphs

Jure Leskovec, Stanford University

http://cs224w.stanford.edu



Key Questions for GNN Design

- **GNN architectural design:**
  - How to find a good GNN design for a specific GNN task?
- **Important but challenging:**
  - GNNs have been very successful in various applications
  - Domain experts want to use SOTA GNN on their specific tasks, however...
    - There are tons of possible GNN models
    - Best design in one task can perform badly for another task
    - Redo grid search for each new task is NOT feasible
- **Our key contribution:**
  - The first systematic study for the ***GNN design space and task space***
  - **GraphGym**, a powerful platform for exploring different GNN designs and tasks

Background: Terminology

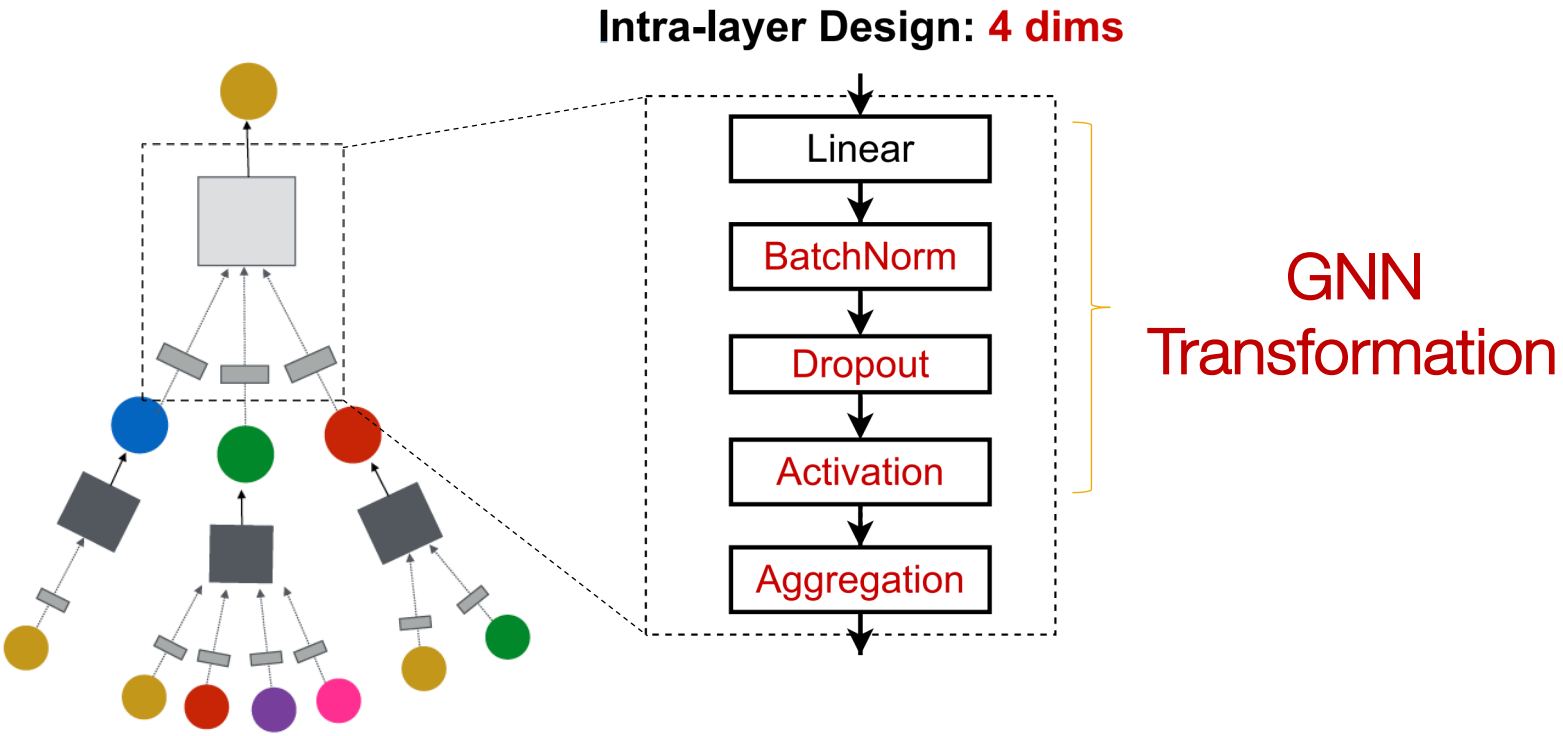
- **Design:** a concrete model instantiation
  - E.g., a 4-layer GraphSAGE
- **Design dimensions** characterize a design
  - E.g., the number of layers $L \in \{2, 4, 6, 8\}$
- **Design choice** is the actual selected value in the design dimension
  - E.g., $L = 2$
- **Design space** consists of a Cartesian product of design dimensions
- **Task:** A specific task of interest
  - E.g., node classification on Cora, graph classification on ENZYMES
- **Task space** consists of multiple tasks

Recap: GNN Design Space

Intra-layer Design:

GNN Layer = Transformation + Aggregation

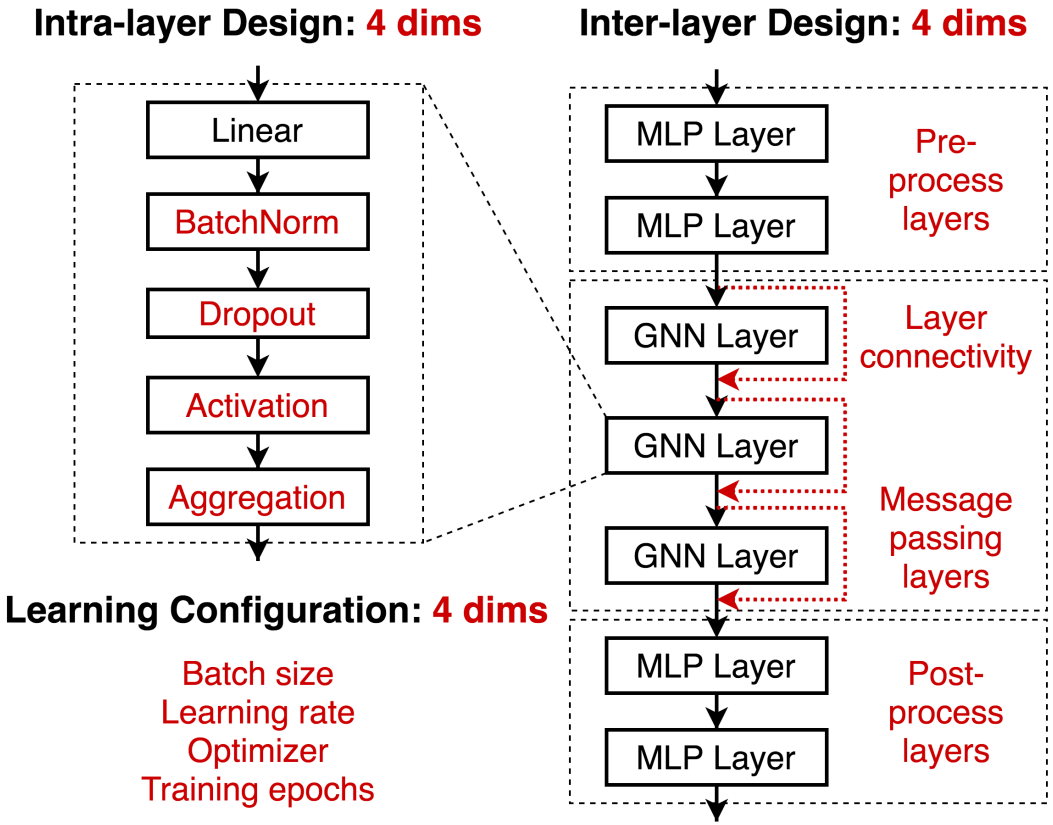
- **We propose a general instantiation under this perspective**



Recap: GNN Design Space

Inter-layer Design

- **We explore different ways of organizing GNN layers**

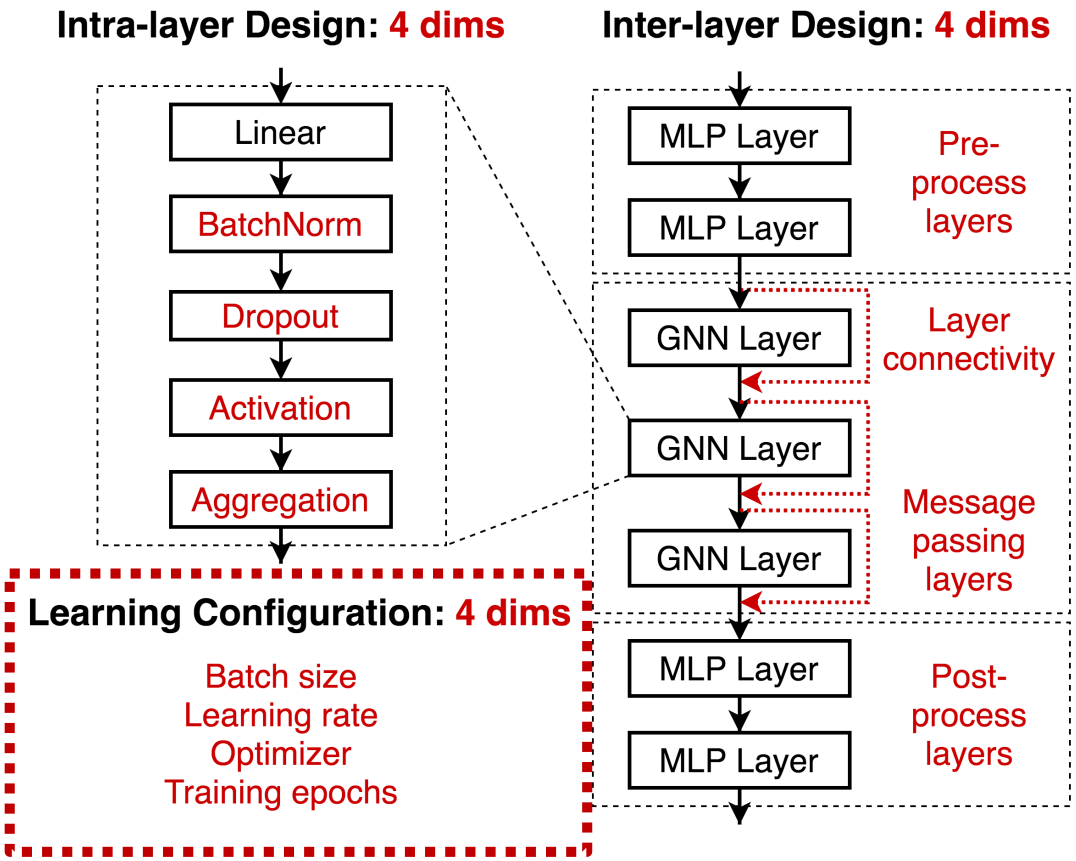


Pre-process layers:
Important when expressive node feature encoder is needed
E.g., when nodes are images/text

Skip connections:
Improve deep GNN's performance

Post-process layers:
Important when reasoning or transformation over node embeddings are needed
E.g., graph classification, knowledge graphs

Recap: GNN Design Space



Learning configurations

- **Often neglected in current literature**
- **But we found they have high impact on performance**

Summary: GNN Design Space

- **Overall: A GNN design space**
  - Intra-layer design

Batch Normalization	Dropout	Activation	Aggregation
True, False	False, 0.3, 0.6	RELU, PRELU, SWISH	MEAN, MAX, SUM

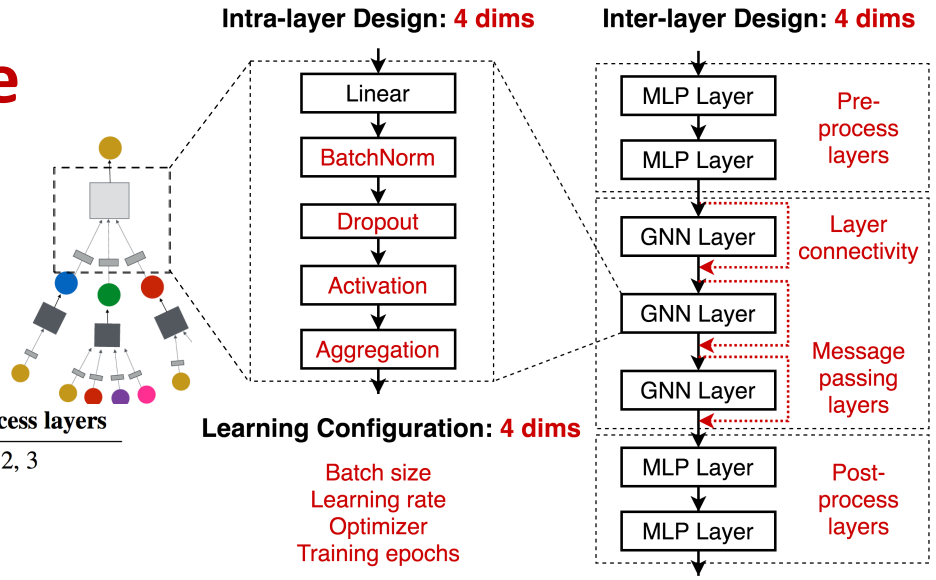
  - Inter-layer design

Layer connectivity	Pre-process layers	Message passing layers	Post-precess layers
STACK, SKIP-SUM, SKIP-CAT	1, 2, 3	2, 4, 6, 8	1, 2, 3

  - Learning configuration

Batch size	Learning rate	Optimizer	Training epochs
16, 32, 64	0.1, 0.01, 0.001	SGD, ADAM	100, 200, 400

  - **In total:** 315K possible designs



- **Our Purpose:**
  - We don't want to (and we cannot) cover all the possible designs
  - We want to demonstrate that **studying a design space is more effective than studying individual GNN designs**

A General GNN Task Space

- **Categorizing GNN tasks**
  - **Common practice**: node / edge / graph level task
  - Reasonable but not precise
    - **Node prediction:** predict clustering coefficient vs. predict a node's subject area – completely different task
  - But creating a precise taxonomy of GNN tasks is very hard!
    - Novel GNN applications can always emerge
- **Our innovation: a quantitative task similarity metric**
  - **Purpose: understand GNN tasks, transfer the best GNN models across tasks**

A General GNN Task Space

- **Innovation:** quantitative **task similarity metric**
  - **1)** Select "anchor" models ($M_1, \ldots, M_5$)
  - **2)** Characterize a task by ranking the performance of anchor models
  - **3)** Tasks with similar rankings are considered as similar

Task Similarity Metric

	Anchor Model Performance ranking					Similarity to Task A
Task A	M_1	M_2	M_3	M_4	M_5	1.0
Task B	M_1	M_3	M_2	M_4	M_5	0.8
Task C	M_5	M_1	M_4	M_3	M_2	-0.4

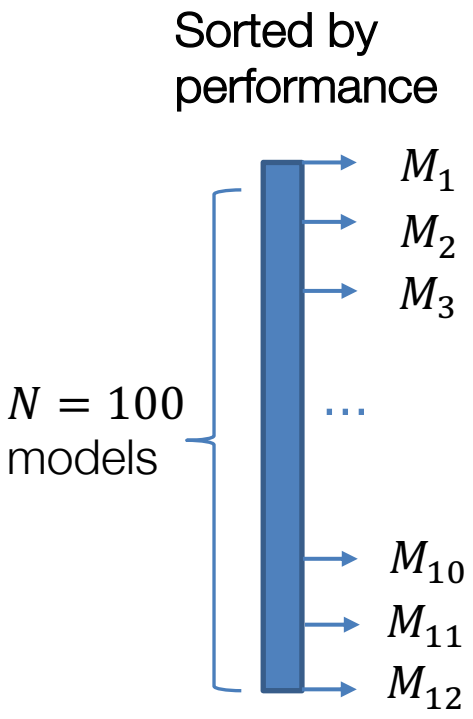
Task A is similar to Task B

Task A is not similar to Task C

- **How do we select the anchor models?**

A General GNN Task Space

- **Selecting the anchor models**
  - 1) Select a small dataset
    - E.g., node classification on Cora
  - 2) Randomly **sample N models from our design space**, run on the dataset
    - E.g., we sample 100 models
  - 3) Sort these models based on their performance, evenly select M models as the anchor models, whose **performance range from the worst to the best**
    - E.g., we sample 12 models in our experiments
  - **Intuition:** a bad model in one task could be great for another task



A General GNN Task Space

- **We collect 32 tasks**: node / graph classification

Task name
node-AmazonComputers-N/A-N/A
node-AmazonPhoto-N/A-N/A
node-CiteSeer-N/A-N/A
node-CoauthorCS-N/A-N/A
node-CoauthorPhysics-N/A-N/A
node-Cora-N/A-N/A
node-scalefree-clustering-pagerank
node-scalefree-const-clustering
node-scalefree-const-pagerank
node-scalefree-onehot-clustering
node-scalefree-onehot-pagerank
node-scalefree-pagerank-clustering
node-smallworld-clustering-pagerank
node-smallworld-const-clustering
node-smallworld-const-pagerank
node-smallworld-onehot-clustering
node-smallworld-onehot-pagerank
node-smallworld-pagerank-clustering
graph-PROTEINS-N/A-N/A
graph-BZR-N/A-N/A
graph-COX2-N/A-N/A
graph-DD-N/A-N/A
graph-ENZYMES-N/A-N/A
graph-IMDB-N/A-N/A
graph-scalefree-clustering-path
graph-scalefree-const-path
graph-scalefree-onehot-path
graph-scalefree-pagerank-path
graph-smallworld-clustering-path
graph-smallworld-const-path
graph-smallworld-onehot-path
graph-smallworld-pagerank-path
graph-ogbg-molhiv-N/A-N/A

(We include link prediction results in the Appendix)

6 Real-world node classification tasks

12 Synthetic node classification tasks
Predict node properties:
- Clustering coefficient
- PageRank

6 Real-world graph classification tasks

8 Synthetic node classification tasks
Predict graph properties:
- Average path length

Evaluating GNN Designs

- **Evaluating a design dimension**:
  - "Is BatchNorm generally useful for GNNs?"
- **The common practice:**
  - **(1) Pick one model** (e.g., a 5-layer 64-dim GCN)
  - **(2) Compare two models**, with BN = True / False
- **Our approach:**
  - Note that **we have defined** $315\text{K}\ (\text{models}) * 32\ (\text{tasks}) \approx$ **10M model-task combinations**
  - **(1) Sample from 10M possible model-task combinations**
  - **(2) Rank the models** with BN = True / False
- How do we make it **scalable & convincing?**

Evaluating GNN Designs

- **Evaluating a design dimension**: controlled random search
  - **a) Sample random model-task configurations**, perturb `BatchNorm = [True, False]`
  - **Here we control the computational budget for all the models**

(a) Controlled Random Search

GNN Design Space					GNN Task Space	
BatchNorm	Activation	…	Message layers	Layer Connectivity	Task level	dataset
True	relu	…	8	skip_sum	node	CiteSeer
False	relu	…	8	skip_sum	node	CiteSeer
True	relu	…	2	skip_cat	graph	BZR
False	relu	…	2	skip_cat	graph	BZR
…						
True	prelu	…	4	stack	graph	scale free
False	prelu	…	4	stack	graph	scale free

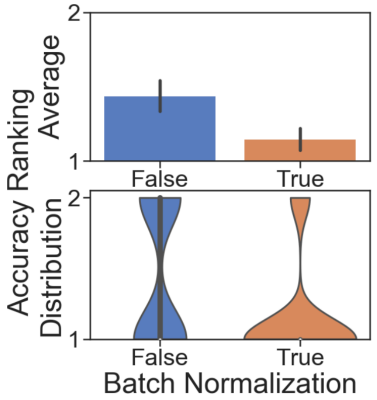
Evaluating GNN Designs

- **b) Rank** `BatchNorm = [True, False]` by their performance （lower ranking is better）
- **c) Plot Average / Distribution of the ranking** of `BatchNorm = [True, False]`

(b) Rank Design Choices by Performance

GNN Design Space	Experimental Results	
BatchNorm	Val. Accuracy	Design Choice Ranking
True	0.75	1
False	0.54	2
True	0.88	1 (a tie)
False	0.88	1 (a tie)
True	0.89	1
False	0.36	2

(c) Ranking Analysis



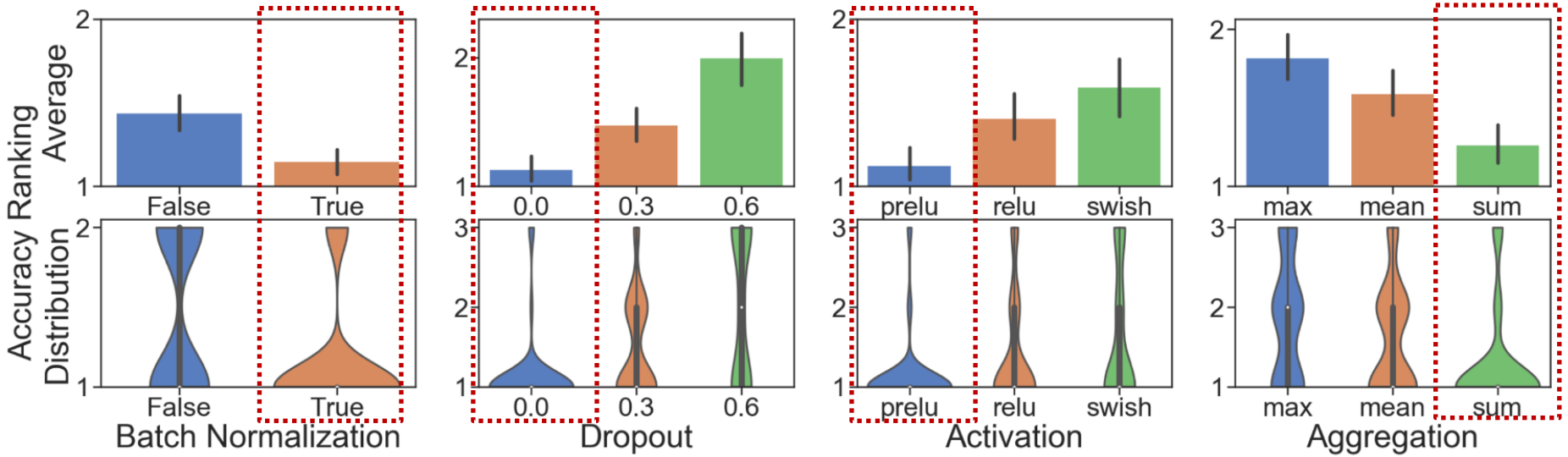
- **Summary: Convincingly evaluate any new design dimension**, e.g., a new GNN layer

Results 1: A Guideline for GNN Design

- Certain design choices exhibit clear advantages

Explanation:
GNNs are hard to optimize

Explanation:
This is our new finding!



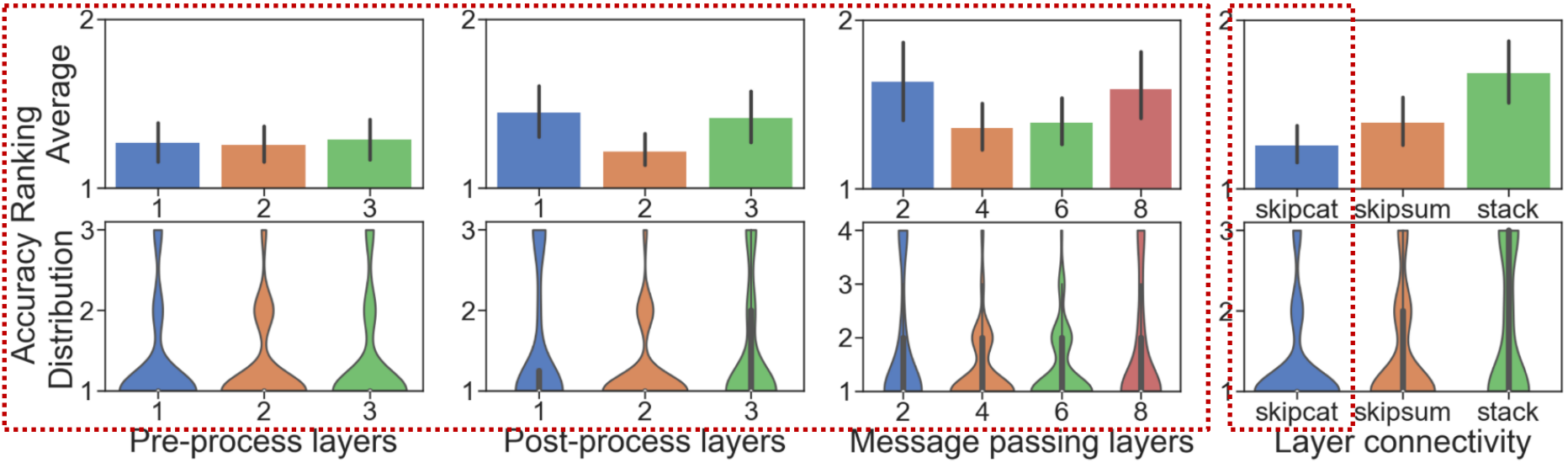
Explanation:
GNNs experience underfitting more often

Explanation:
Sum is the most expressive aggregator

Results 1: A Guideline for GNN Design

- Certain design choices exhibit clear advantages

Optimal number of layers is hard to decide
Highly dependent on the task



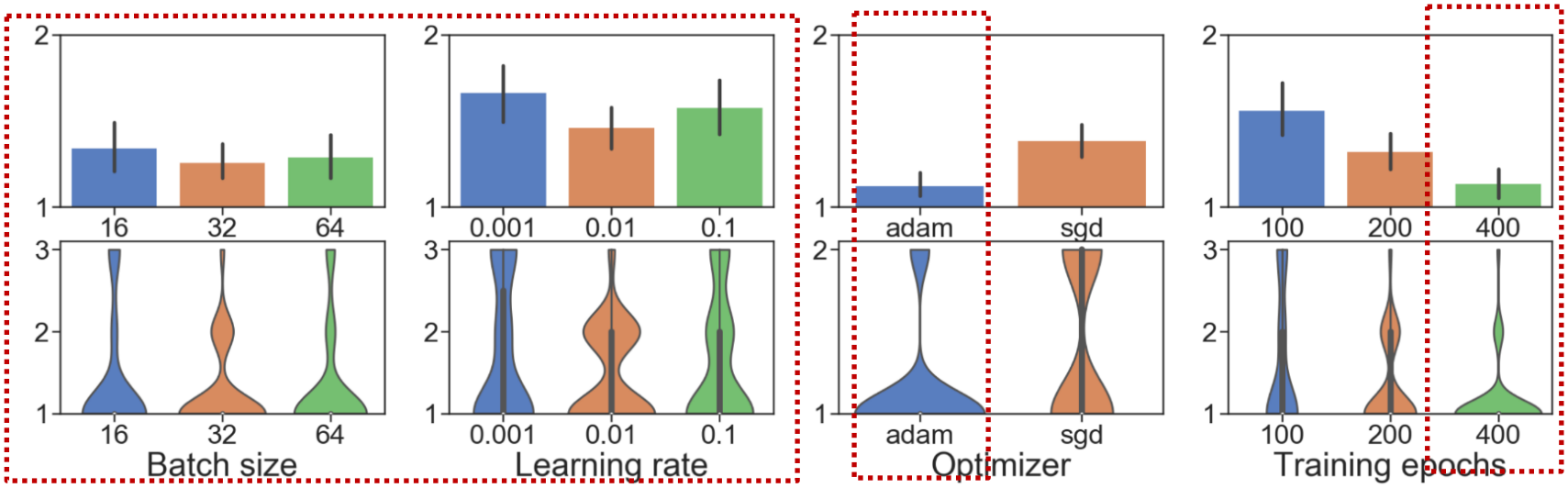
Explanation:
Skip connection enable hierarchical node representation

Results 1: A Guideline for GNN Design

- Certain design choices exhibit clear advantages

Optimal batch size and learning rate is hard to decide

Highly dependent on the task



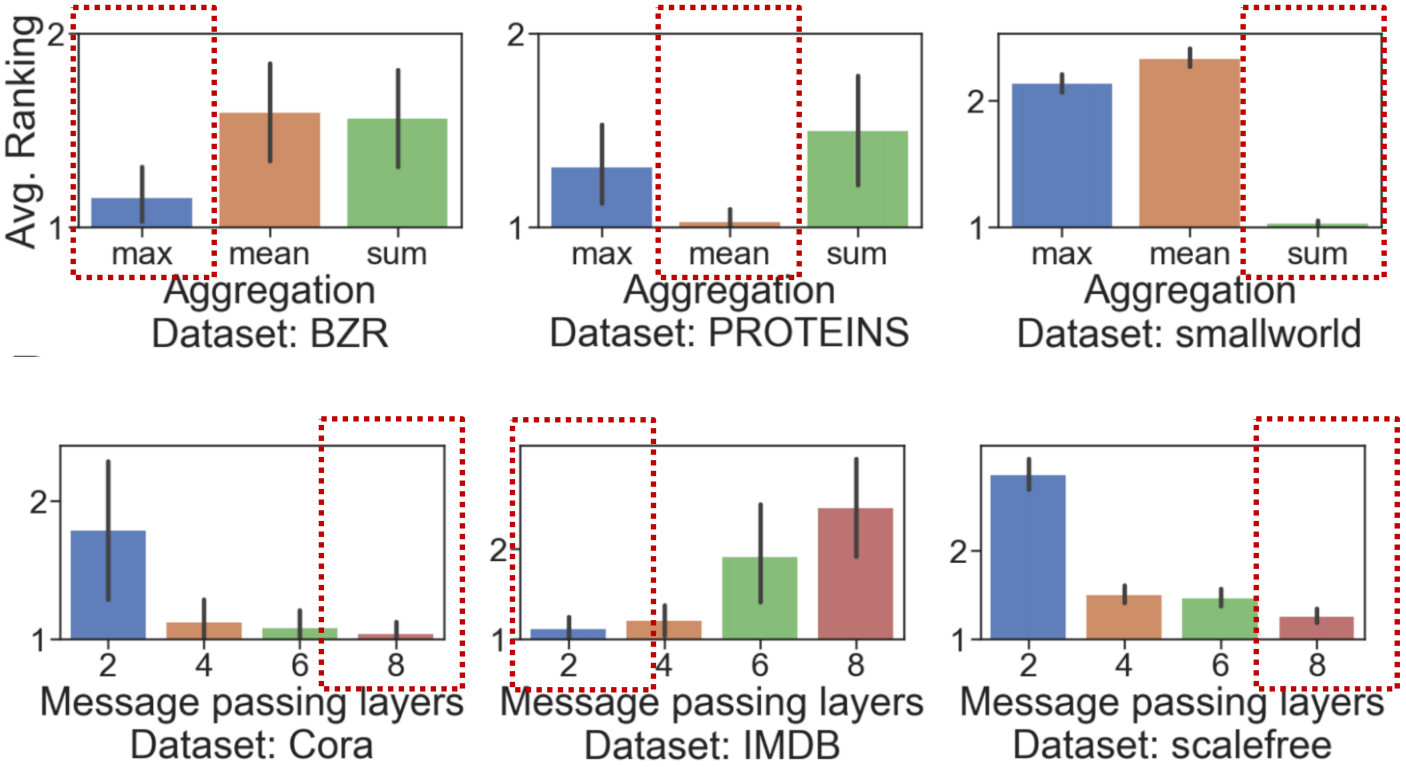
Explanation:

Adam is more robust

More training epochs is better

Results 2: Understanding GNN Tasks

- **Best GNN designs in different tasks vary significantly**
  - Motivate that studying a task space is crucial

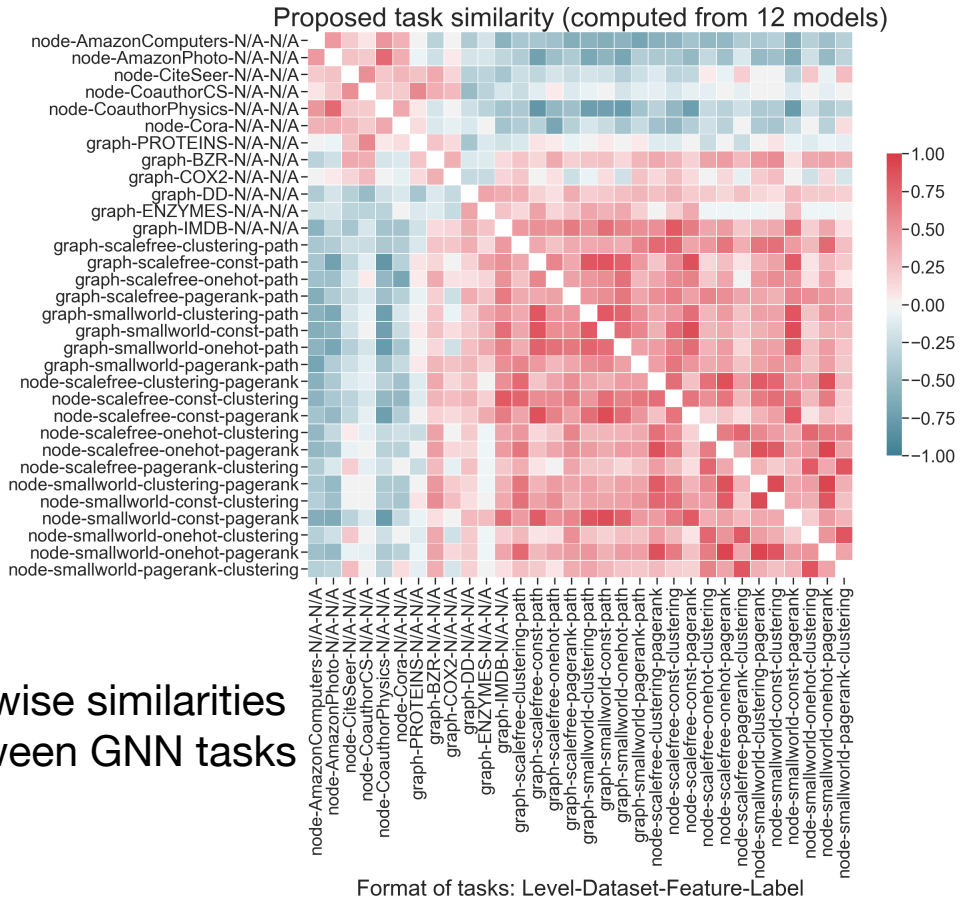


Results 2: Understanding GNN Tasks

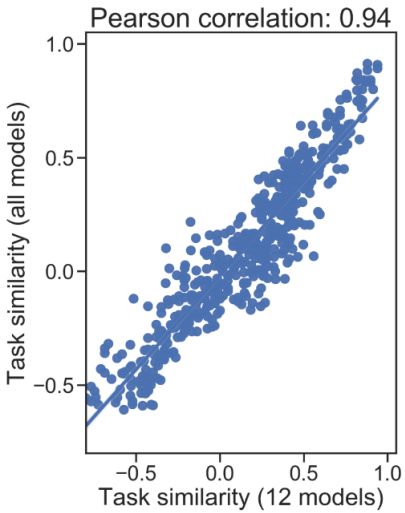
- **Build a GNN task space**

Recall how we **compute task similarity**

	Anchor Model Performance ranking					Similarity to Task A
Task A	M_1	M_2	M_3	M_4	M_5	1.0
Task B	M_1	M_3	M_2	M_4	M_5	0.8
Task C	M_5	M_1	M_4	M_3	M_2	-0.4



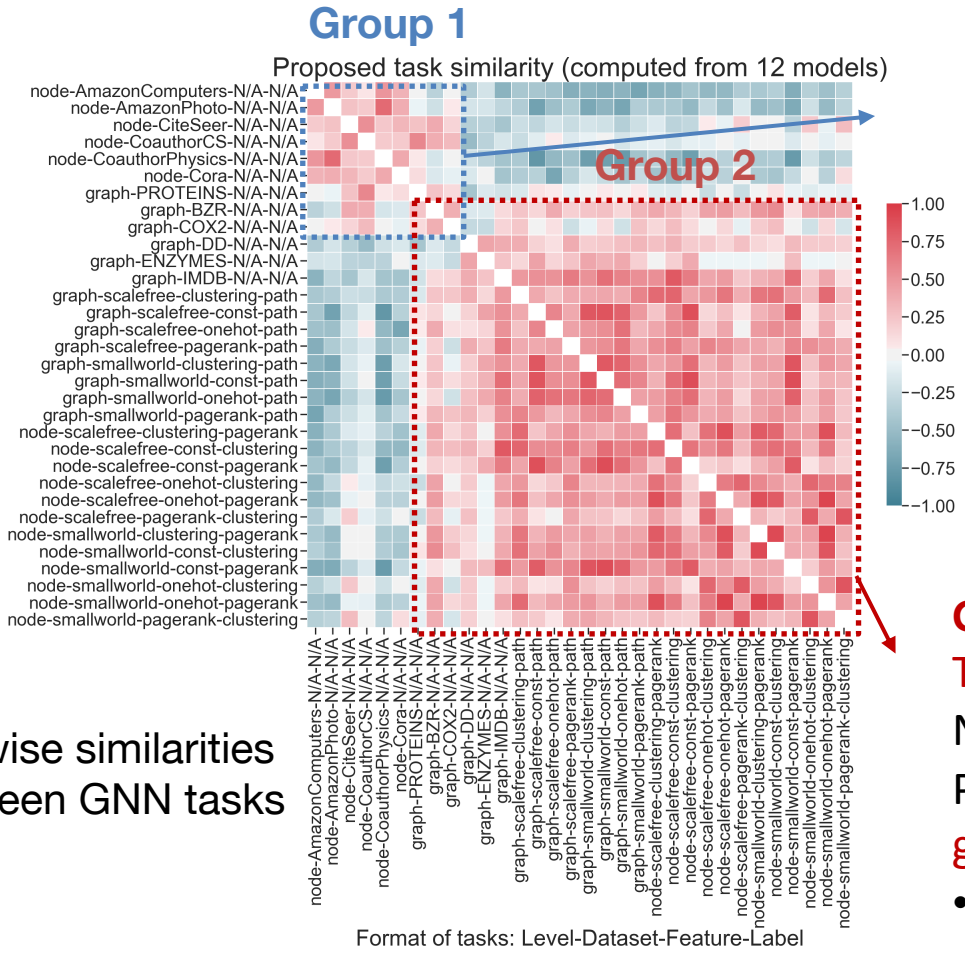
Pairwise similarities between GNN tasks



Task similarity computation is cheap: Using **12 anchor models** is a good approximation!

Results 2: Understanding GNN Tasks

- **Proposed GNN task space is informative**



Pairwise similarities between GNN tasks

Group 1:

Tasks rely on **feature information**

Node/graph classification tasks, where input graphs have high-dimensional features

- Cora graph has 1000+ dim node feature

Group 2:

Tasks rely on **structural information**

Nodes have few features

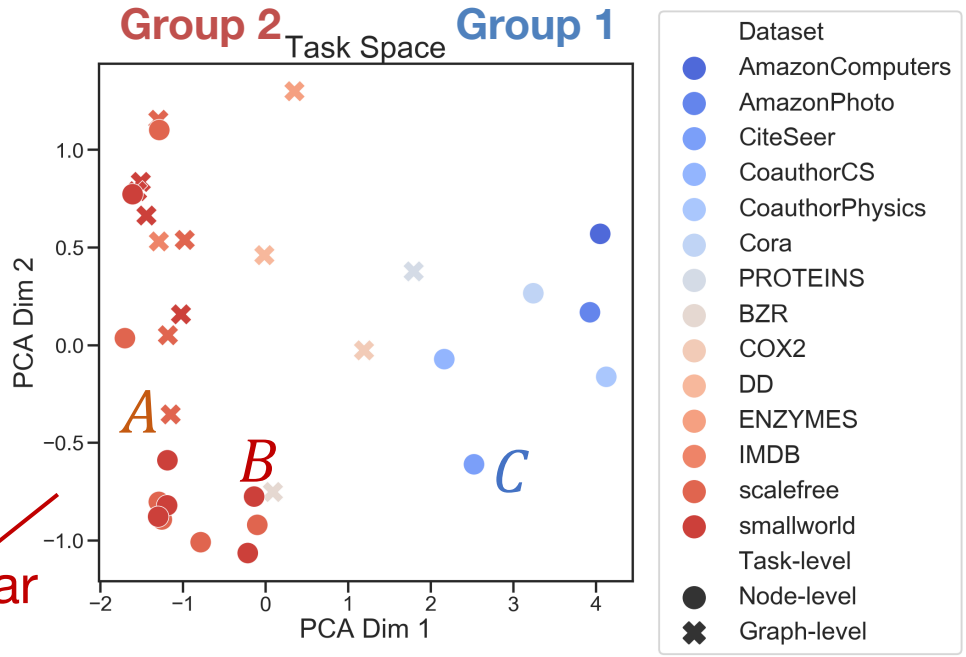
Predictions are highly dependent on graph structure

- Predicting clustering coefficients

Results 2: Understanding GNN Tasks

- **Proposed GNN task space is informative**

PCA

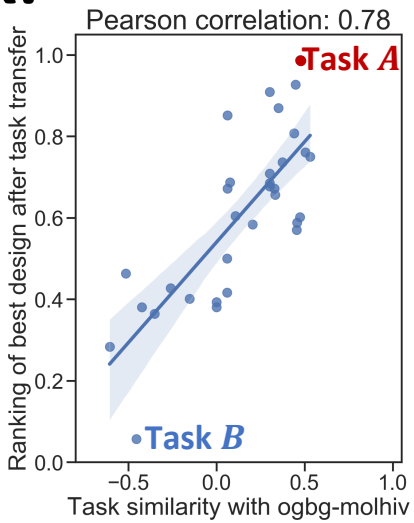


Similar tasks have similar best architectures

Best GNN Designs Found in Different Tasks					
	Pre layers	MP layers	Post layers	Connectivity	AGG
Task A	2	8	2	skip-sum	sum
Task B	1	8	2	skip-sum	sum
Task C	2	6	2	skip-cat	mean

Results 3: Transfer to Novel Tasks

- **Case study**: generalize best models to **unseen** OGB ogbg-molhiv task
  - **ogbg-molhiv is unique from other tasks**: 20x larger, imbalanced (1.4% positive) and requires out-of-distribution generalization
- **Concrete steps for applying to a novel task:**
  - **Step 1:** Run 12 anchor models on the new task
  - **Step 2:** Compute similarity between the new task and existing tasks
  - **Step 3:** Recommend the best designs from existing tasks with high similarity



Results 3: Transfer to Novel Tasks

- **Our task space can guide best model transfer to novel tasks!**

We pick 2 tasks:

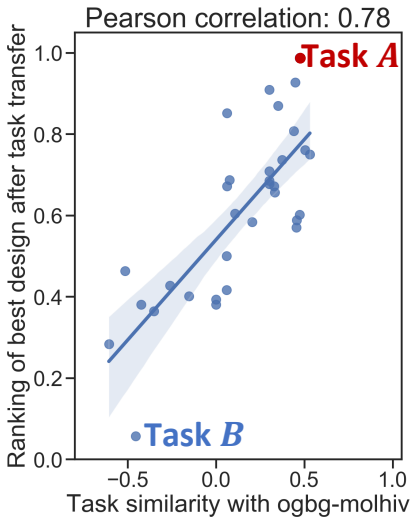
Task A: Similar to OGB

Task B: Not similar to OGB

Findings:

Transfer the best model from Task A achieves SOTA on OGB

Transfer the best model from Task B performs badly on OGB



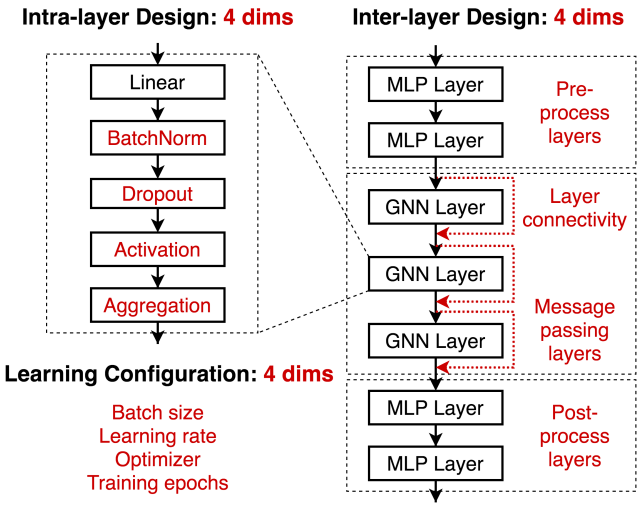
	Task A: graph-scalefree-const-path	Task B: node-CoauthorPhysics
Best design in our design space	(1, 8, 3, skipcat, sum)	(1, 4, 2, skipcat, max)
Task Similarity with ogbg-molhiv	**0.47**	**-0.61**
Performance after transfer to ogbg-molhiv	***0.785***	***0.736***

Previous SOTA: 0.771

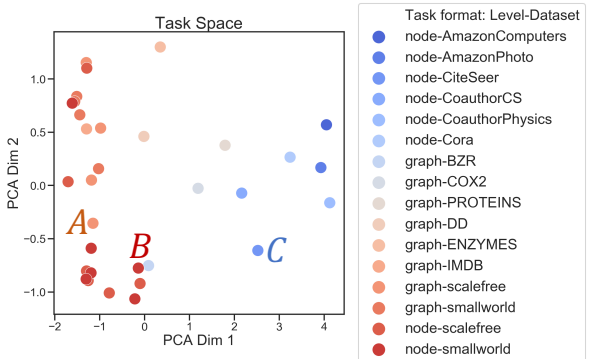
GNN Design Space: Summary

- The first systematic investigation of:
  - **General guidelines for GNN design**
  - **Understandings of GNN tasks**
  - **Transferring best GNN designs across tasks**
  - **GraphGym:** Easy-to-use **code platform for GNN**

(a) GNN Design Space



(b) GNN Task Space



Best GNN Designs Found in Different Tasks					
	Pre layers	MP layers	Post layers	Connectivity	AGG
Task *A*	2	8	2	skip-sum	sum
Task *B*	1	8	2	skip-sum	sum
Task *C*	2	6	2	skip-cat	mean

(c) Controlled Random Search

GNN Design Space					GNN Task Space	
BatchNorm	Act	…	MP layers	Connectivity	level	dataset
True	relu	…	8	skip_sum	node	CiteSeer
False	relu	…	8	skip_sum	node	CiteSeer
True	relu	…	2	skip_cat	graph	BZR
False	relu	…	2	skip_cat	graph	BZR
…						

(d) Rank Design Choices by Performance

Experimental Results	
Val. Accuracy	Design Choice Ranking
0.75	1
0.54	2
0.88	1 (a tie)
0.86	1 (a tie)
…	

(e) Ranking Analysis

