Yale

GNN AutoML

CPSC483: Deep Learning on Graph-Structured Data

Rex Ying

Readings

- Readings are updated on the website (syllabus page)
- **Lecture 15 Readings:**
  - Trustworthy Graph Neural Networks
  - GraphFramEx Evaluation
- **Lecture 16 readings:**
  - Design Space
  - Auto-GNN

Introduction to AutoML

- Automatic machine learning (**AutoML**)
  - Find the best model **architecture** and **hyperparameter** setting for a given task
  - AutoML can be performed with heuristic search or machine learning
- **Overall pipeline**
  - Define the machine learning problem
    - Task, loss function, dataset, …
  - Specify the model **design space**
    - All the possible architecture and hyperparameter setting
  - Apply a search method to **navigate** the design space to find the top-performing model design

Outline of Today's Lecture

1. Design Space of GNN

2. AutoML based on Meta-model

Outline of Today's Lecture

1. Design Space of GNN

2. AutoML based on Meta-model

Key Questions for GNN Design

- **Goal**
  - How to find a good GNN design for a specific GNN task?
- **Important but challenging**
  - Domain experts want to use SOTA GNN on their specific tasks, however...
    - There are tons of possible GNN architectures
      - GCN, GraphSAGE, GAT, GIN, MAGNA, spectral GNNs …
    - **Issue:** Best design in one task can perform badly for another task
    - Search for each new task is very expensive
- **GraphGym**
  - The first systematic study for the ***GNN design space and task space***
  - A powerful platform for exploring different GNN designs and tasks

Background: Terminology

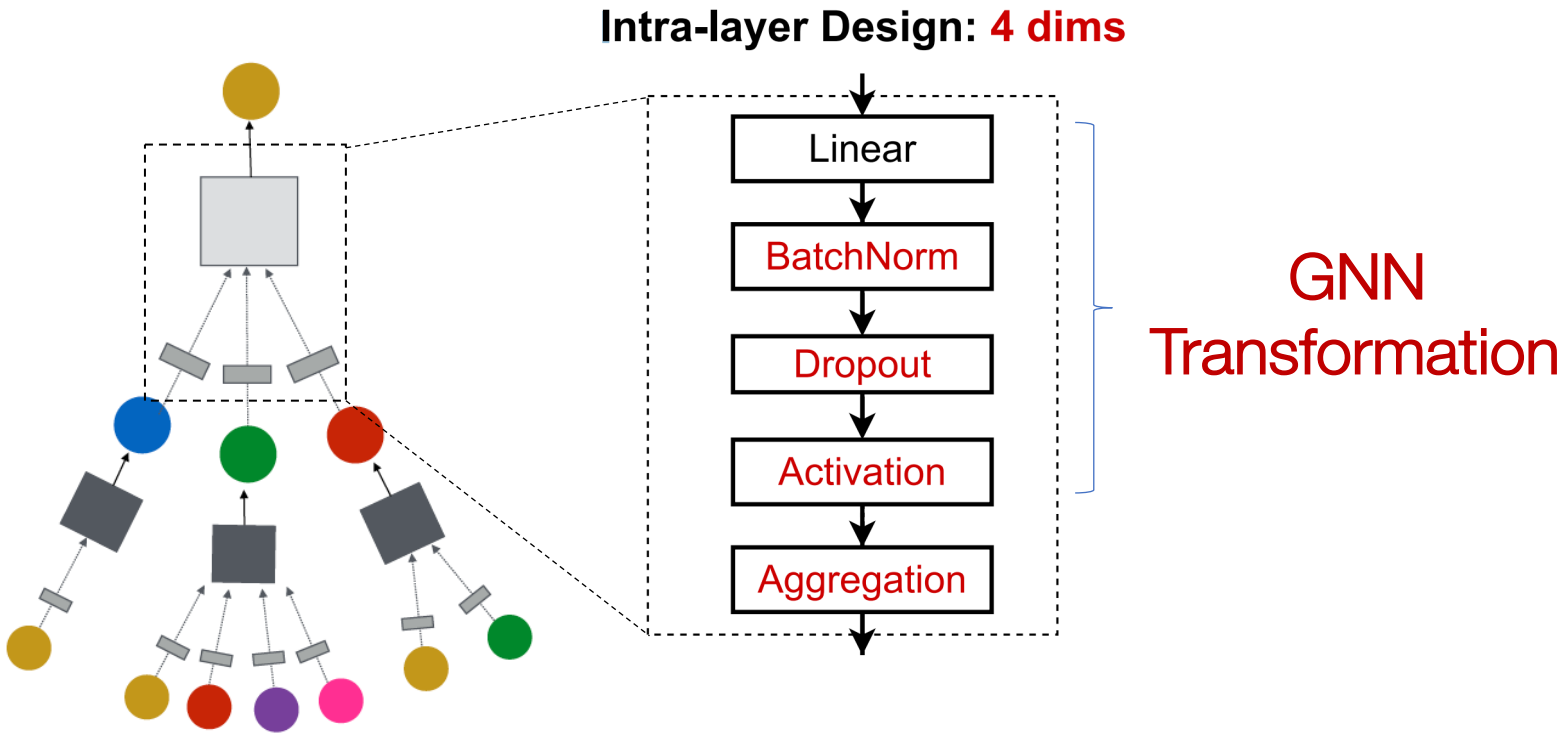
- **Design**: a concrete model instantiation
  - E.g., a 4-layer GraphSAGE with mean aggregation, skip layer, and batch normalization
- **Design dimensions** characterize a design
  - E.g., the number of layers $L \in \{2, 4, 6, 8\}$
- **Design choice** is the actual selected value in the design dimension
  - E.g., the number of layers $L = 2$
- **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 all the tasks we care about

Recap: GNN Design Space

Intra-layer Design:

GNN Layer = Transformation + Aggregation

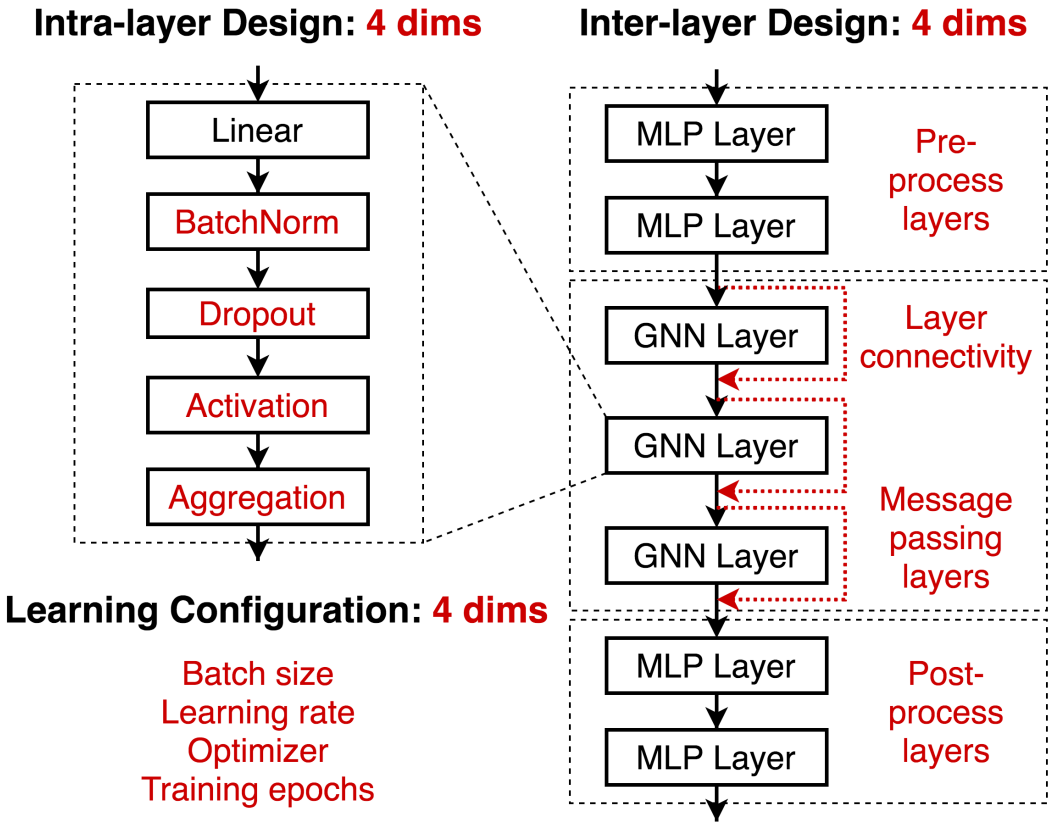
- **We propose a general instantiation under this perspective**



Recap: GNN Design Space

Inter-layer Design

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

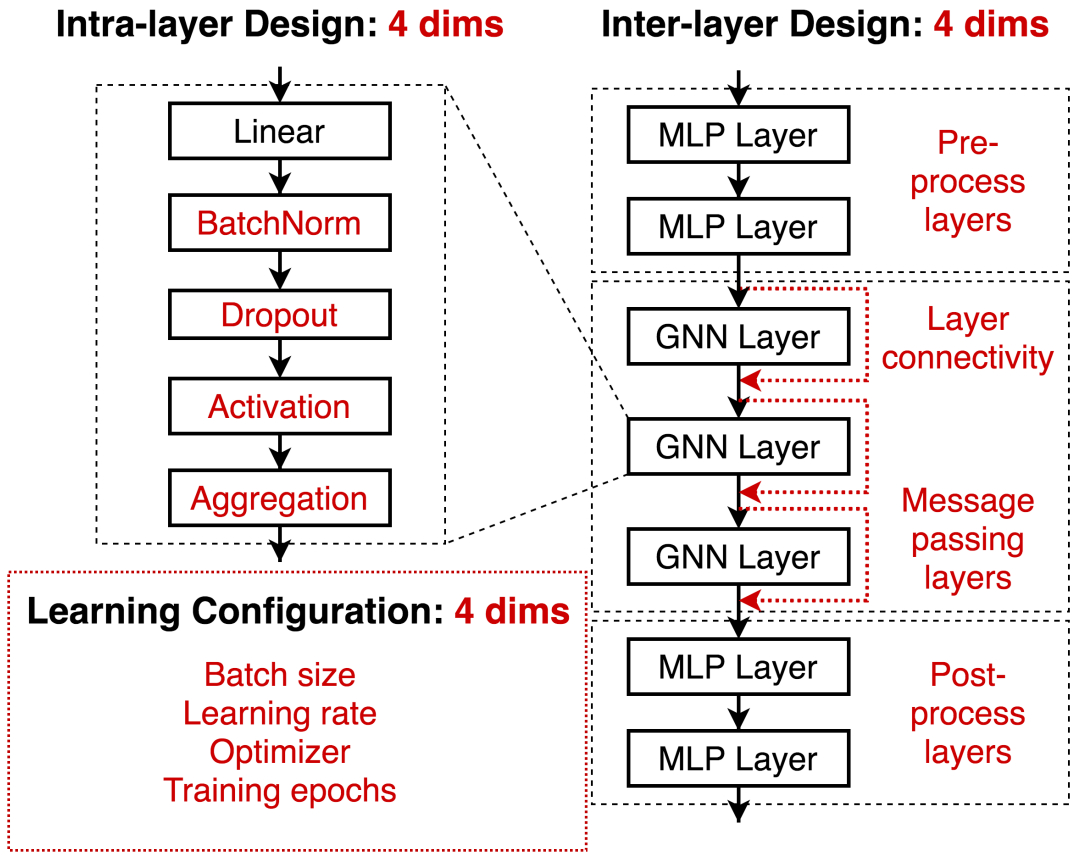


Pre-process layers:
Important when expressive node feature encoder is needed
E.g., when nodes are images/text

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

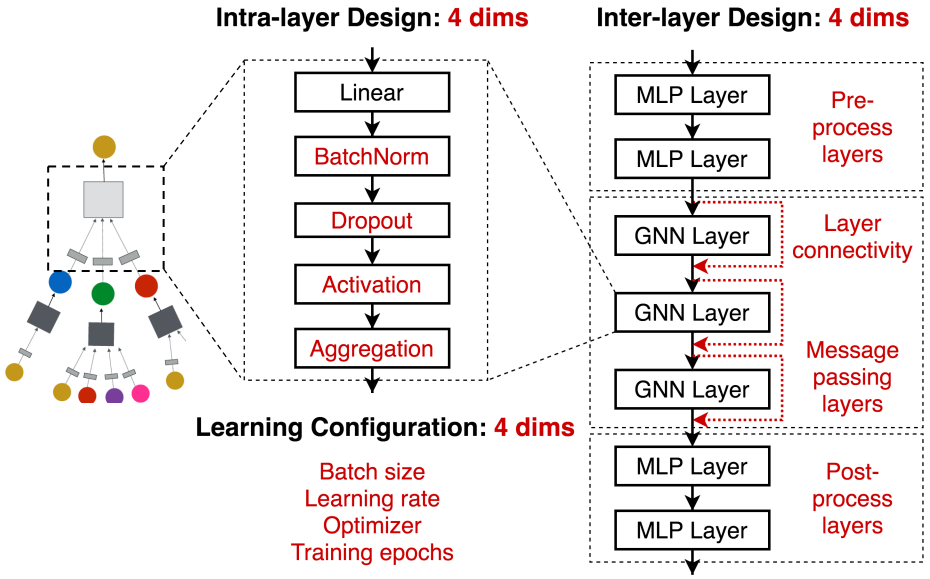
  - Inter-layer design

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
  - **A mindset transition:** We want to demonstrate that **studying a design space is more effective than studying individual GNN designs**

A General GNN Task Space (1)

- **Categorizing GNN tasks**
  - **Common practice**: categorize into node / edge / graph-level tasks
  - Reasonable but not precise
    - **Node prediction:** predict clustering coefficient vs. predict a node's subject area in a citation networks – **completely different task**
  - But creating a precise taxonomy of GNN tasks is very hard!
    - **Subjective**; **Novel GNN tasks** can always emerge
- **We instead propose a quantitative task similarity metric**
  - **Understand GNN tasks** based on performance
  - **Transfer** the best GNN models across tasks

A General GNN Task Space (2)

- **Innovation:** quantitative **task similarity metric**
  - **1)** Select **"anchor" models** $(M_1, \dots, M_5)$
  - **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

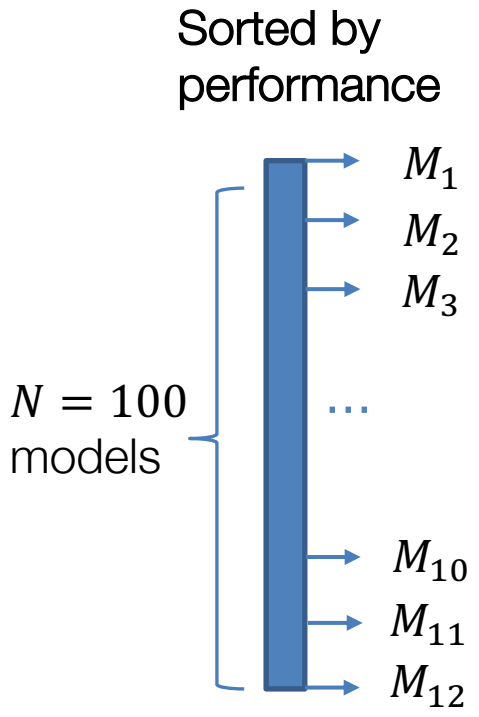
Task A is similar to Task B

Task A is not similar to Task C

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

A General GNN Task Space (3)

- **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
  - **Goal: Cover a wide spectrum of models:** A bad model in one task could be great for another task



A General GNN Task Space (4)

- **GraphGym collects 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

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 graph classification tasks
Predict graph properties:
- Average path length

Insight: Similar tasks share similar best architecture / hyper-parameter settings

Evaluating GNN Designs (1)

- **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
- **Approach:**
  - Note that **we have defined** $315\text{K}\ (\text{models}) * 32\ (\text{tasks}) \approx \mathbf{10M}$ **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 (2)

- **Evaluating a design dimension**: controlled random search
  - **a) Sample random model-task configurations**, perturb: `BatchNorm = [True, False]`
  - 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

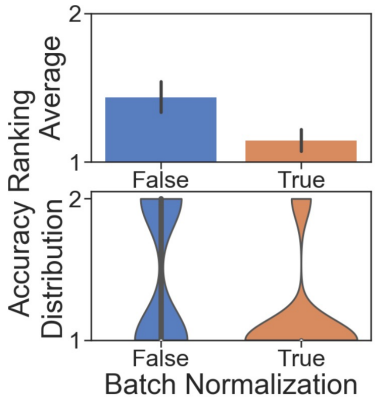
Evaluating GNN Designs (3)

- **Evaluating a design dimension**: controlled random search
  - **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



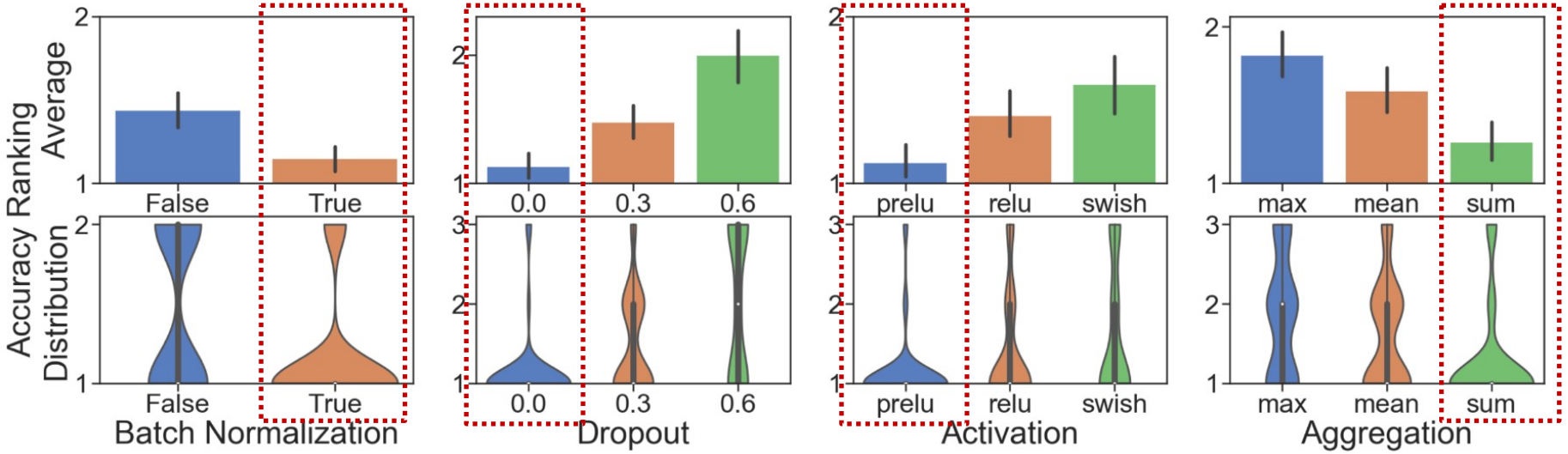
- **Goal: Comprehensively evaluate any new design dimension**
  e.g., evaluate a newly proposed GNN layer

Results 1: A Guideline for GNN Design (1)

- **Certain design choices exhibit clear advantages**
  - **Intra-layer designs:**

Explanation:
GNNs often
experience underfitting

Explanation:
Sum is generally an
effective aggregator



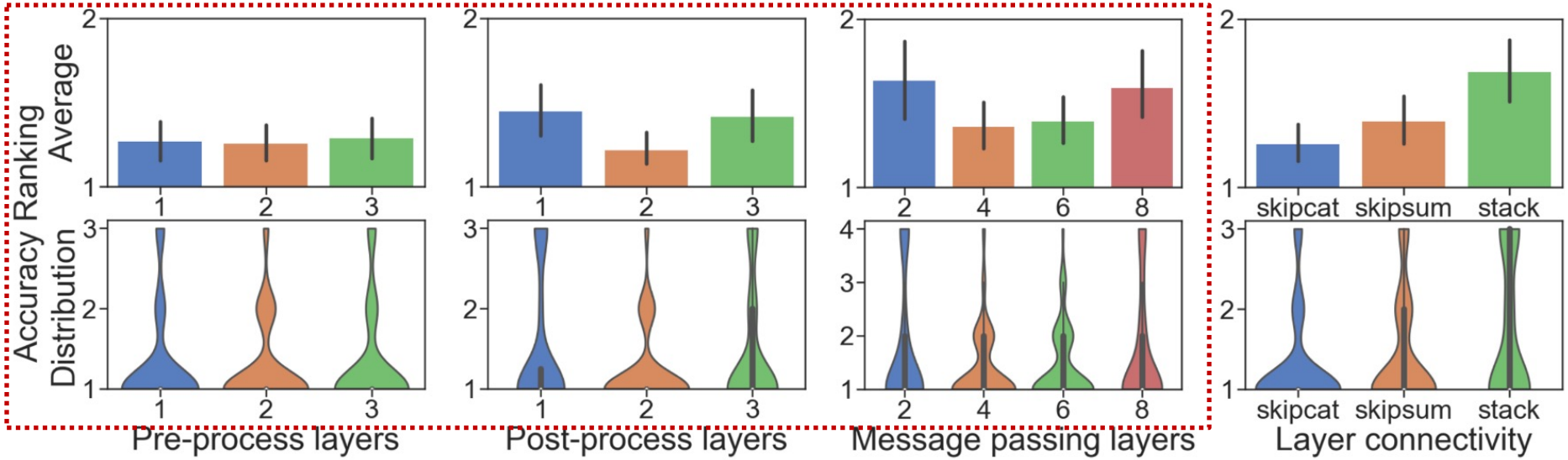
Explanation:
GNNs are hard to optimize

Explanation:
Prelu is effective!

Results 1: A Guideline for GNN Design (2)

- **Certain design choices exhibit clear advantages**
  - **Intra-layer designs:**

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



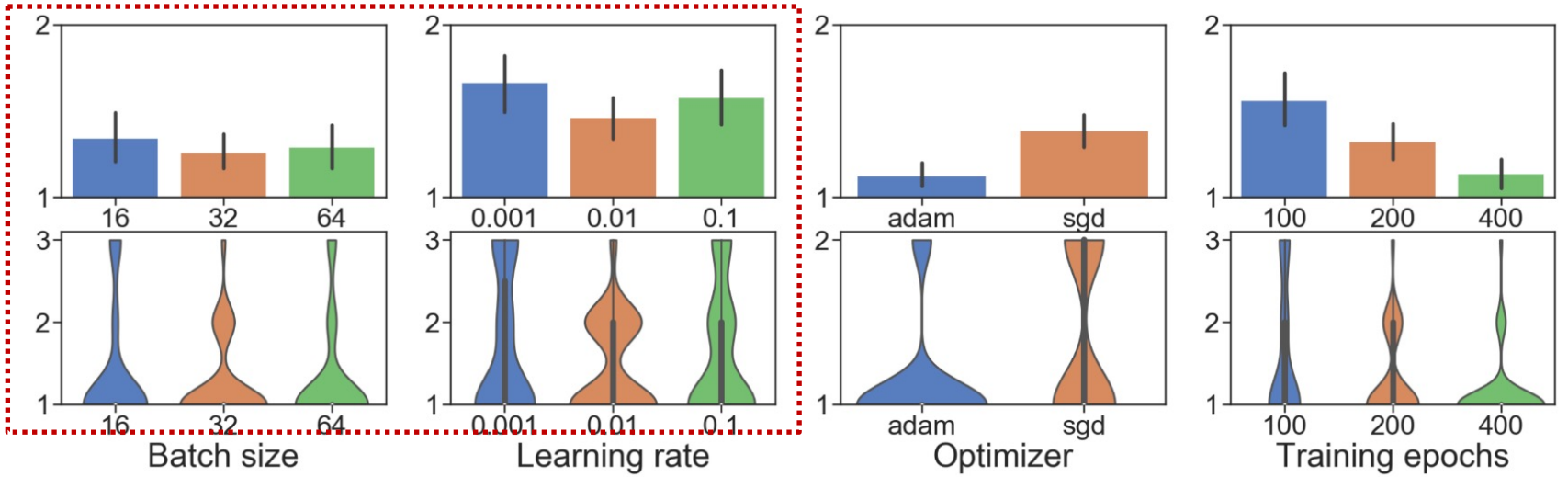
Explanation:
Skip connection generally improves performance

Results 1: A Guideline for GNN Design (3)

- Certain design choices exhibit clear advantages
  - Learning configurations

Optimal batch size and learning rate is hard to decide

Highly dependent on the task



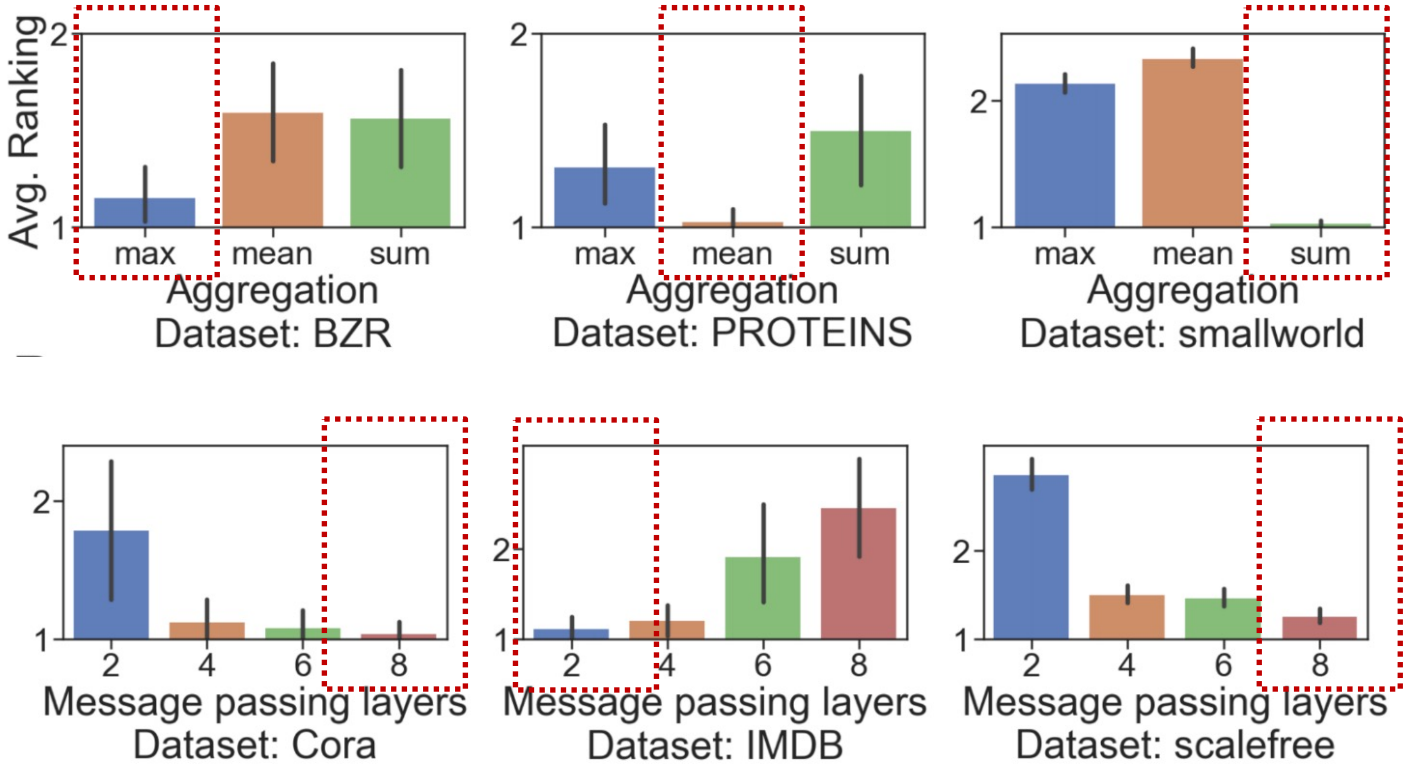
Explanation:

Adam is more robust

More training epochs is useful

Results 2: Understanding GNN Tasks (1)

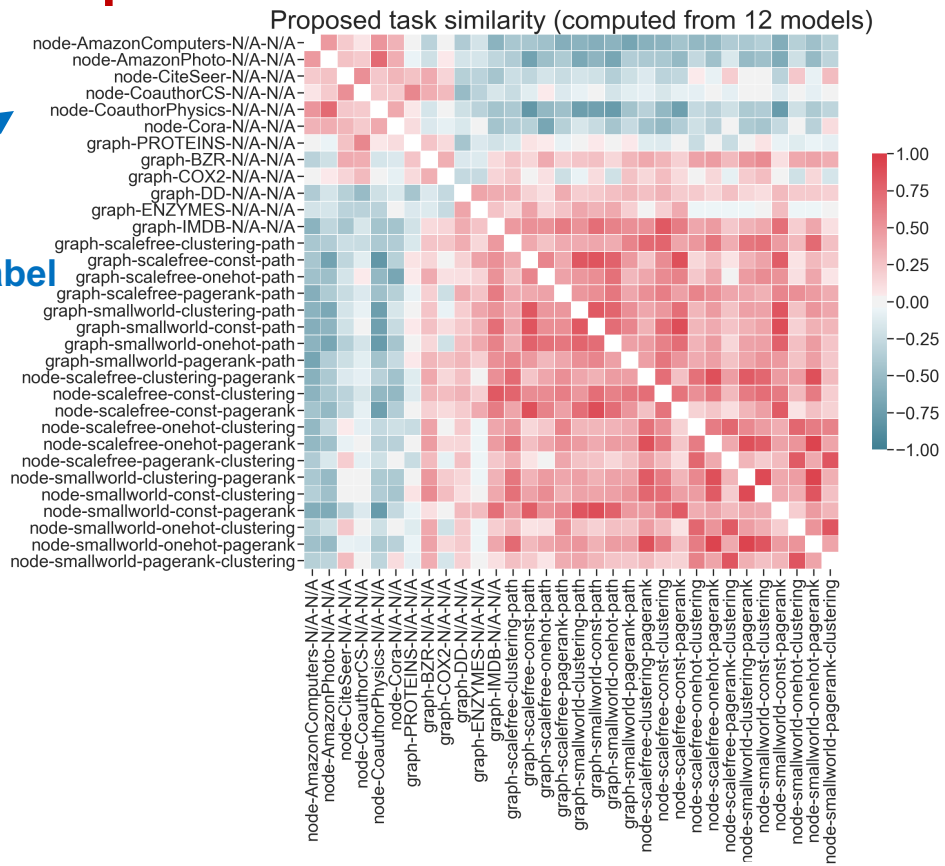
- **Best GNN designs in different tasks vary significantly**
  - Motivate that studying a task space is crucial



Results 2: Understanding GNN Tasks (2)

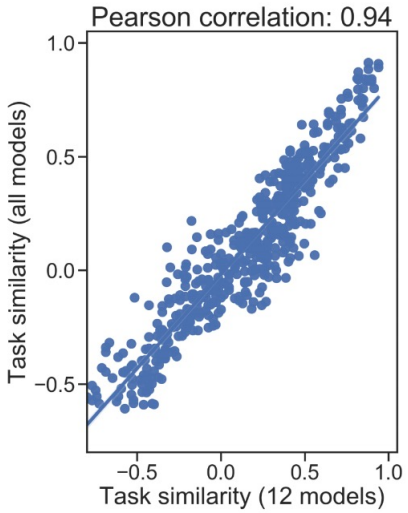
- **Build a GNN task space**
  - We compute **pairwise similarities between all GNN tasks**

Format:
Level-Dataset-Feature-Label



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

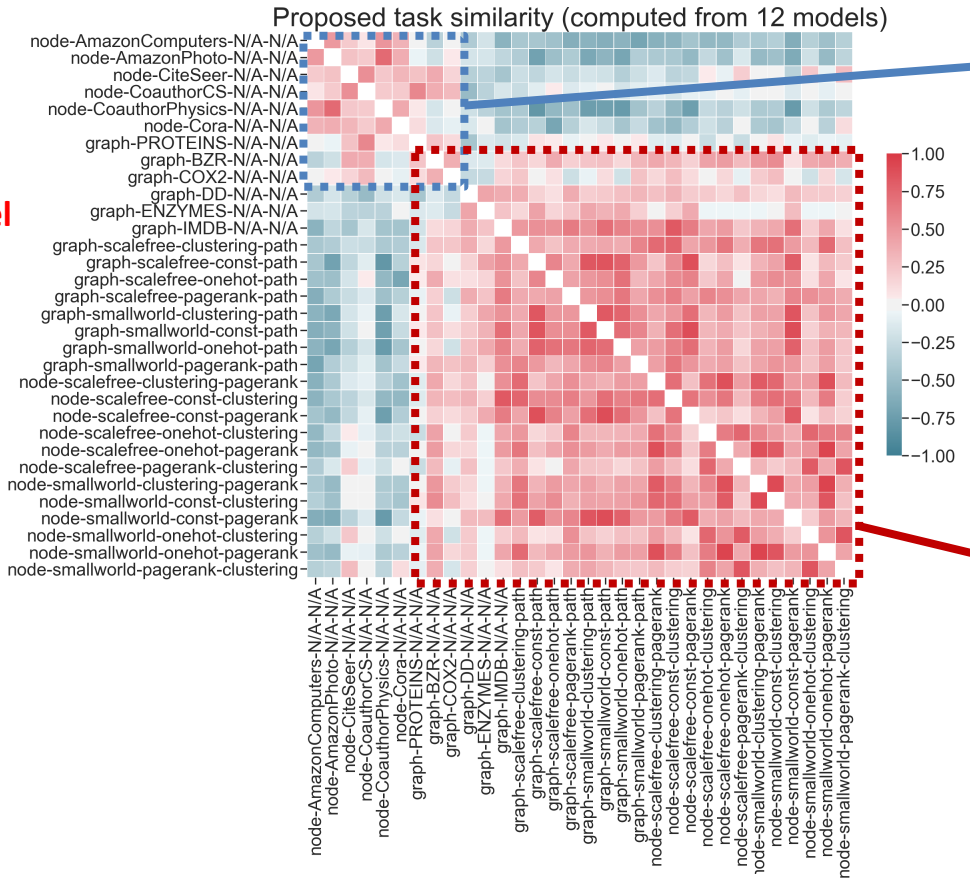


Task similarity computation is cheap: Using **12 anchor models** is a good approximation!

Results 2: Understanding GNN Tasks (3)

- **Proposed GNN task space is informative**

Format:
Level-Dataset-Feature-Label



Group 1:
Tasks rely on **feature information**
Node/graph classification tasks, where input graphs have high-dimensional features

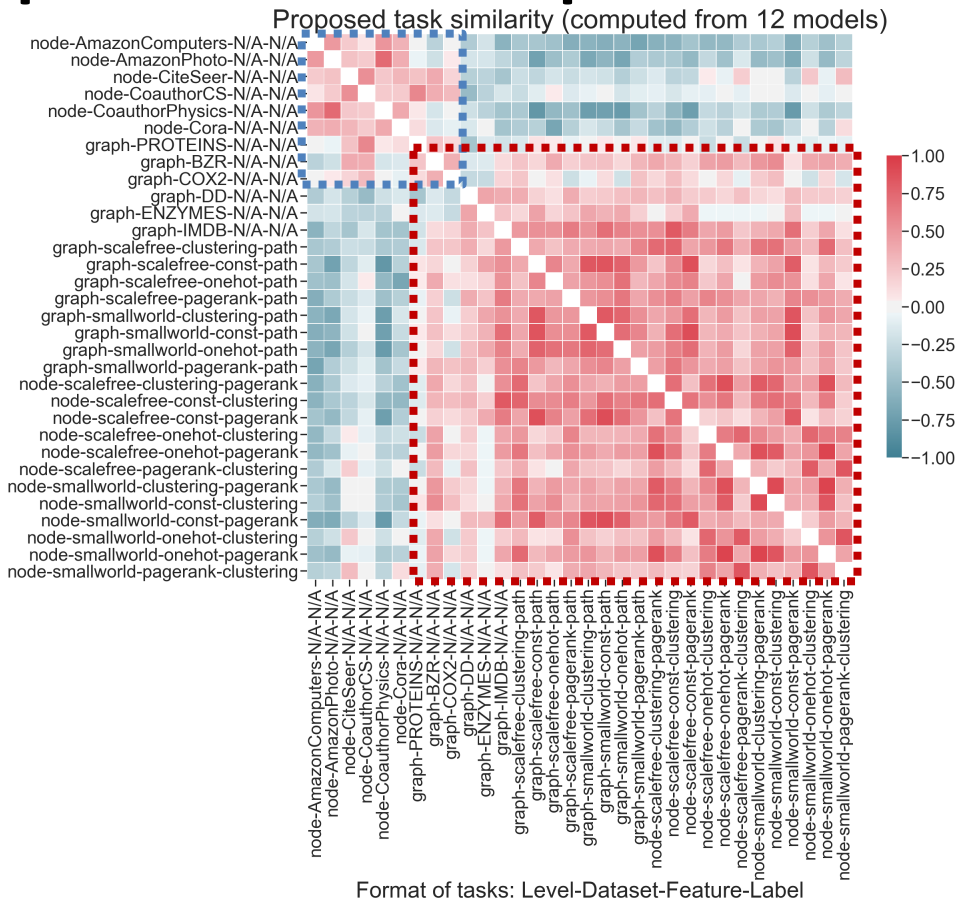
- E.g., Cora graph has 1000+ dim node feature

Group 2:
Tasks rely on **structural information**
Nodes have few features
Predictions are highly dependent on graph structure

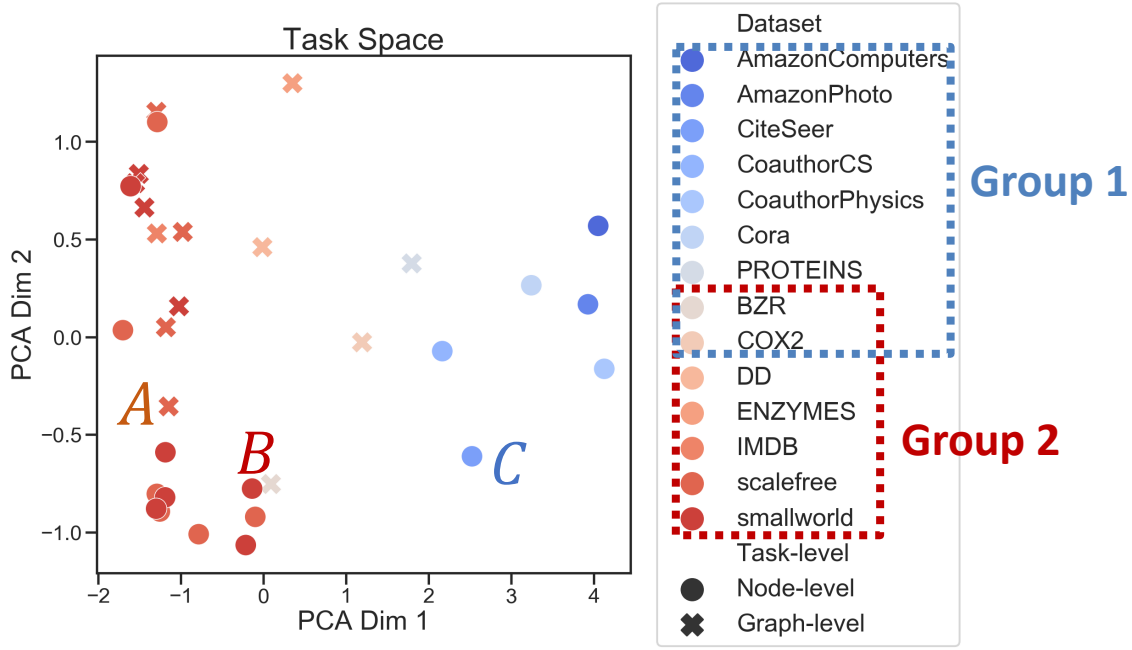
- E.g., Predicting clustering coefficients

Results 2: Understanding GNN Tasks (4)

- **Proposed GNN task space is informative**



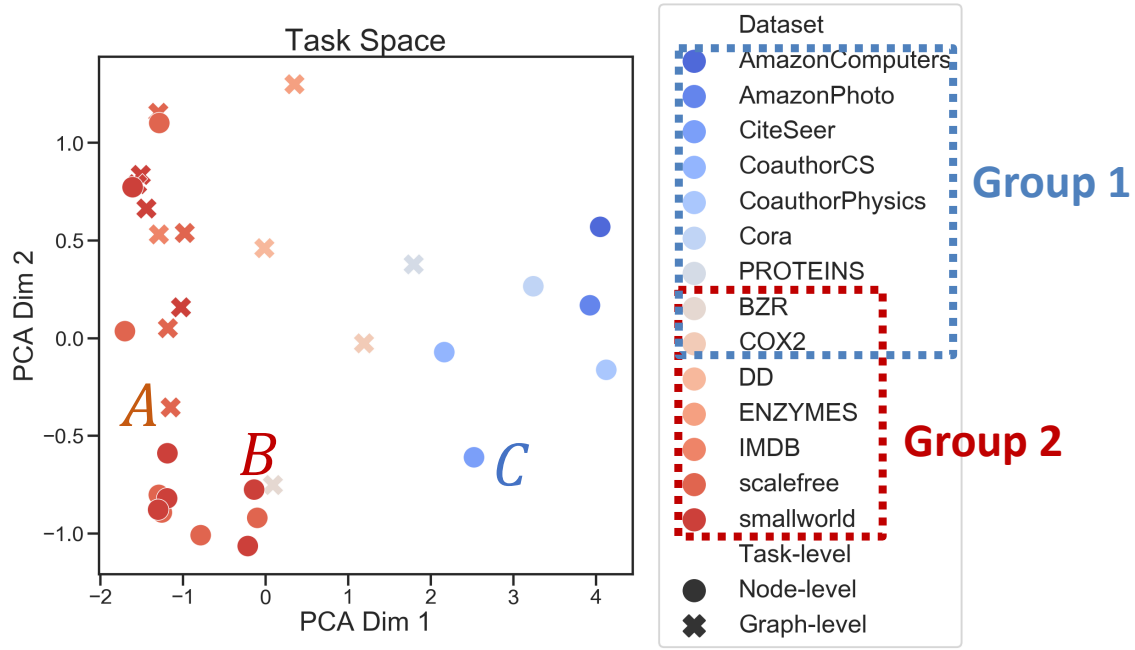
PCA



Results 2: Understanding GNN Tasks (5)

- **Proposed GNN task space is informative**

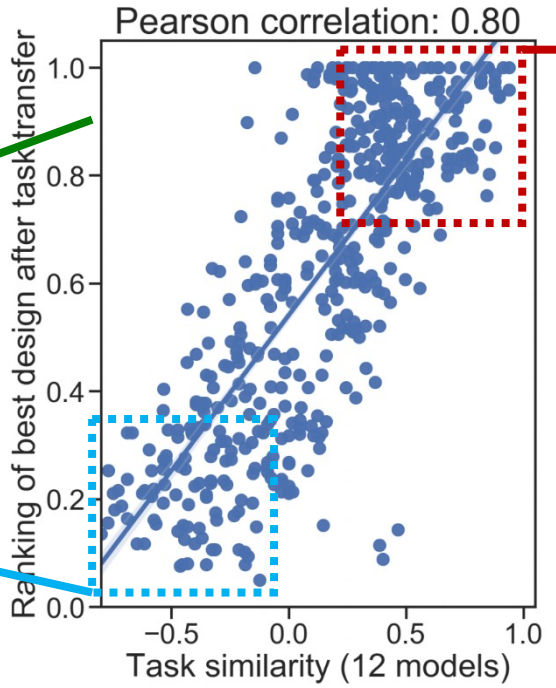
Similar tasks have similar best architectures:
When Task A and B are similar, the best design in A performs well in B



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 2: Understanding GNN Tasks (6)

- **GNN task space can guide task transfer**
  - Evidence: Pearson correlation between task similarity and the performance after transferring architecture



When Task A and B are similar, the best design can transfer well

Metric: **Kendall rank correlation**
1.0 if rankings are exactly the same

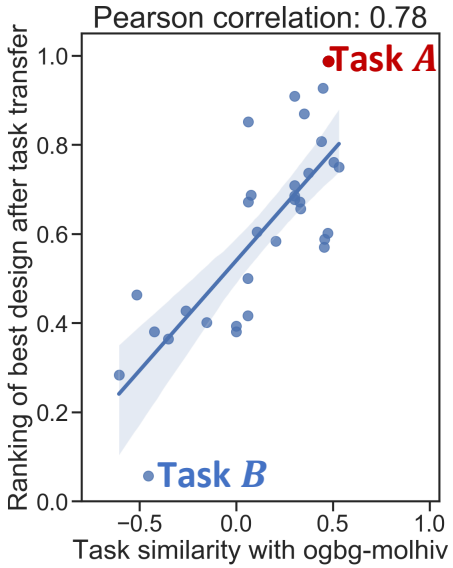
When Task A and B are not similar, the best design in Task A performs miserably in Task B

Results 3: Transfer to Novel Tasks (1)

- **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:

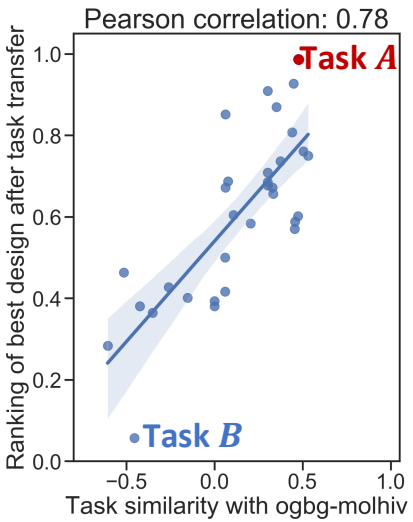
- **Step 1:** Measure 12 anchor model performance on the new task
- **Step 2:** Compute similarity between the new task and existing tasks
- **Step 3:** Recommend the best designs from existing tasks with high similarity



Results 3: Transfer to Novel Tasks (2)

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

Pick Task A and Task B:
Task A: Similar to OGB
Task B: Not similar to OGB



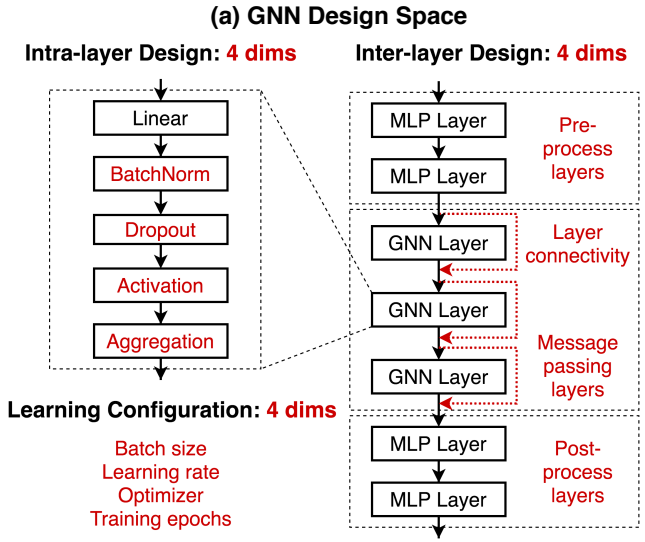
Transfer the best model from Task A achieves SOTA on OGB
Transfer the best model from Task B performs badly on OGB

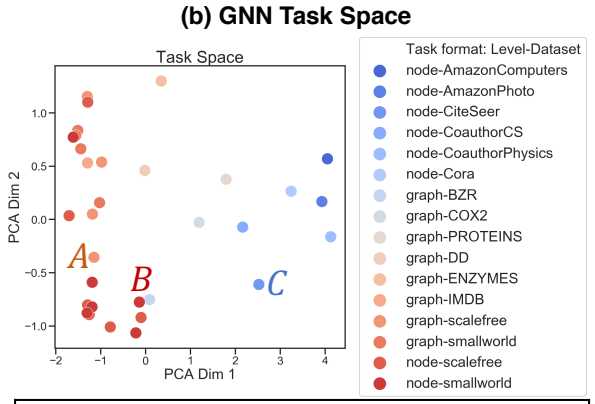
	Task A: graph-scalefree-const-path	Task B: node-CoauthorPhysics
Best design in our design space	(1, 8, 3, skipcat, sum)	(1, 4, 2, skipcat, max)
Task Similarity with ogbg-molhiv	**0.47**	**-0.61**
Performance after transfer to ogbg-molhiv	***0.785***	***0.736***

Previous SOTA on OGB: 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**





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

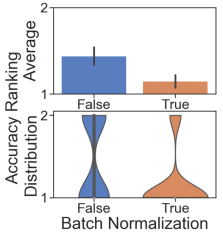
(c) Controlled Random Search

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



Outline of Today's Lecture

1. Design Space of GNN

2. AutoML based on Meta-model

Bayesian Optimization for AutoML (1)

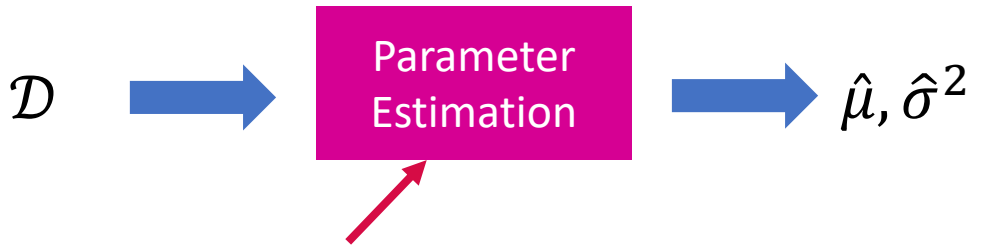
- A **non-ML** approach for searching the best hyperparameters
- Build a probabilistic model of the function mapping from hyperparameter settings to the objective
  - Assume that the probabilistic model follows **Gaussian distribution**
- Notation
  - $f(\cdot)$: a model architecture
  - $\mathcal{X}$: hyperparameter space, $x_i \sim \mathcal{X}$: a hyperparameter setting
  - $y_i = f(x_i)$: the performance of $f(\cdot)$ with hyperparameter x_i
  - M: the probabilistic model mapping x_i to y_i

Bayesian Optimization for AutoML (2)

- We assume that the mapping from hyperparameters to the performance follows the **Gaussian distribution**
  - μ: mean of distribution, σ^2 : variance of distribution

$$p(y|x) = N(\mu, \sigma^2)$$

- Given a set of hyperparameter and corresponding evaluated performance $\mathcal{D} = \{(x_i, y_i)\}$, we can **estimate** the μ, σ^2



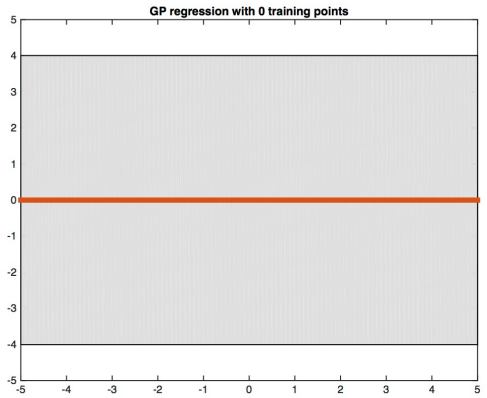
A widely used estimator is Gaussian Processes (GP)

More details on estimation can be found in Bayesian Optimization Primer

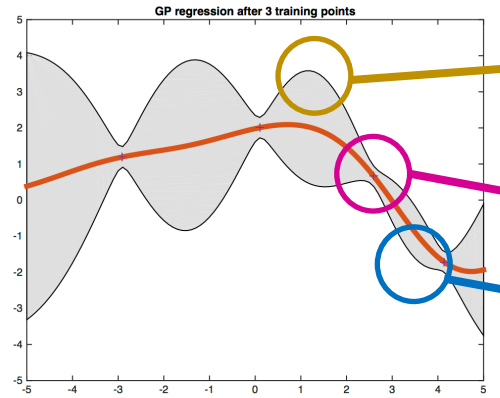
Bayesian Optimization for AutoML (3)

- Pipeline of Bayesian Optimization
  - 1) Initialize a dataset $\mathcal{D} = \{(x_i, y_i)\}$ by sampling a set of hyperparameters and computing the corresponding performance metric
  - 2) **Estimate** the probabilistic model M based on $\mathcal{D}$
  - 3) Sample the **top-performing** hyperparameter setting x_{top} from $\mathcal{D}$ based on M
  - 4) Compute the performance of x_{top}, and add (x_{top}, y_{top}) to the $\mathcal{D}$
  - 5) Back to step 1 until the dataset size reaches a certain number

Initial estimation:



After 4 steps



High uncertainty

Low uncertainty

Next sample

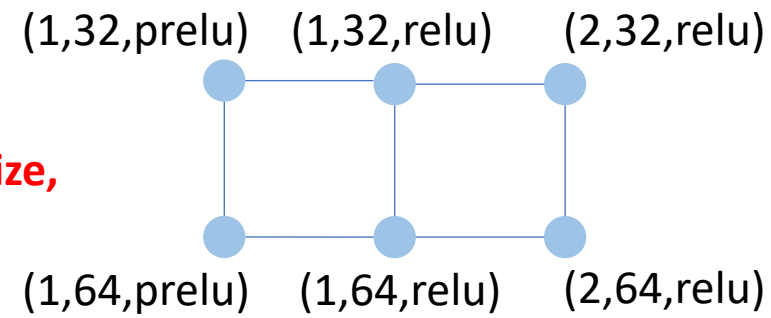
Uncertainty / optimality tradeoff

Inductive bias for GNN Design

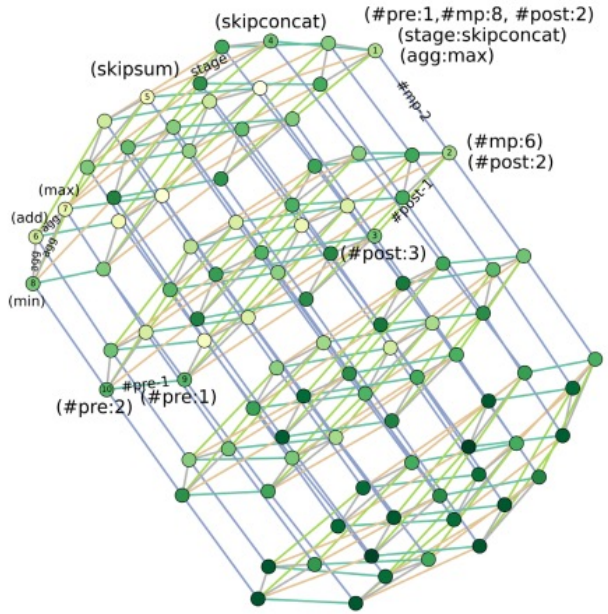
- We usually get some **insights** from the previous tasks
  - E.g., setting batch normalization of a 3-layer and a 4-layer GAT to false both degrade the performance
- Such inductive bias can provide valuable information in searching for the best-performing model
  - E.g., infer that a 3-layer GraphSAGE with batch normalization outperforms the one without
- We can utilize the **existing model design insights** to quickly find the best design for the **new task**!
- How to capture the relation between the different designs?

Design Graph (1)

- Design graph $\mathcal{G}(\mathcal{N}, \mathcal{E})$
  - $\mathcal{N}$ is the node set includes the candidate designs
  - $\mathcal{E}$ is the edge set which denotes the similarity between two designs
- Two designs that **only differ in one design dimension** are connected
  - For example, considering three design dimensions: **layer number** `[1,2,3]`, **hidden size** `[32,64,128]`, **activation function** `[relu,prelu]`:

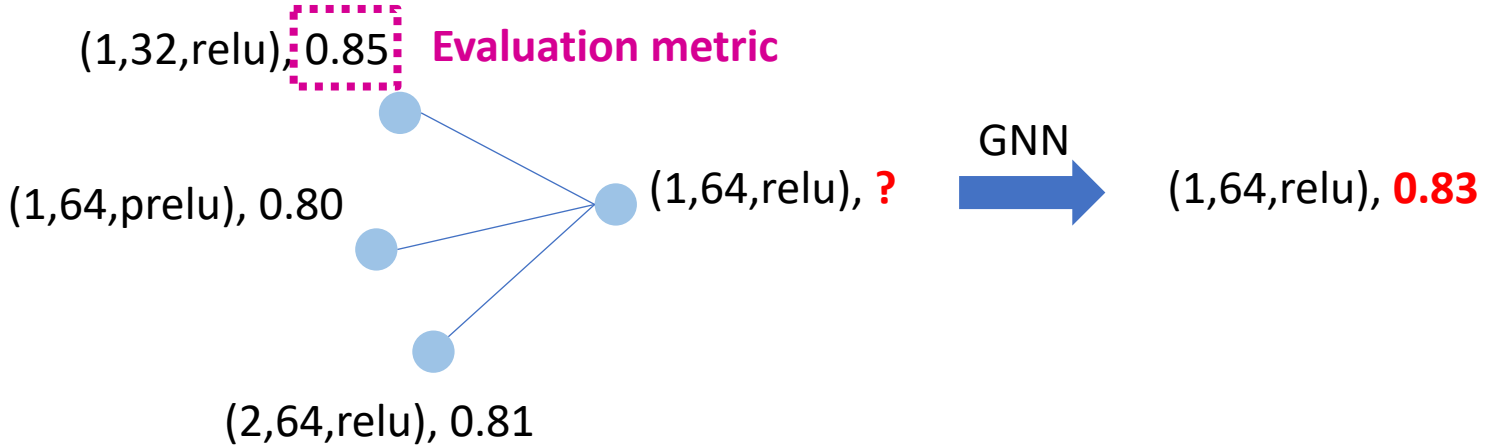


Format:
(layer number, hidden size, activation)



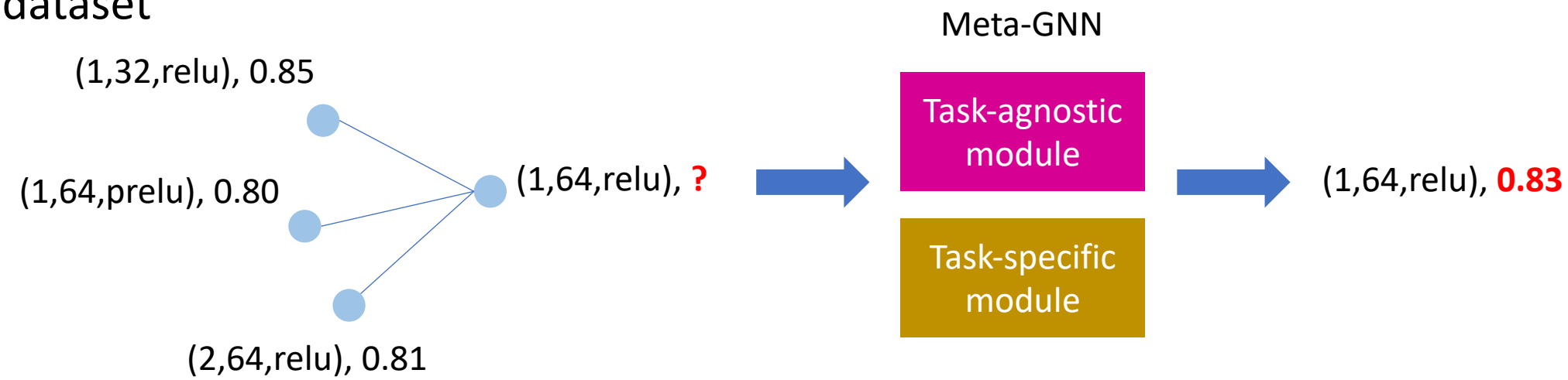
Design Graph (2)

- Design graph connects the similar model designs
- We can use the adjacent designs to predict the performance of a new design
  - GNN is a suitable tool to conduct node-level prediction on a graph
  - **Intuition**: similar architectures have similar performance



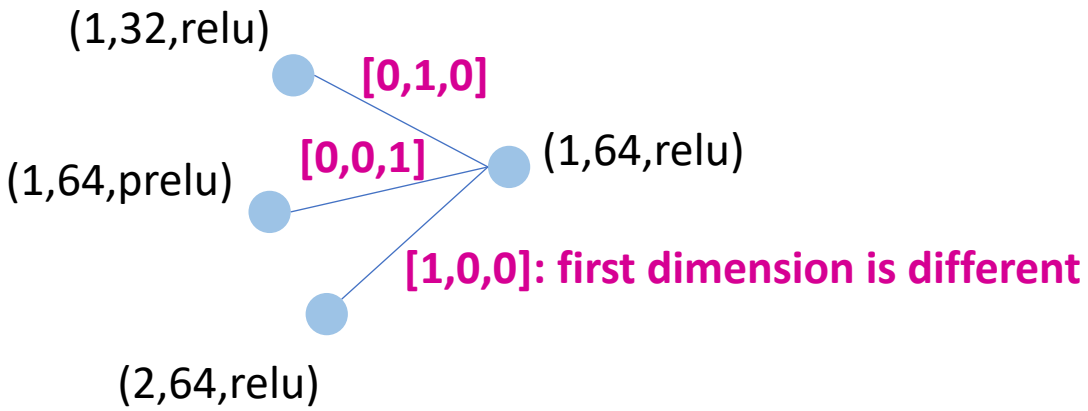
Meta-GNN

- To better predict the performance, here we combine two modules
  - **Task-agnostic** module. A GNN model is used to perform message passing and output node representation
  - **Task-specific** module. Propagate the information of design performance on the target dataset



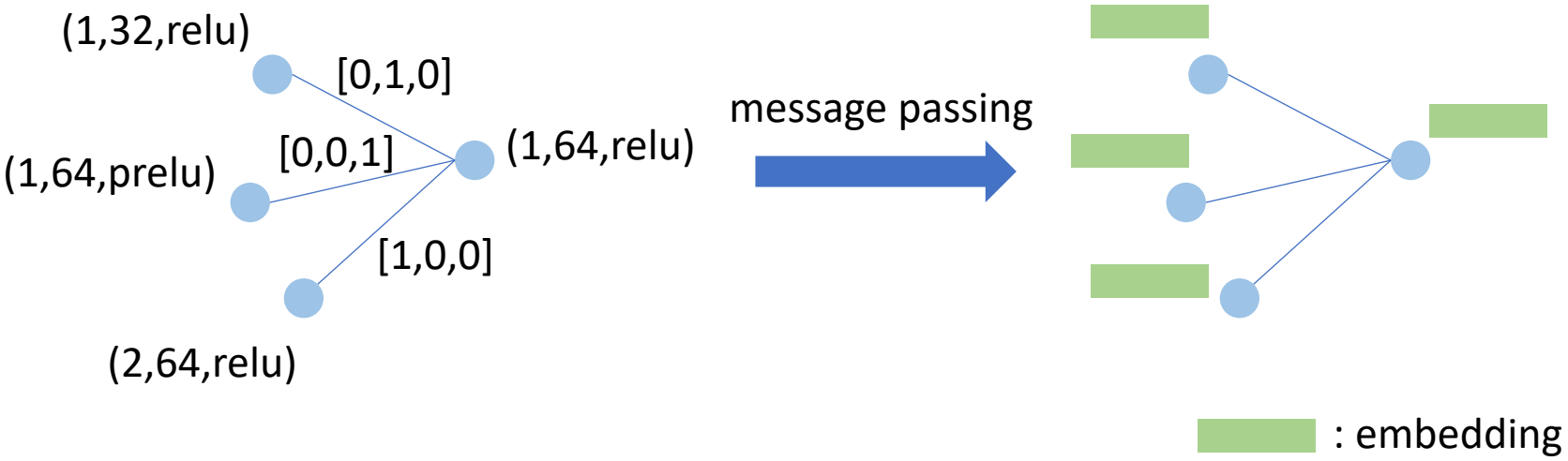
Meta-GNN: Task-agnostic Module (1)

- Task-agnostic Module
  - 1) compute the node and edge features
    - Node feature: the concatenation of the feature encoding of each design dimension
    - Edge feature: the one-hot encoding of design dimension difference



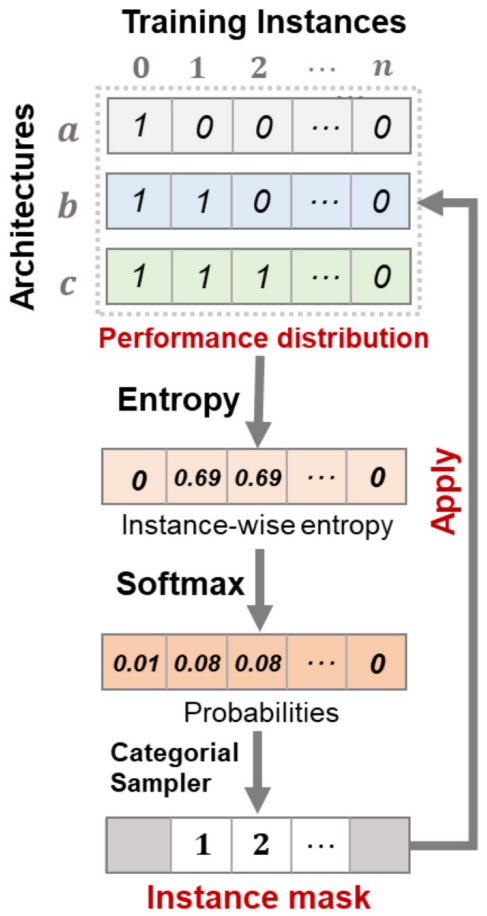
Meta-GNN: Task-agnostic Module (2)

- Task-agnostic Module
  - 2) perform message passing to get the node representation



Meta-GNN: Task-specific Module (1)

- Task-specific Module
  - 1) identifying **critical instances**: identify the training instances that results in different performances across different designs
    - Use the **explored designs** to provide instance-wise performances
    - Compute the **entropy** of each training instance's performance
    - Obtain the instance–wise **probability** vis Softmax
    - Sample instances that **result in high variation** across designs

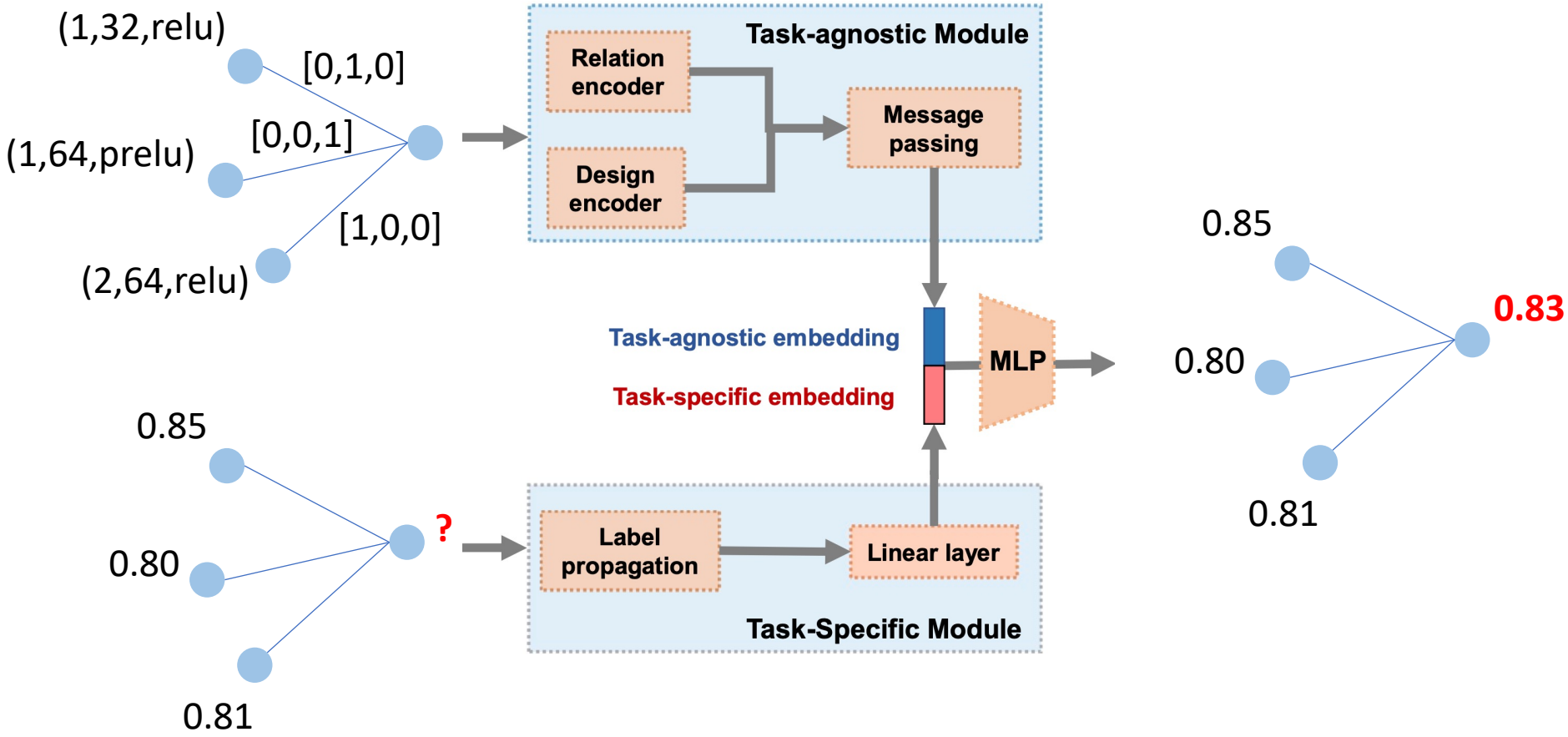


Meta-GNN: Task-specific Module (2)

- Task-specific Module
  - 2) **label propagation** and linear projection: make the task-specific information available to all candidate designs
    - Compute the performance of explored designs on critical instances
    - Propagate the **label information** and use a linear layer to project the information to an embedding
    - $N(v)$: neighbor nodes set, k: propagate number, α: scalar, y: aggregated label

$$y_v^{(k+1)} = \alpha \sum_{u \in N(v)} \frac{1}{|N(v)||N(u)|} y_u^{(k)} + (1-\alpha) y_v^{(k)}$$

Meta-GNN: Overview



Objective for Meta-GNN

- Use the performance of explored designs to guide the optimization of meta-GNN
  - Apply MSE loss and pair-wise rank loss. λ: scalar

$$L = L_{mse} + \lambda L_{rank}$$

  - N: number of explored designed, $\hat{y}$:predicted performance, y: performance, τ: temperature

$$L_{mse} = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

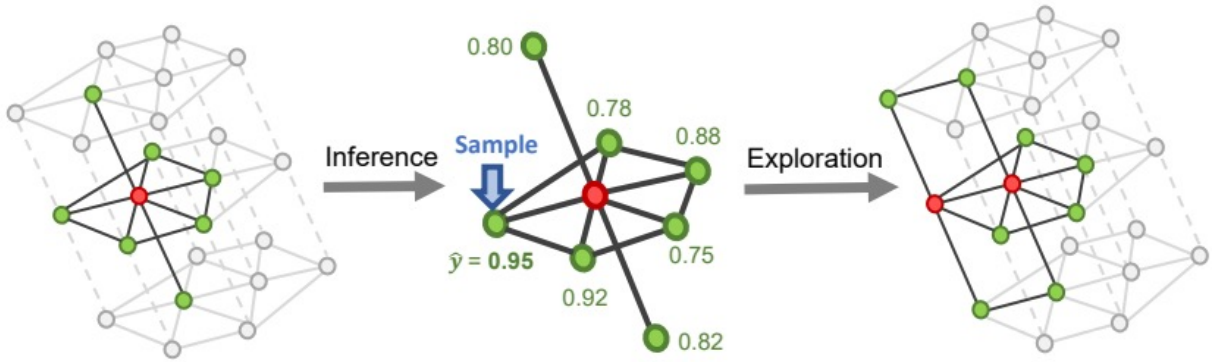
$$L_{rank} = \sum_{i=1}^{N} \sum_{j=1}^{N} (-1)^{\mathbb{I}(y_i > y_j)} \cdot \sigma\left(\frac{\hat{y}_i - \hat{y}_j}{\tau}\right)$$

if y_i is greater than y_j, $\hat{y}_i$ should greater than $\hat{y}_j$ as well and vice versa

Whether y_i is greater than y_j

Search on New Task

- Pipeline
  - Start with a few random model designs
  - Train the meta-GNN on these designs, using performance as supervision
  - Use meta-GNN to decide what nodes to explore
    - Among **fringe nodes**
    - **3-hop neighborhood**
  - Repeat until reaching max num steps



Explored nodes (obtained performance through training)

Candidate designs to be considered to explore next

Summary

- Design space of GNN
  - A general GNN task space
  - Evaluate the GNN design by anchor models
  - Measure the similarity of different tasks
- AutoML based on Meta-model
  - Use design graph to capture the relation between different designs
  - Meta-GNN predicts the performance of unexplored design based on adjacent explored designs