

Towards Graph Foundation Models

WWW 2024 Tutorial

Philip S. Yu, Chuan Shi, Cheng Yang, Yuan Fang, Lichao Sun



SINGAPORE
MANAGEMENT
UNIVERSITY





Towards Graph Foundation Models

Part II: GNN-based Methods

Cheng Yang

yangcheng@bupt.edu.cn

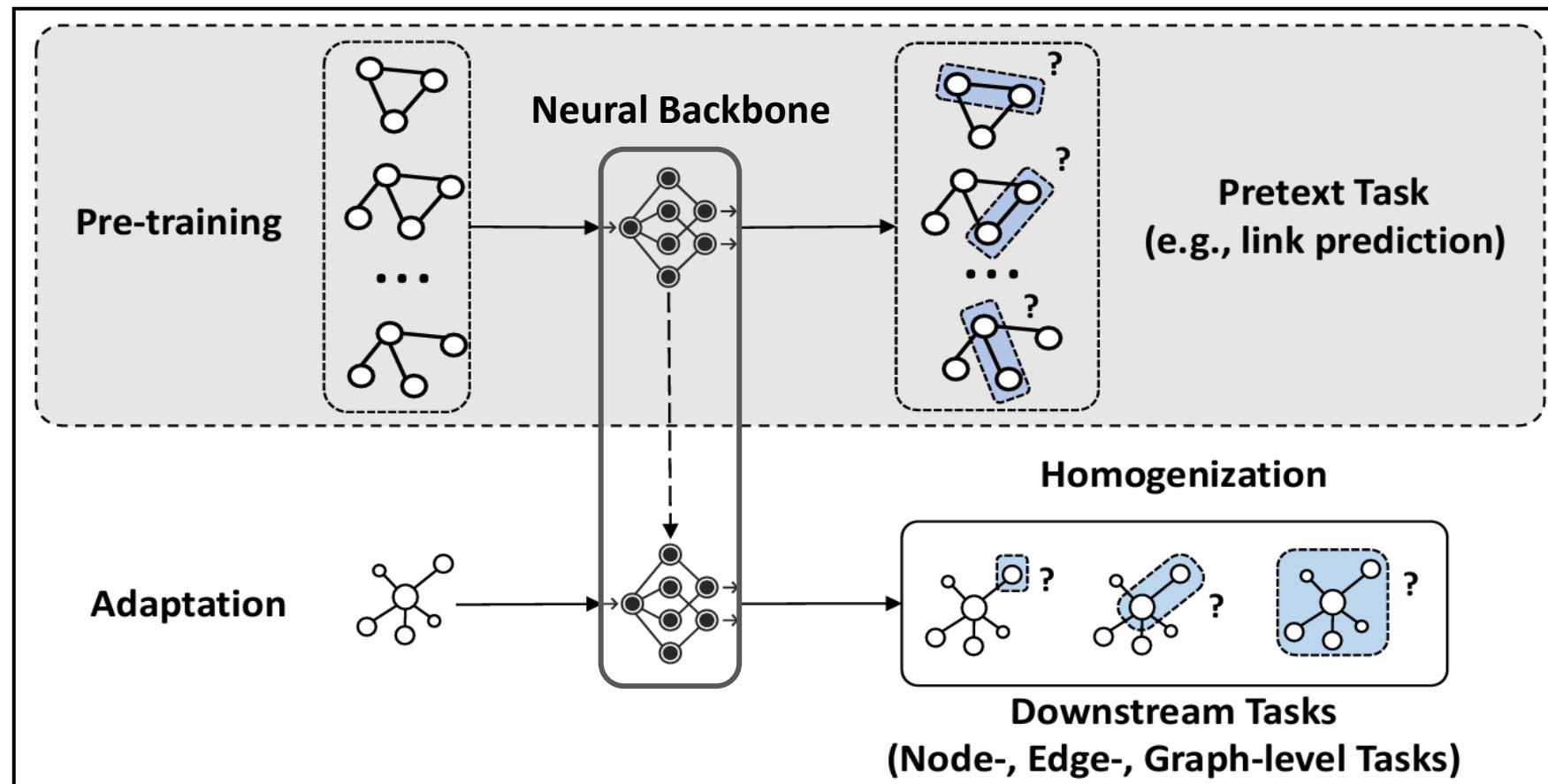
Beijing University of Posts and Telecommunications



GNN-based Methods

Backbone: No unified architecture
(Message Passing/Graph Transformer)

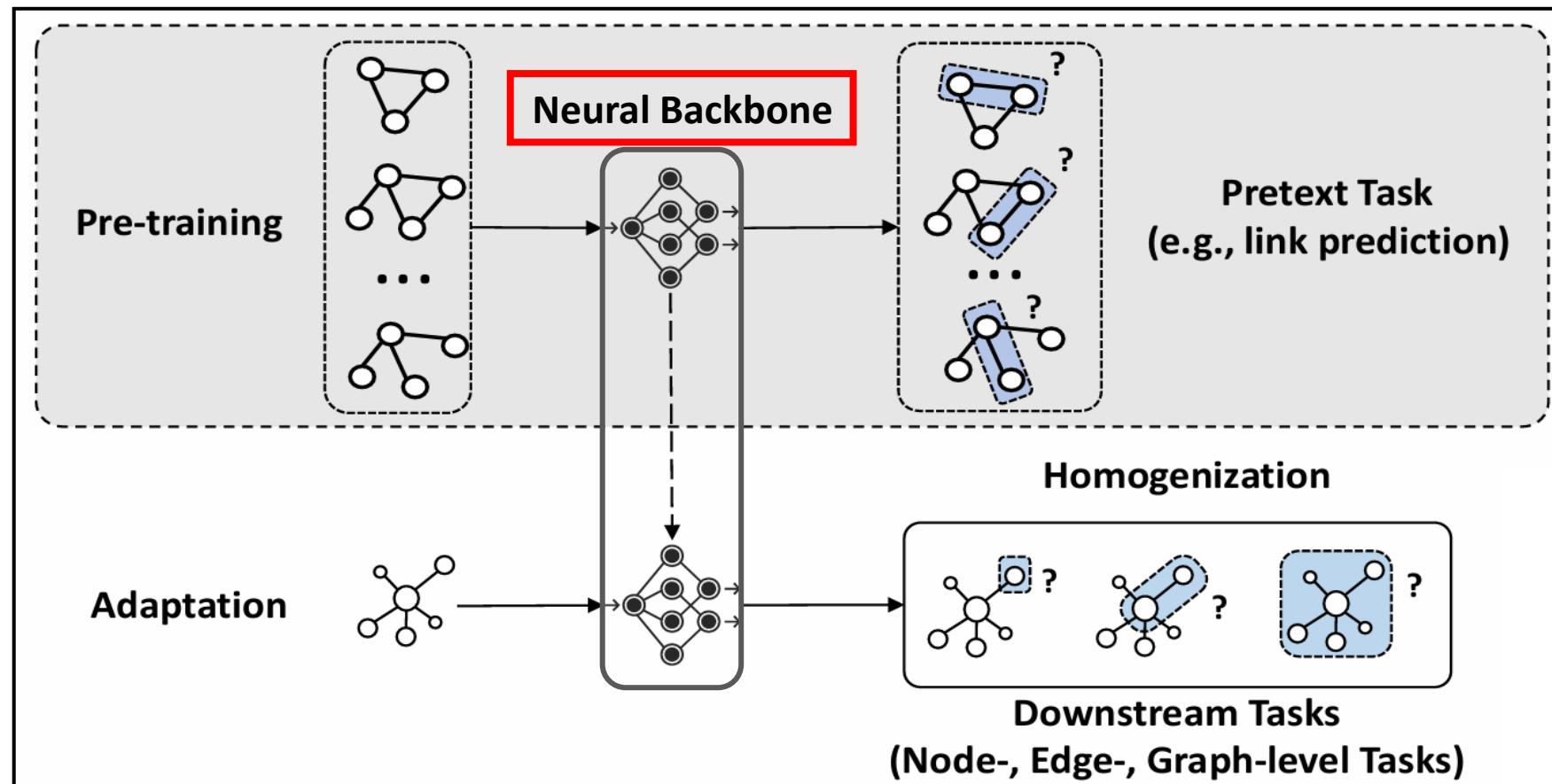
Paradigm: Pre-training + Adaptation



GNN-based Methods

Backbone: No unified architecture
(Message Passing/Graph Transformer)

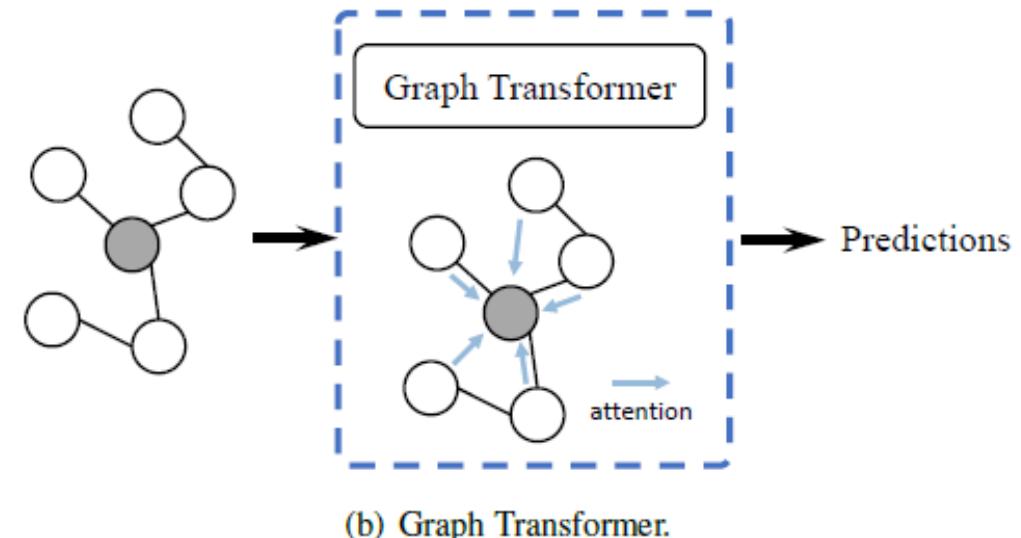
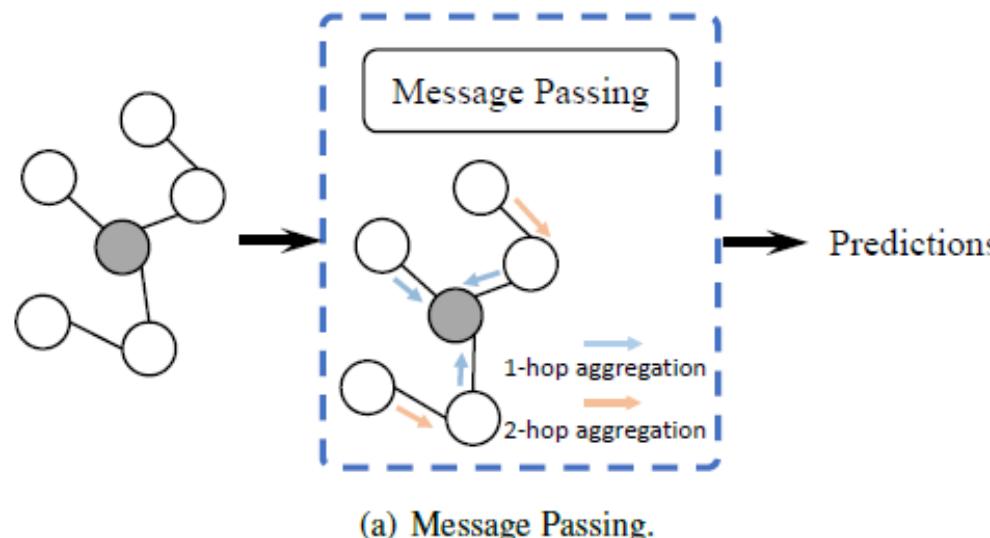
Paradigm: Pre-training + Adaptation



Backbone Architecture

Backbone Architecture

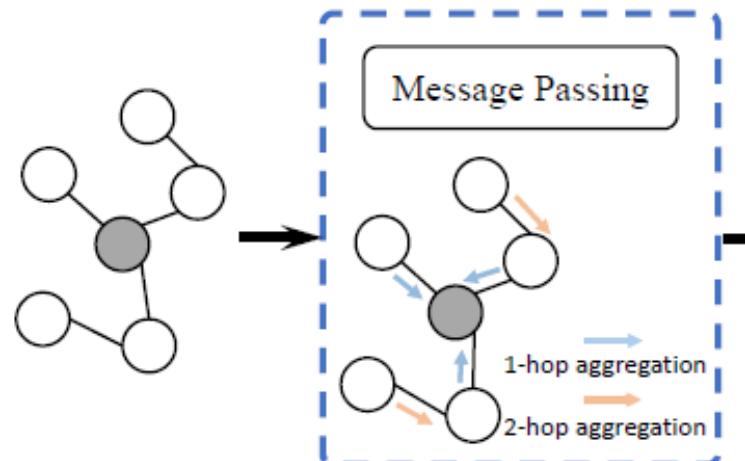
- Message Passing: propagates information between nodes that are **explicitly connected** in the graph structure
- Graph Transformer: considers and measures the similarity between **every pair of nodes** in the graph



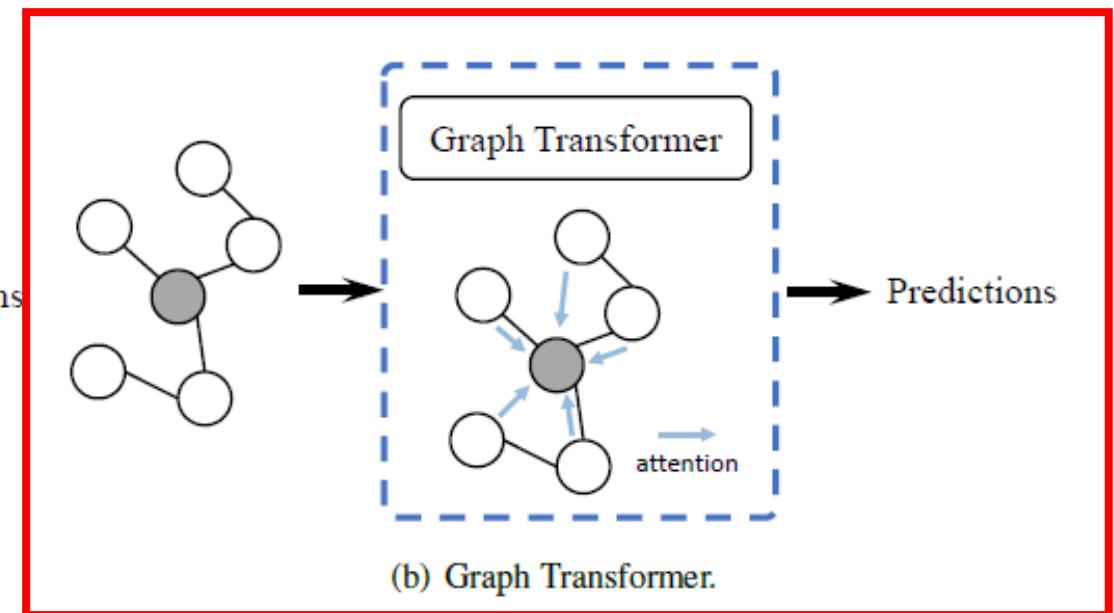
Backbone Architecture

Backbone Architecture

- Message Passing: propagates information between nodes that are **explicitly connected** in the graph structure
- Graph Transformer: considers and measures the similarity between **every pair of nodes** in the graph



(a) Message Passing.



(b) Graph Transformer.

Graphomer

Motivation:

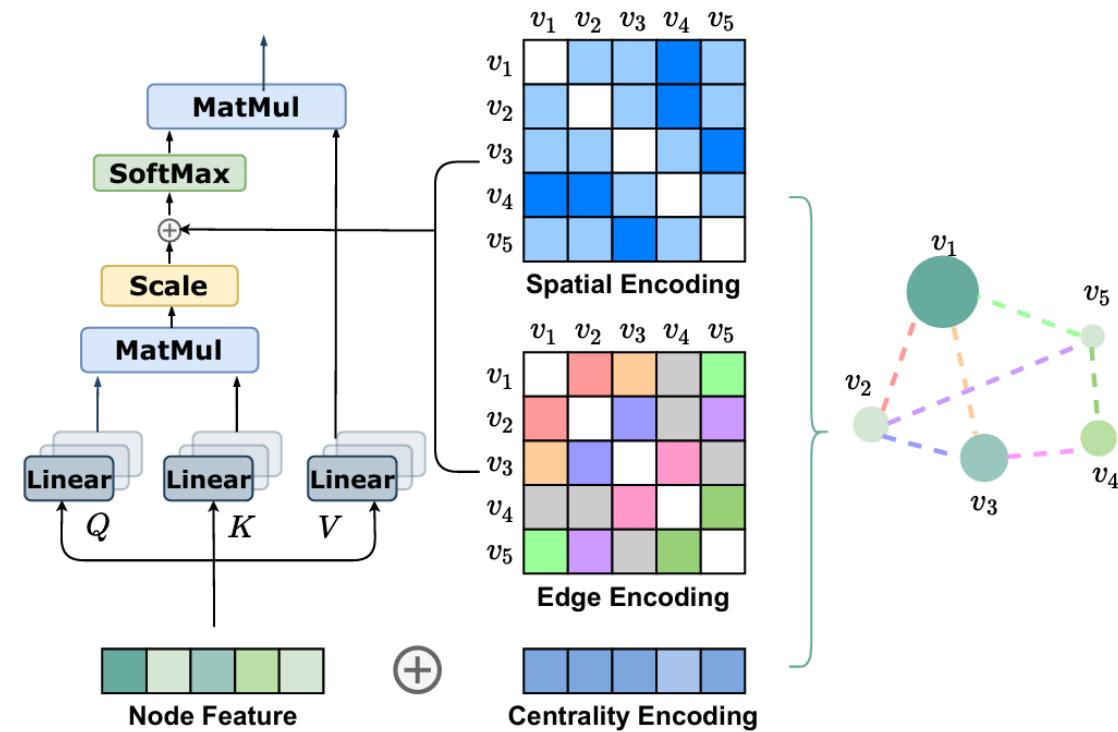
- The Transformer is well acknowledged as the **most powerful** neural network in modelling sequential data, such as natural language and speech.
- Model variants built upon Transformer have also been shown **great performance** in computer vision and programming Language.
- However, Transformer has still not been the de-facto standard on public graph representation leaderboards.

Whether Transformer architecture is suitable to model graphs and how to make it work in graph representation learning?

Graphomer

Core idea:

- properly incorporate structural information of graphs into the model.
 - propose a **Centrality Encoding** to capture the node importance in the graph.
 - propose a novel **Spatial Encoding** to capture the structural relation between nodes
 - design a new **Edge Encoding** method to take such signal into the Transformer layers.



Graphomer

Structural Encodings in Graphomer :

- Centrality Encoding:
 - develop a Centrality Encoding which assigns each node two real-valued embedding vectors according to its **indegree and outdegree**.

$$h_i^{(0)} = x_i + z_{\deg^-(v_i)}^- + z_{\deg^+(v_i)}^+$$

- Spatial Encoding:
 - choose $\phi(v_i, v_j)$ to be the distance of the **shortest path (SPD)** between v_i and v_j .
- Edge Encoding:
 - find the shortest path $SP_{ij} = (e_1, e_2, \dots, e_N)$ from v_i to v_j , and compute an average of the **dot-products of the edge feature** and a learnable embedding along the path.

$$A_{ij} = \frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d}} + b_{\phi(v_i, v_j)} + c_{ij}, \text{ where } c_{ij} = \frac{1}{N} \sum_{n=1}^N x_{e_n} (w_n^E)^T$$

GRAPH-BERT

Motivation :

- Traditional message passing-based models have **limited representation capabilities**.
- The inherently interconnected nature **precludes large-sized graph parallelization**, as memory constraints limit batching across the nodes.
- Existing GNN models have several serious learning performance problem, e.g., suspended animation problem and over-smoothing problem.

Zhang J, Zhang H, Xia C, et al. Graph-bert: Only attention is needed for learning graph representations. arXiv 2020[J]. arXiv preprint arXiv:2001.05140, 2001.

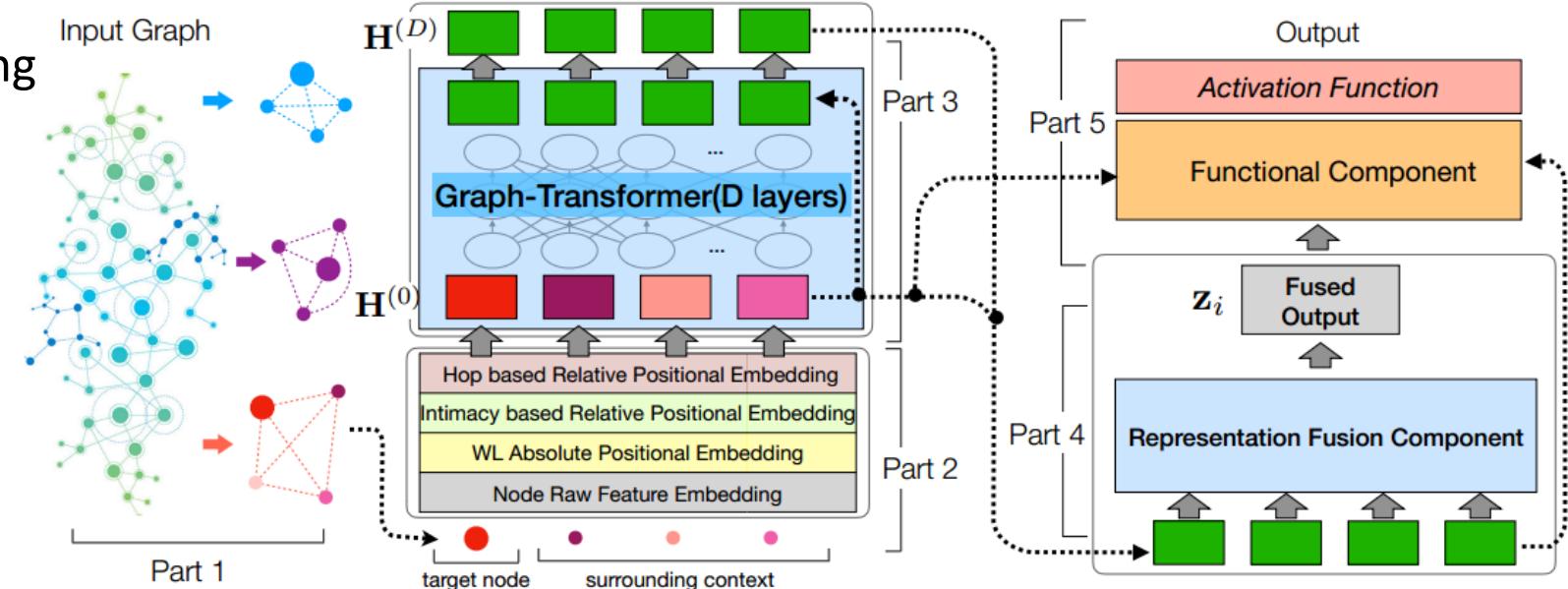
GRAPH-BERT

Part 1: linkless subgraph batching instead of the complete graph

Part 2: Node Input Vector Embeddings

1. raw feature vector embedding
2. Weisfeiler-Lehman absolute role embedding
3. intimacy-based relative positional embedding
4. hop-based relative distance embedding

Part 3: graph transformer based encoder



GRAPH-BERT

How to handle large-scale graph input?

- Linkless Subgraph Sampling and Batching
 - Introduce the top-k intimacy sampling approach and **trained with linkless subgraph batches** sampled from the input graph instead of complete graph

How to deal with insufficient model representation?

- More node-related information as node initial features

1. raw feature vector embedding

$$\mathbf{e}_j^{(x)} = \text{Embed}(\mathbf{x}_j) \in \mathbb{R}^{d_h \times 1}$$

2. Weisfeiler-Lehman absolute role embedding

$$\begin{aligned}\mathbf{e}_j^{(r)} &= \text{Position-Embed}(\text{WL}(v_j)) \\ &= \left[\sin\left(\frac{\text{WL}(v_j)}{10000^{\frac{2l}{d_h}}}\right), \cos\left(\frac{\text{WL}(v_j)}{10000^{\frac{2l+1}{d_h}}}\right) \right]_{l=0}^{\frac{d_h}{2}}\end{aligned}$$

3. intimacy-based relative positional embedding

$$\mathbf{e}_j^{(p)} = \text{Position-Embed}(\text{P}(v_j)) \in \mathbb{R}^{d_h \times 1}$$

4. hop-based relative distance embedding

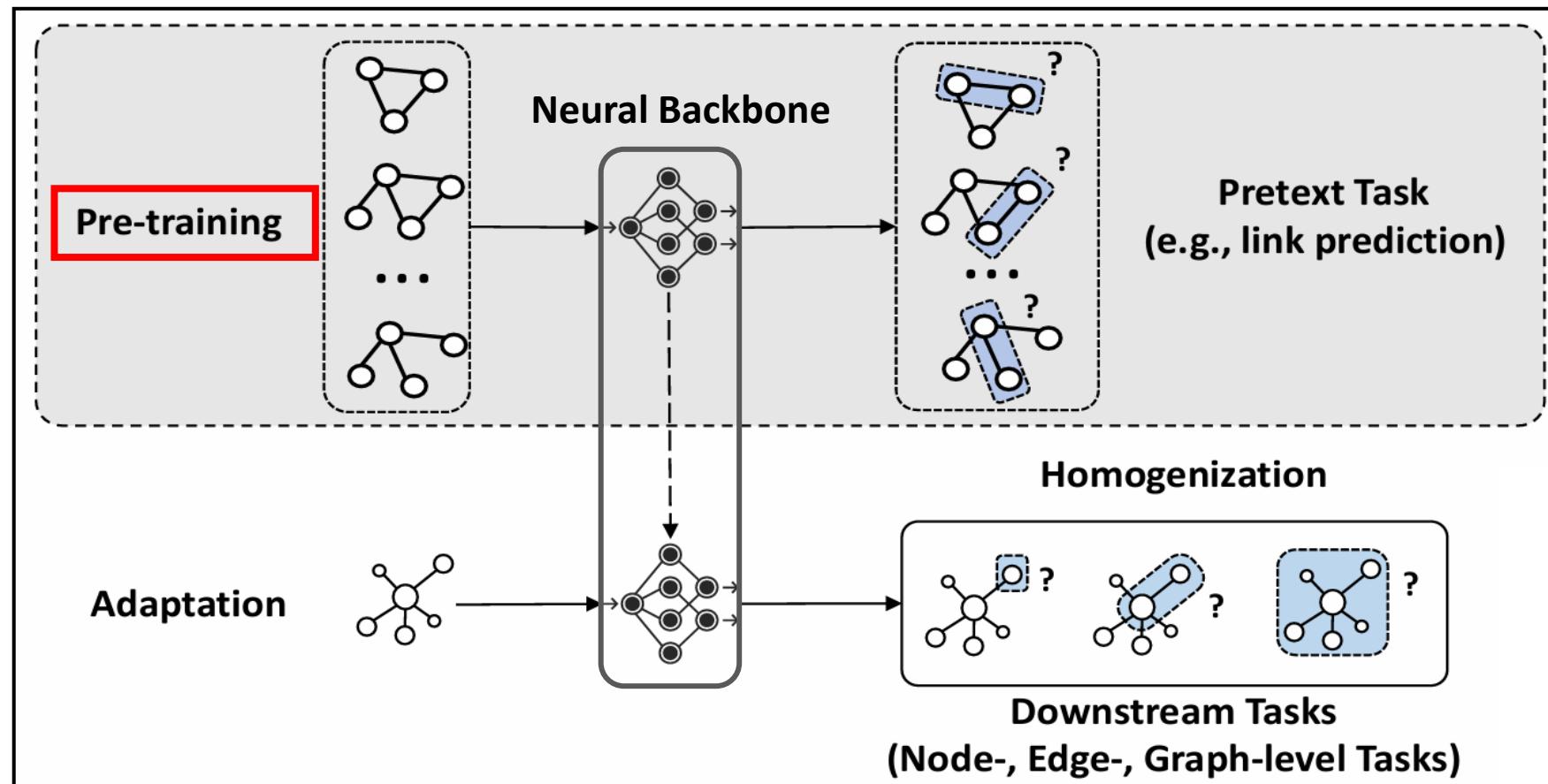
$$\mathbf{e}_j^{(d)} = \text{Position-Embed}(\text{H}(v_j; v_i)) \in \mathbb{R}^{d_h \times 1}$$

$$\mathbf{h}_j^{(0)} = \text{Aggregate}(\mathbf{e}_j^{(x)}, \mathbf{e}_j^{(r)}, \mathbf{e}_j^{(p)}, \mathbf{e}_j^{(d)})$$

GNN-based Methods

Backbone: No unified architecture
(Message Passing/Graph Transformer)

Paradigm: Pre-training + Adaptation



Pre-training

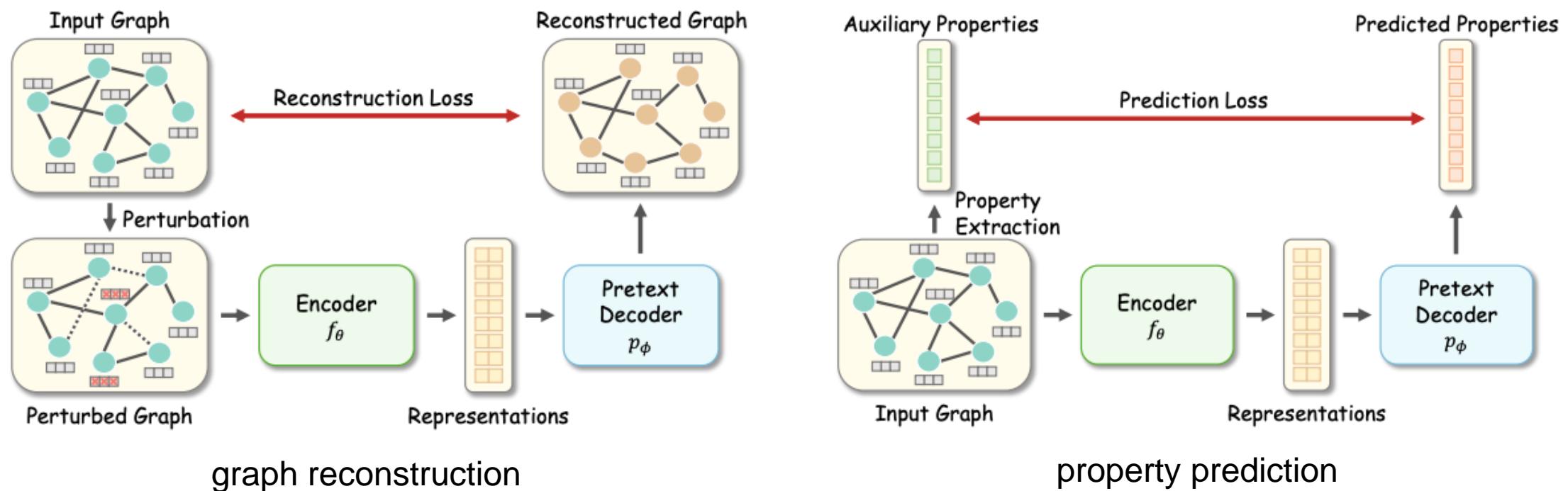
Pre-training

- Generative methods:
 - graph reconstruction
 - property prediction
- Contrastive methods:
 - same-scale contrastive learning
 - cross-scale contrastive learning

Pre-training

Pre-training

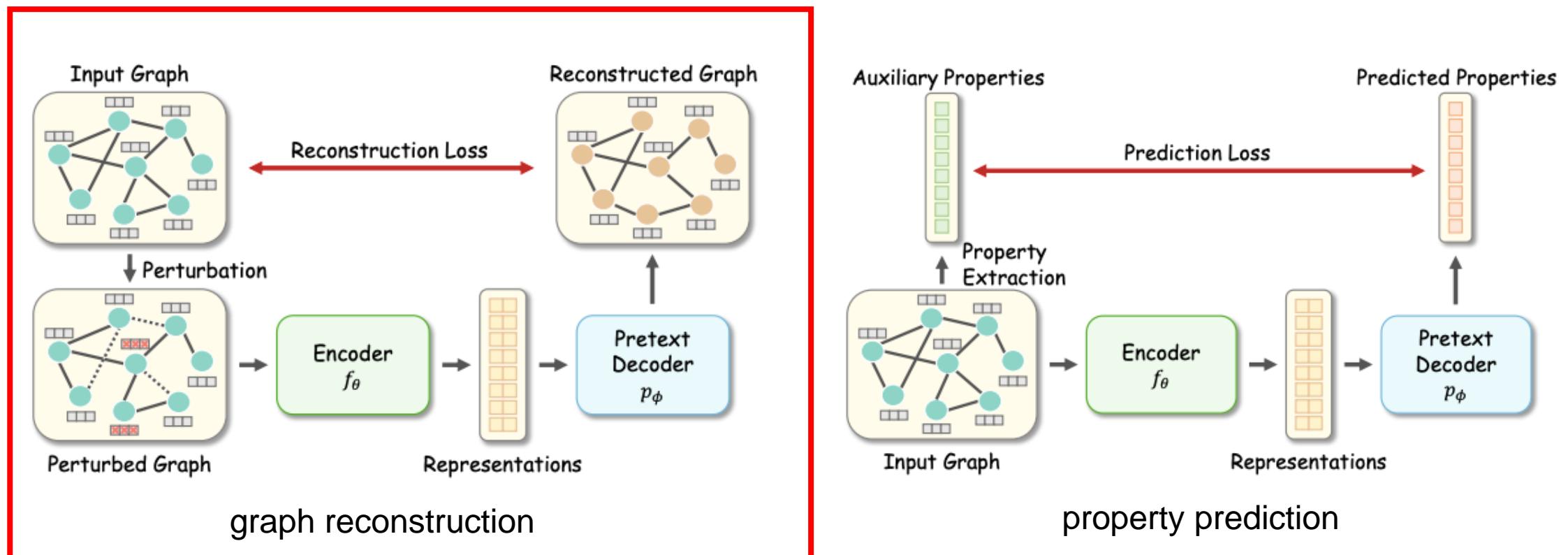
- Generative methods: graph reconstruction, property prediction



Pre-training

Pre-training

- Generative methods: graph reconstruction, property prediction



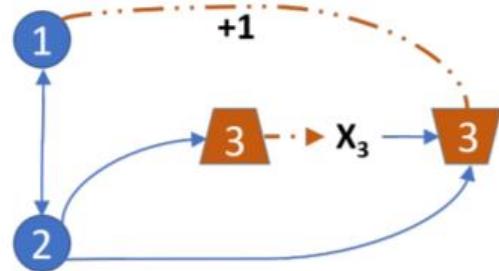
Motivations:

- **Scarcity of Labeled Data:** Adequate labeled data is often unavailable for training GNNs on specific tasks.
- **Proven Success of Pre-training:** Pre-training has significantly enhanced performance in domains like NLP and computer vision.
- **Need for Generalization:** Pre-training GNNs can help them generalize across various tasks with minimal customization.

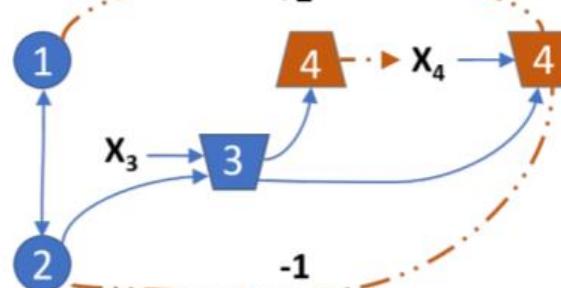
GPT-GNN

Pre-train large-scale graph with **reconstructing** the input graph. Decompose the reconstruction process into two coupled steps:

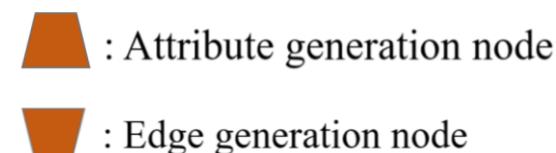
- Attribute generation: given observed edges, generate **node attributes**
- Edge generation: given observed edges and generated attributes, generate **masked edges**



(d) Generate attribute and
masked edges for node 3.



(e) Generate attribute and
masked edges for node 4.



Adaptive Graph Encoder

Motivations:

- Reconstructing the adjacency matrix = contrasting adjacent nodes
- Assumption: A node is similar to its neighbors.

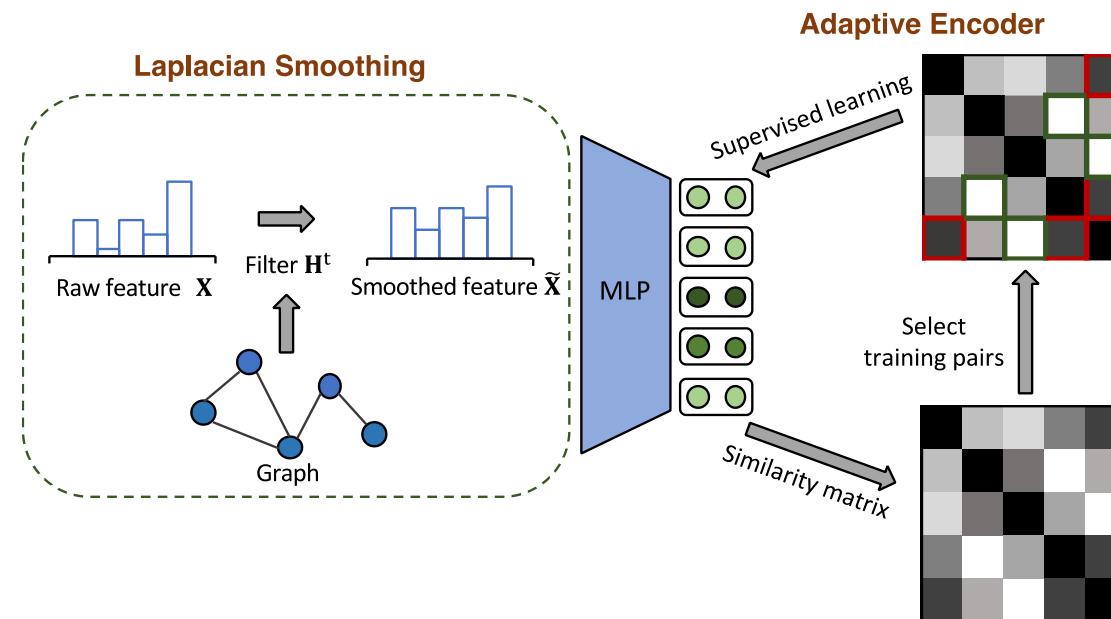


The three main drawbacks of GAE:

- **Entangled Architecture:** Combines multiple layers in a way that complicates training without improving performance.
- **Ineffective Filters:** The graph convolutional filters used are not optimal for filtering out high-frequency noise.
- **Unsuitable Objectives:** The training goals of reconstructing adjacency and feature matrices are not practical, as they can either overlook key data or retain unwanted noise.

Adaptive Graph Encoder

- **Laplacian Smoothing:** Design appropriate Laplacian smoothing filters to filter out high-frequency noise.
- **Adaptive Encoder:** Adaptively select training node pairs from the node similarity and adjust graph representations accordingly.



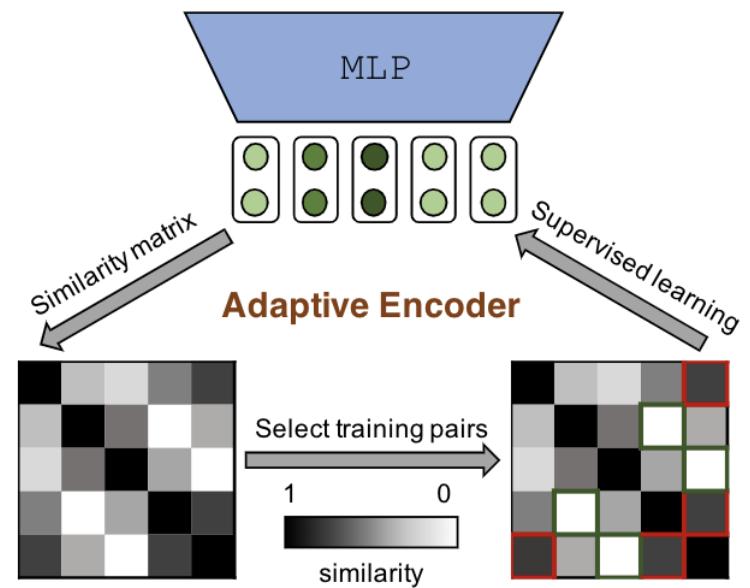
Adaptive Graph Encoder

- How to set training objectives to learn graph representations?
 - The adjacency matrix records only **one-hop** structural information.
 - Smoothed features or trained representations integrate both structural and feature information.
- **Adaptively select** training node pairs:

- High similarity pairs as positive examples.
- Low similarity pairs as negative examples.



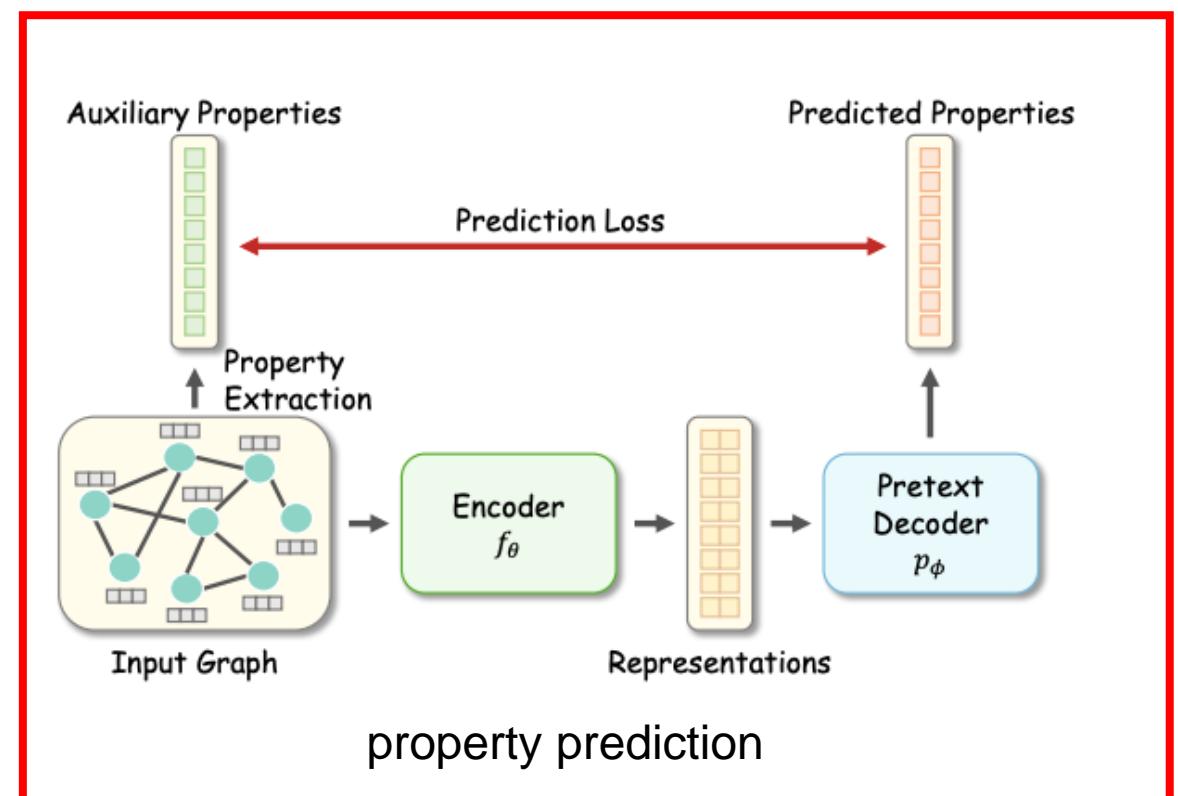
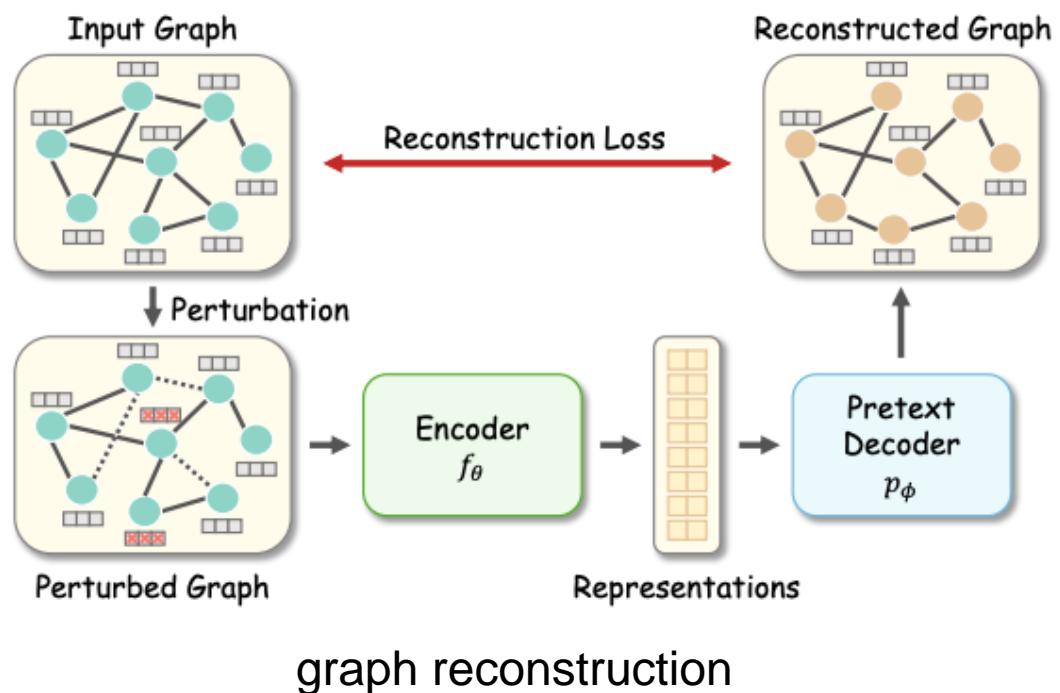
- Select Strategy
 - Calculate the cosine similarity matrix S .
 - Sort all node pairs and select those whose similarity is above/below a certain threshold.
 - Dynamically update the threshold.



Pre-training

Pre-training

- Generative methods: graph reconstruction, property prediction



GraphMAE

Motivations:

- **Lagging Development of GAEs:** GAEs lag behind contrastive methods in critical tasks like node and graph classifications, highlighting a need for enhanced models.
- **Challenges in Current GAE Approaches:** Existing GAEs struggle with issues like non-robust feature reconstruction and sensitivity to MSE, prompting the need for methodological improvements.
- **Decoder Limitations:** The simple MLP decoders commonly used in GAEs are inadequate for complex graph data, suggesting a need for more expressive architectures.

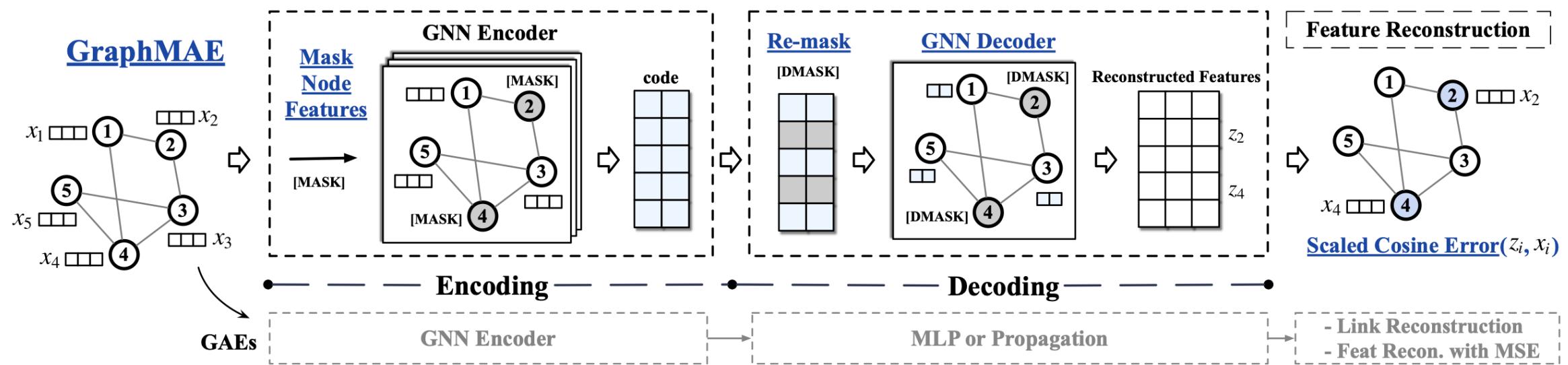
GraphMAE

Masked Feature Reconstruction: Focuses on node feature reconstruction with masking, proven effective in enhancing performance.

Scaled Cosine Error: Uses a scaled cosine error for better handling of feature magnitude variations and sample difficulty imbalances.

Re-mask Decoding: Employs re-masking of encoded node embeddings to improve decoding accuracy.

Advanced Decoder Architecture: Incorporates complex GNNs in the decoder for improved expressiveness.



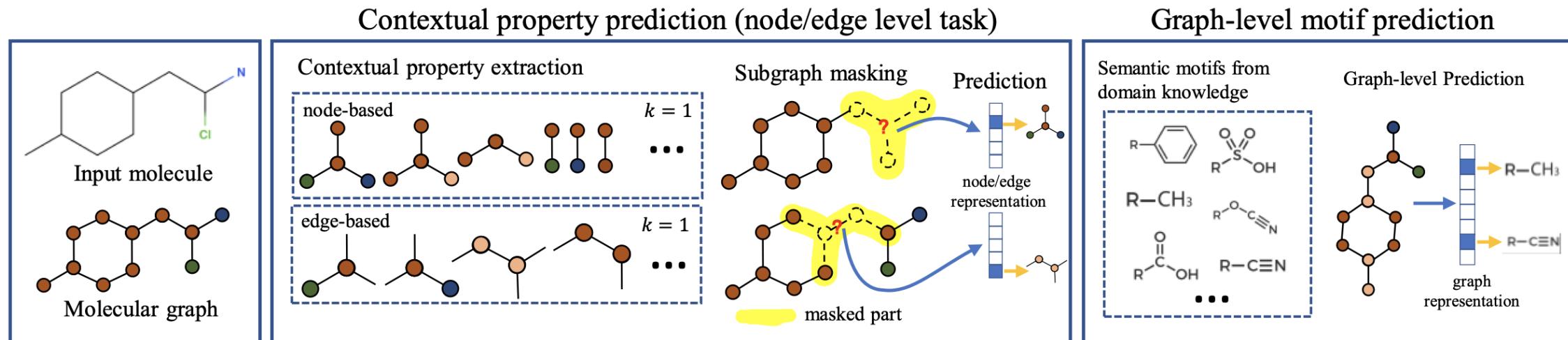
Motivations:

- **Scarcity of Labeled Data:** There is a significant lack of labeled molecular data, making it challenging to apply traditional supervised learning effectively in drug discovery.
- **Limitations of Current Methods:** Existing molecular representation methods, like SMILES, fail to adequately capture the complex topological information of molecules.
- **Need for Novel Strategies:** There is a pressing need for novel computational strategies that can efficiently exploit vast amounts of unlabeled molecular data to improve prediction accuracy and model generalization.

GROVER

Designed self-supervised tasks in node-, edge- and graph-level, learn rich **structural** and **semantic** information of molecules

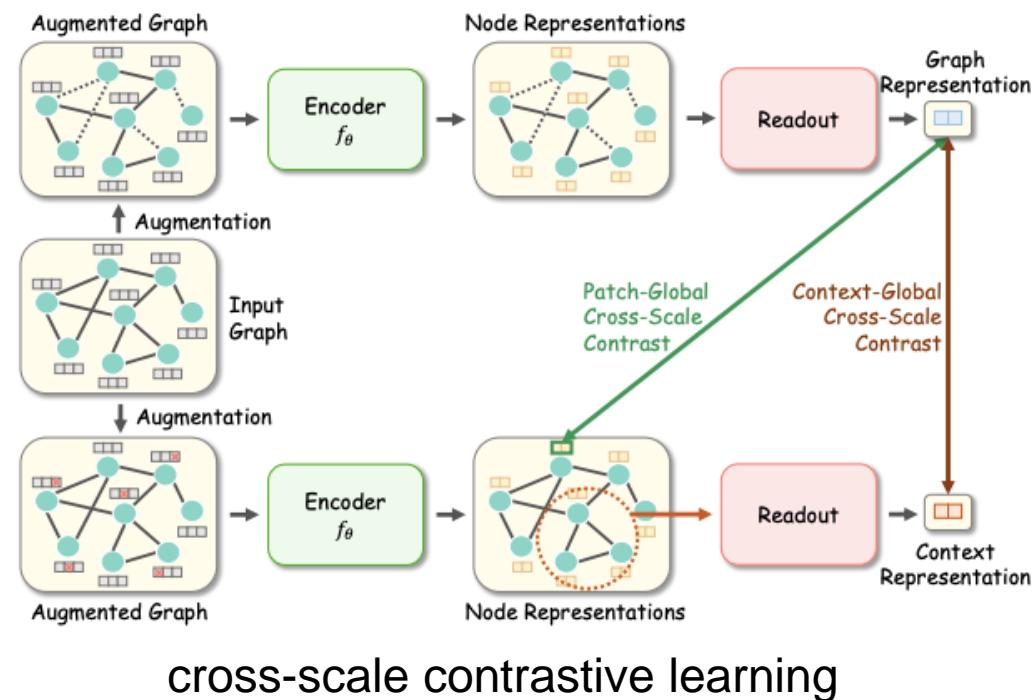
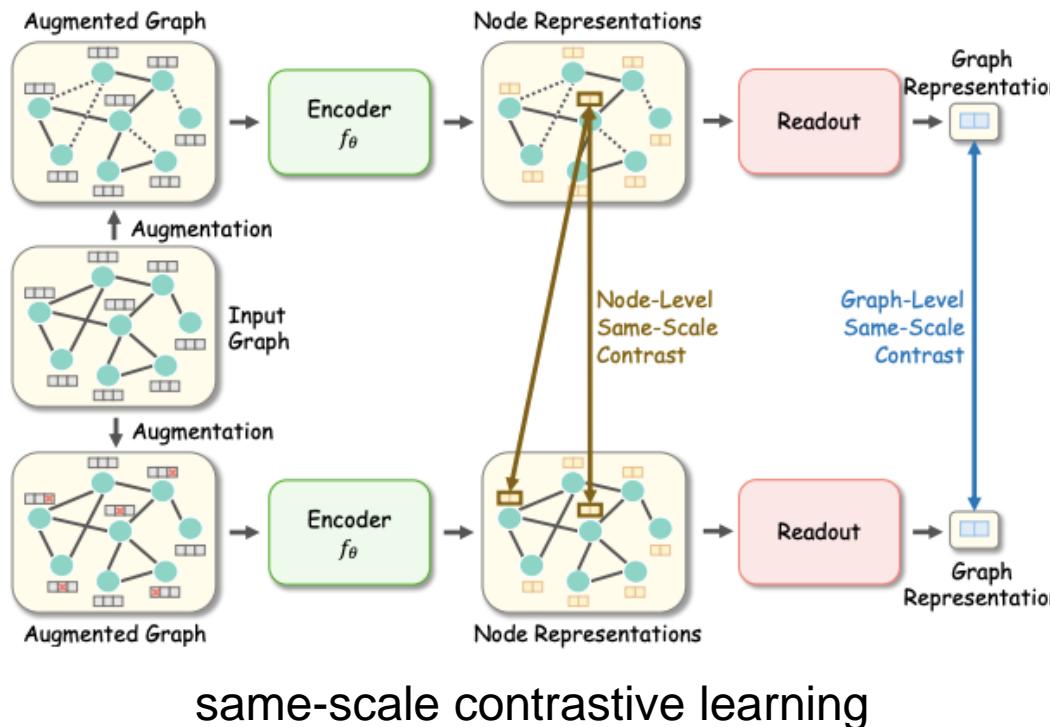
- Contextual property prediction: predict **masked node/edge** attributes set
- Motif prediction : predict the classes of **the motif** that occur in a given molecule



GNN-based Models

Pre-training

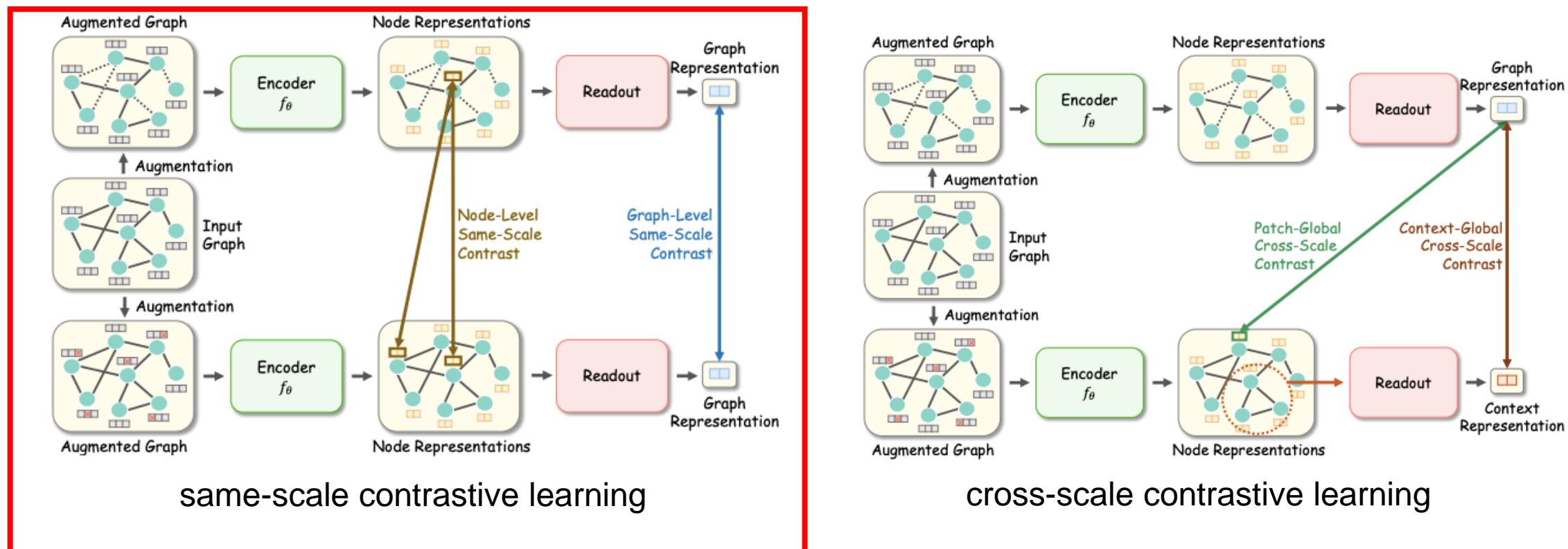
- Contrastive methods: same-scale contrastive learning, cross-scale contrastive learning



GNN-based Models

Pre-training

- Contrastive methods: same-scale contrastive learning, cross-scale contrastive learning



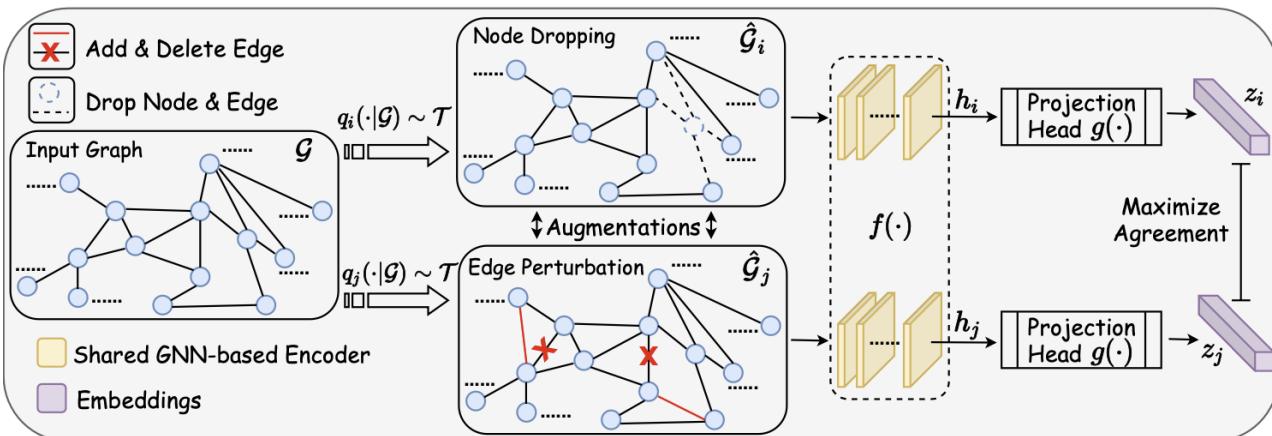
GraphCL

Motivations

- **Label Scarcity:** In fields like biology and chemistry, acquiring labels is costly and slow, making pre-training a valuable strategy to enhance GNNs, akin to its use in CNNs.
- **Complex Graph Data:** The diverse and complex nature of graph data makes designing effective pre-training schemes challenging, as simple methods like adjacency reconstruction may fall short.
- **Contrastive Learning Potential:** Contrastive learning could potentially overcome the limitations of proximity-focused pre-training by promoting feature consistency across different views.

GraphCL

Design four types of **graph augmentations** to incorporate various impacts in four different settings: semi-supervised, unsupervised, transfer learning and adversarial attacks.



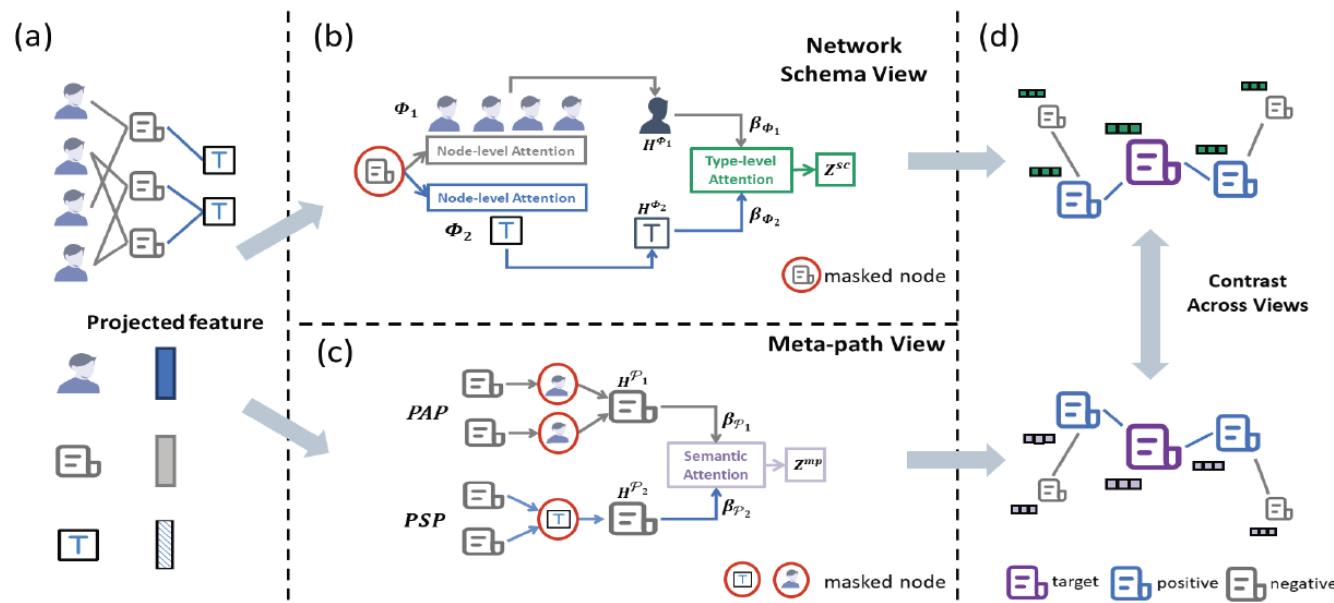
$$\ell_n = -\log \frac{\exp(\text{sim}(z_{n,i}, z_{n,j})/\tau)}{\sum_{n'=1, n' \neq n}^N \exp(\text{sim}(z_{n,i}, z_{n',j})/\tau)}$$

Contrast augmentation of same&different graphs

Data augmentation	Type	Underlying Prior
Node dropping	Nodes, edges	Vertex missing does not alter semantics.
Edge perturbation	Edges	Semantic robustness against connectivity variations.
Attribute masking	Nodes	Semantic robustness against losing partial attributes.
Subgraph	Nodes, edges	Local structure can hint the full semantics.

Core idea:

- Two views of a HIN (**network schema** and **meta-path views**) are proposed to capture both of local and high-order structures simultaneously.
- The **cross-view contrastive learning** is proposed to extract the positive and negative embeddings from two views.
- The two views to **collaboratively supervise each other** and finally learn high-level node embeddings.



Network Schema View Guided Encoder

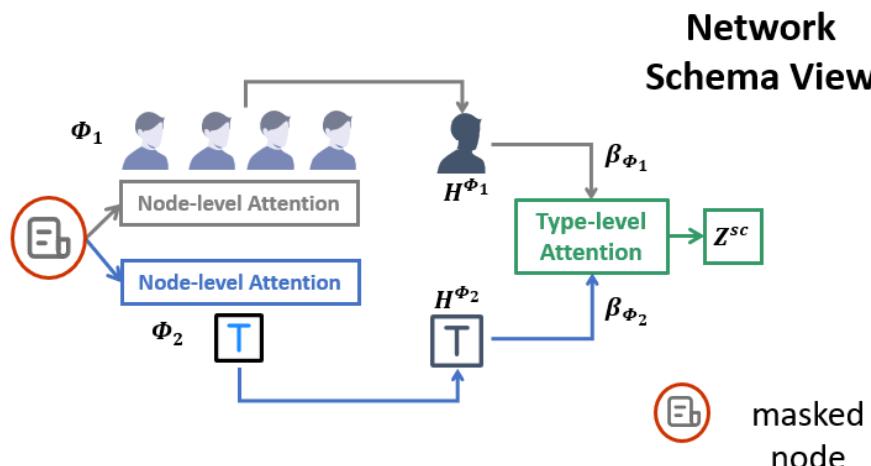
Node-level attention

For target paper

1. randomly sample Φ_m -type neighbors with threshold T_{Φ_m}
2. aggregate them with attention to get embedding of type Φ_m

Type-level aggregation

Aggregate different type of nodes with attention mechanism



Meta-path View Guided Encoder

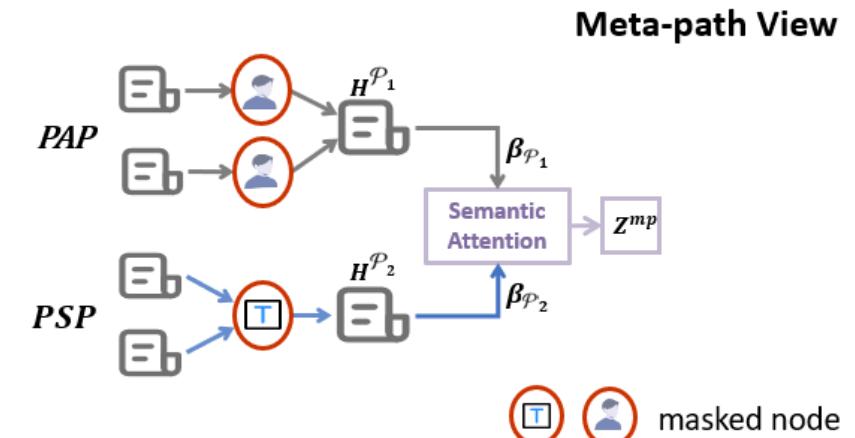
Meta-path specific GCN

For target paper

1. find its neighbors based on different meta-paths
2. aggregate them with meta-path based vanilla GCN

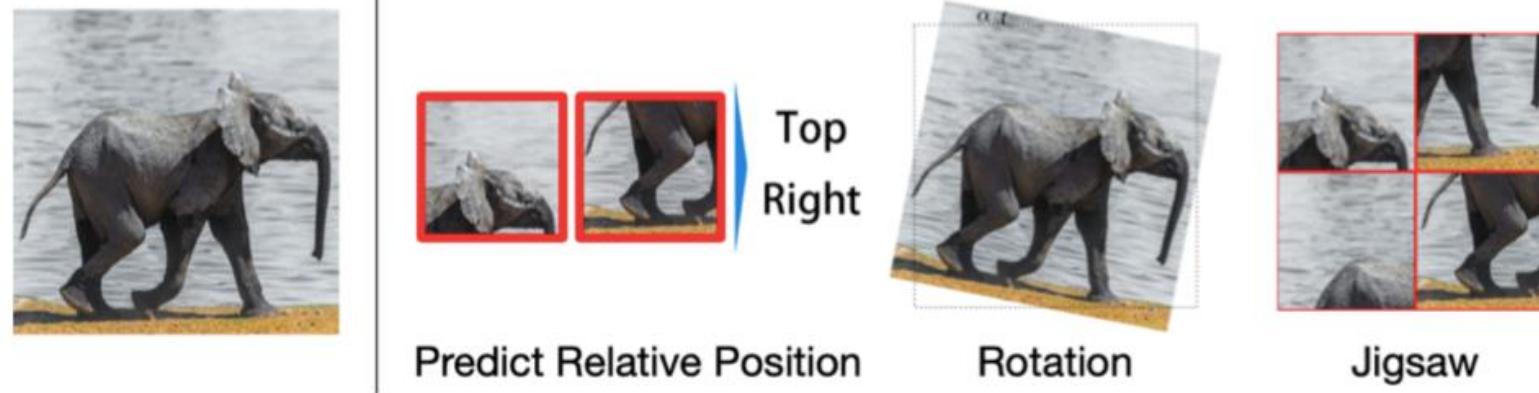
Semantic-level aggregation

Aggregate embedding under different meta-path with attention mechanism



Motivation

- Contrastive learning captures **invariant** information among different augmentation views.
- **Good augmentations** should introduce as much perturbation as possible without changing the core semantics.



- However, in graph contrastive learning (GCL), we have few prior knowledge on how to generate such good augmentations.

Can we generate better augmentations than typical random dropping-based methods?

MA-GCL

Core idea

- We interpret a GNN as a sequence of propagation operator g and transformation operator h :
 - propagation operator g is typically the non-parametric graph filter.
 - transformation operator h is typically a weight matrix with a non-linear function.

$$g(\mathbf{Z}; \mathbf{F}) = \mathbf{F}\mathbf{Z}, \quad h(\mathbf{Z}; \mathbf{W}) = \sigma(\mathbf{Z}\mathbf{W}), \quad \mathbf{F} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}},$$

$$GCN(\mathbf{X}) = h_L \circ g \circ h_{L-1} \circ g \circ \cdots \circ h_1 \circ g(\mathbf{X}),$$

$$SGC(\mathbf{X}) = h \circ g^{[L]}(\mathbf{X}),$$

- Intuition: different architectures (i.e., operator sequences) **won't** affect the core semantics.
- Thus we **perturb the neural architecture of graph encoder** as model augmentations.

We propose three strategies to introduce perturbations:

- Asymmetric strategy
 - Use the same number of operator h with shared parameters for different views
 - Use **different numbers of operator g for different views**
- Random strategy
 - **Randomly vary the number** of propagation operator g **in every training epoch**
- Shuffling strategy
 - **Randomly shuffle the permutation** of propagation and transformation operators

MA-GCL

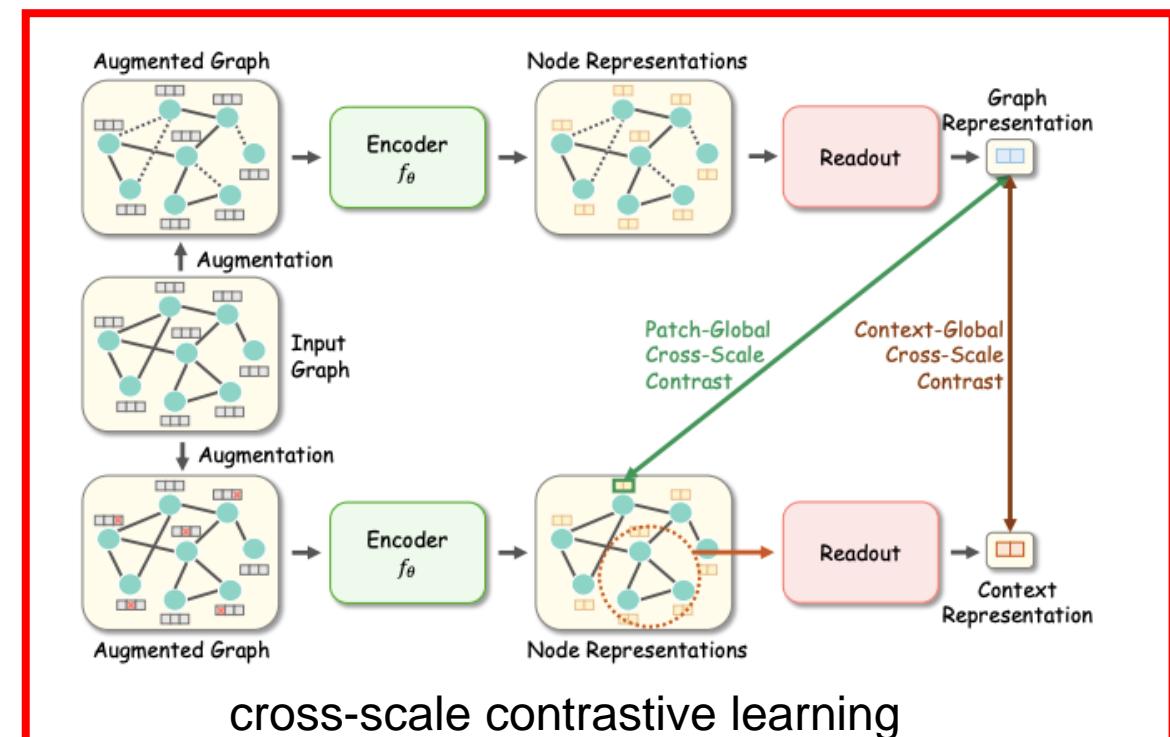
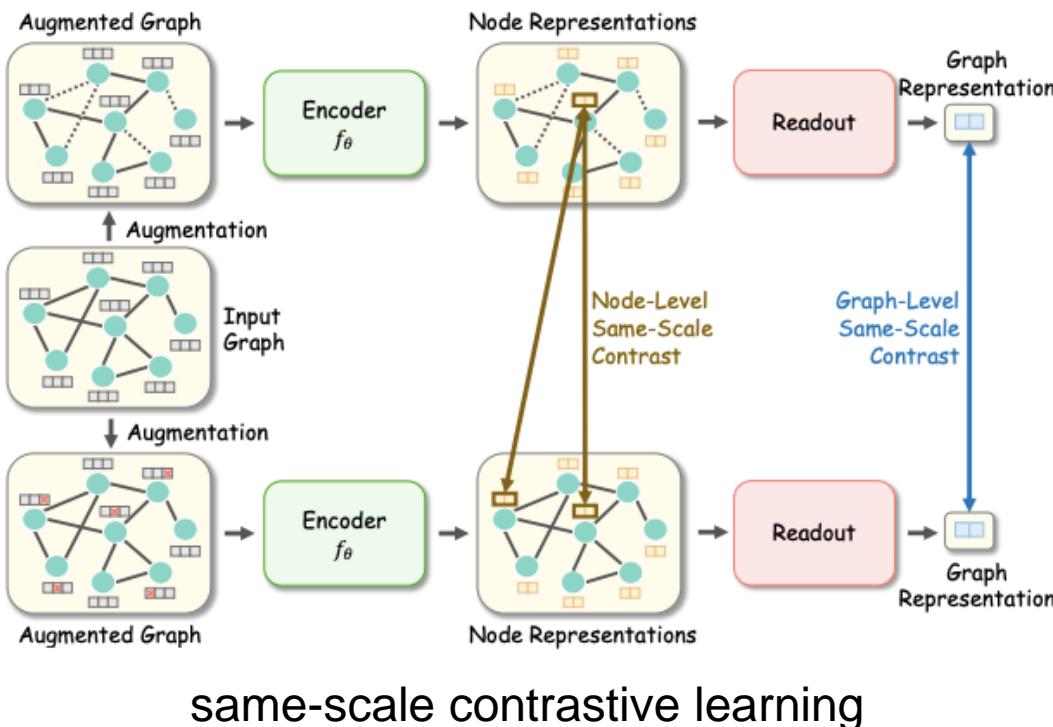
We conducted extensive experiments on node/graph classification/clustering.

Datasets	Cora	CiteSeer	PubMed	Coauthor-CS	Amazon-C	Amazon-P	Avg. Acc.	Avg. Rank
GCN	82.5 ± 0.4	71.2 ± 0.3	79.2 ± 0.3	93.03 ± 0.3	86.51 ± 0.5	92.42 ± 0.2	-	-
GAT	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.3	92.31 ± 0.2	86.93 ± 0.3	92.56 ± 0.4	-	-
InfoGCL	83.5 ± 0.3	73.5 ± 0.4	79.1 ± 0.2	-	-	-	-	-
DGI	82.3 ± 0.6	71.8 ± 0.7	76.8 ± 0.3	92.15 ± 0.6	83.95 ± 0.5	91.61 ± 0.2	83.10	8.5
GRACE	81.7 ± 0.4	71.5 ± 0.5	80.7 ± 0.4	92.93 ± 0.0	87.46 ± 0.2	92.15 ± 0.2	84.44	6.5
MVGRL	83.4 ± 0.3	73.0 ± 0.3	80.1 ± 0.6	92.11 ± 0.1	87.52 ± 0.1	91.74 ± 0.0	84.63	6.5
BGRL	81.7 ± 0.5	72.1 ± 0.5	80.2 ± 0.4	93.01 ± 0.2	88.23 ± 0.3	92.57 ± 0.3	84.63	6.5
GCA	83.4 ± 0.3	72.3 ± 0.1	80.2 ± 0.4	93.10 ± 0.0	87.85 ± 0.3	92.53 ± 0.2	84.89	4.0
SimGRACE	77.3 ± 0.1	71.4 ± 0.1	78.3 ± 0.3	93.45 ± 0.4	86.04 ± 0.2	91.39 ± 0.4	82.98	8.5
COLES	81.2 ± 0.4	71.5 ± 0.2	80.4 ± 0.7	92.65 ± 0.1	79.64 ± 0.0	89.00 ± 0.5	82.40	8.8
ARIEL	82.5 ± 0.1	72.2 ± 0.2	80.5 ± 0.3	93.35 ± 0.0	88.27 ± 0.2	91.43 ± 0.2	84.71	4.8
CCA-SSG	83.9 ± 0.4	<u>73.1 ± 0.3</u>	<u>81.3 ± 0.4</u>	93.37 ± 0.2	<u>88.42 ± 0.3</u>	92.44 ± 0.1	<u>85.42</u>	<u>2.3</u>
Base Model	81.1 ± 0.4	71.4 ± 0.1	79.1 ± 0.4	92.86 ± 0.3	87.65 ± 0.2	91.19 ± 0.3	83.88	9.0
MA-GCL	83.3 ± 0.4	73.6 ± 0.1	83.5 ± 0.4	94.19 ± 0.1	88.83 ± 0.3	93.80 ± 0.1	86.20	1.2

GNN-based Models

Pre-training

- Contrastive methods: same-scale contrastive learning, cross-scale contrastive learning



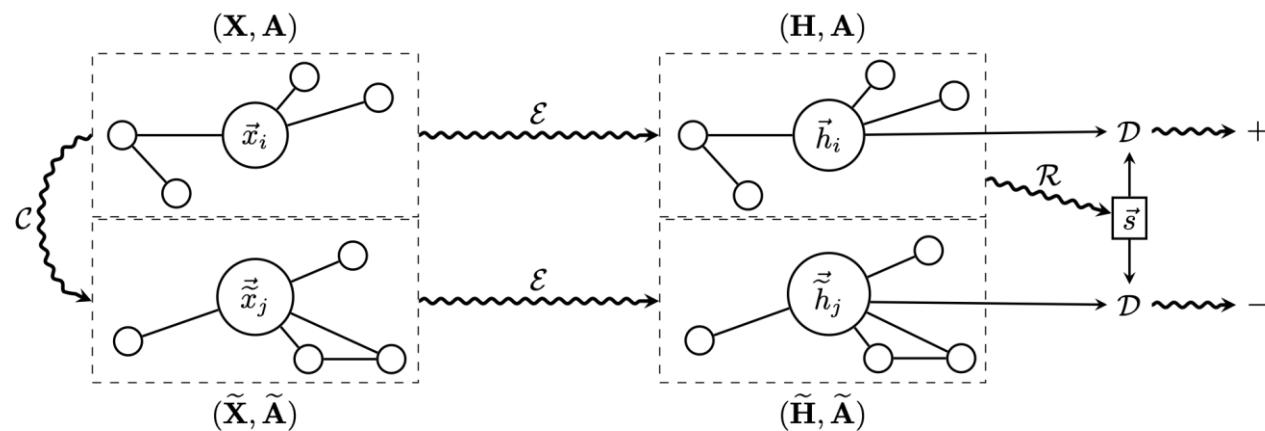
Deep Graph Infomax

Motivations:

- **Label Scarcity:** Most real-world graph data lacks labels, restricting the use of supervised methods.
- **Structure Discovery:** Unsupervised learning is vital for uncovering new structures in large-scale graphs.
- **Current Method Limitations:** Existing methods like random walks over-emphasize proximity and may neglect broader structural details.

Deep Graph Infomax

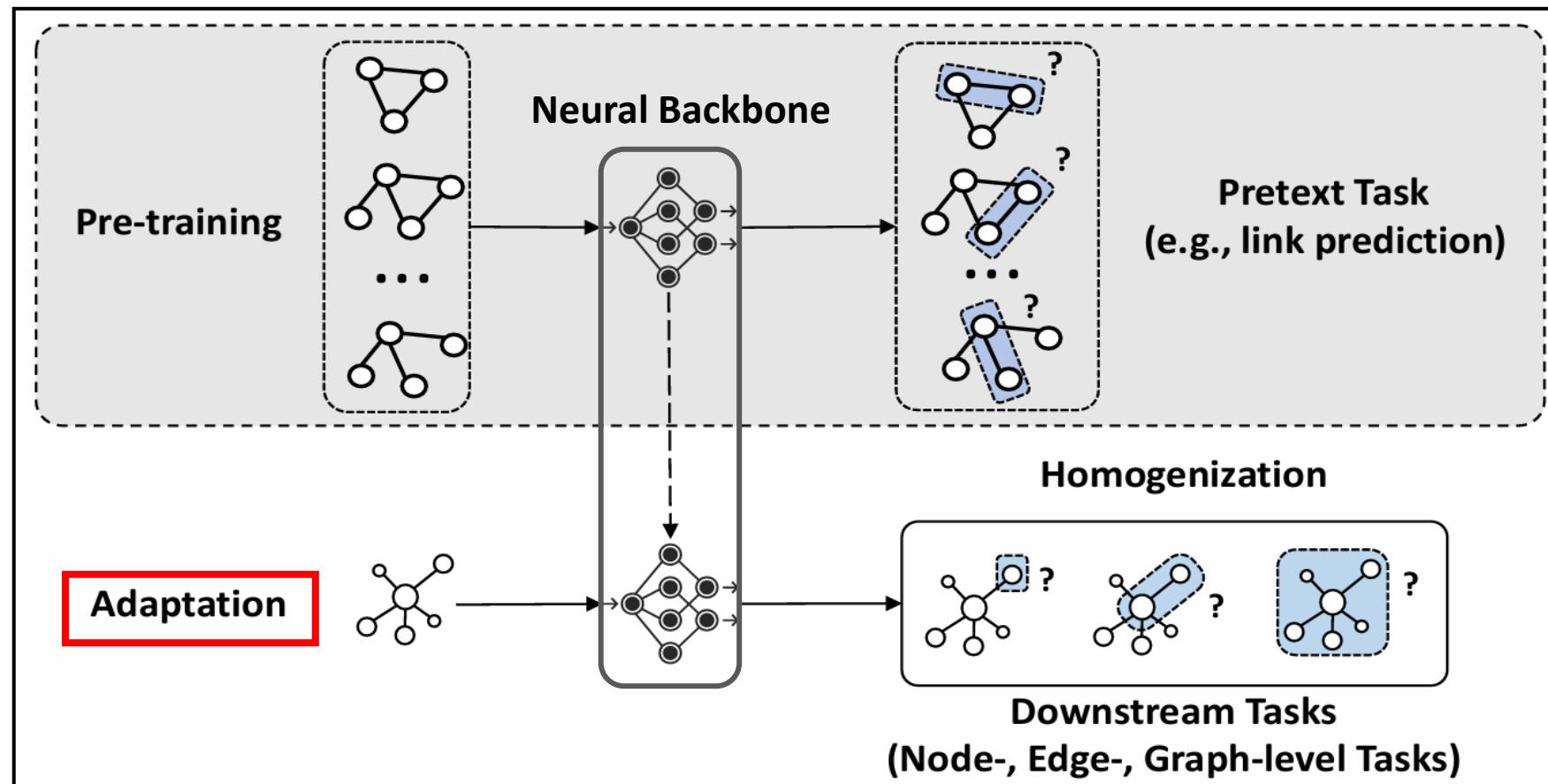
- **Node Representation:** GCN generates a representation for each node in the graph.
- **Graph Representation:** The global representation of the graph is produced by aggregating all node representations, typically through summation or averaging.
- **Negative Sampling:** Perturbed versions of the graph are generated, for example, by shuffling node features or edges to create negative samples.
- **Maximization of Mutual Information:** The network is trained by **maximizing the mutual information between node representations and the global representation** in the positive samples (original graph) and minimizing it in the negative samples (perturbed graph).



GNN-based Methods

Backbone: No unified architecture
(Message Passing/Graph Transformer)

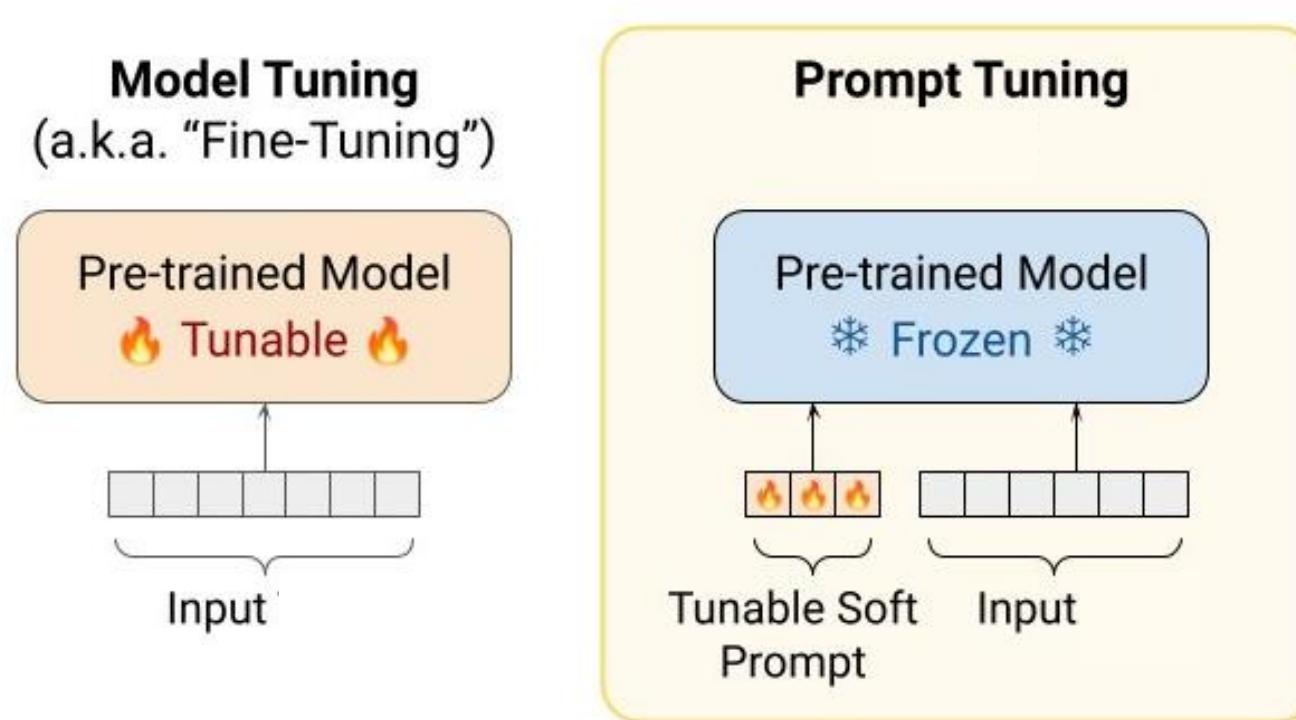
Paradigm: Pre-training + Adaptation



Adaptation

Downstream Adaptation

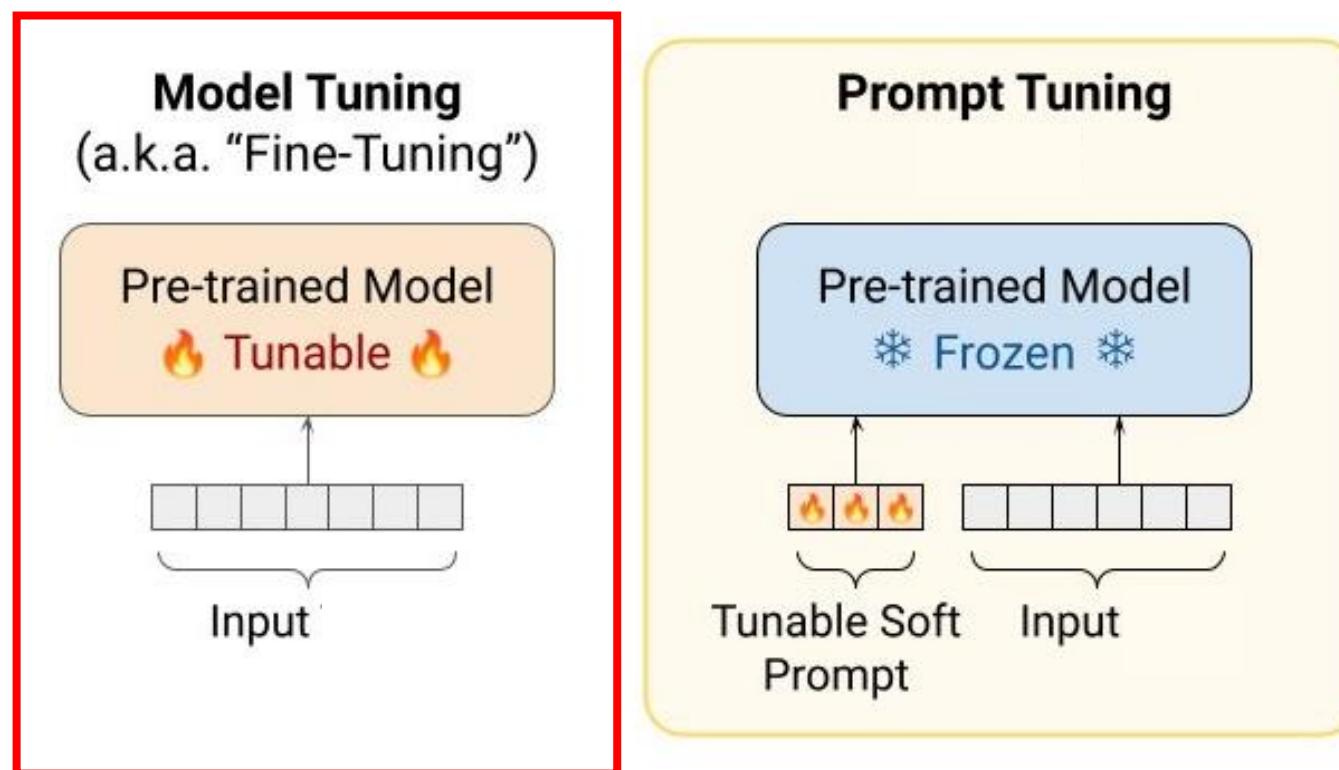
- Fine-tuning: keep input graph intact, **modify model parameters** accordingly
- Prompt-tuning: keep pre-trained model intact, **modify input graph or output embedding**



Adaptation

Downstream Adaptation

- Fine-tuning: keep input graph intact, **modify model parameters** accordingly
 - Parameter-efficient Fine-tuning (PEFT): only tune **a small portion** of parameters

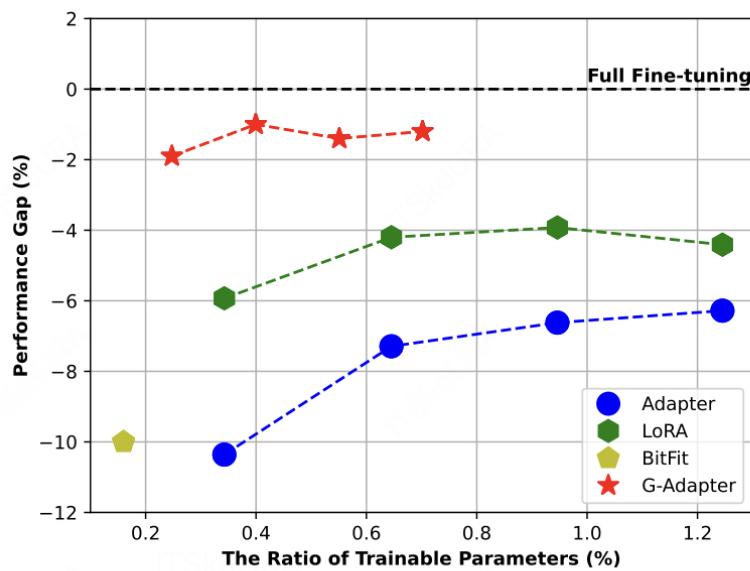


G-Adapter

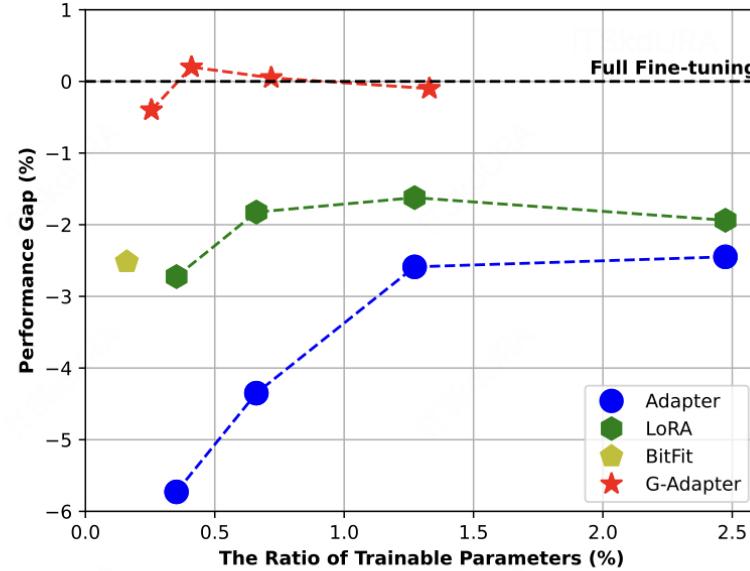
Can PEFTs from the language domain be transferred directly to graph-based tasks?

- There is a significant gap between traditional PEFTs and full fine-tuning, especially on large-scale datasets.

How to design a graph-specific PEFT method?



(a) On large-scale datasets.

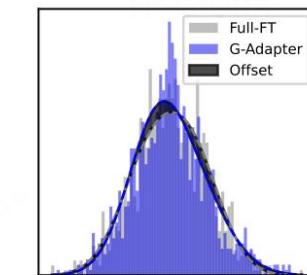
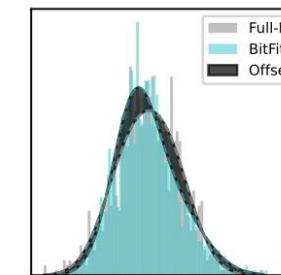
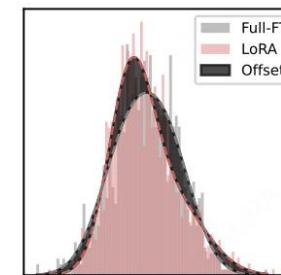
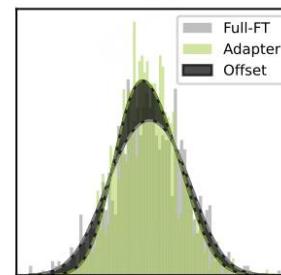
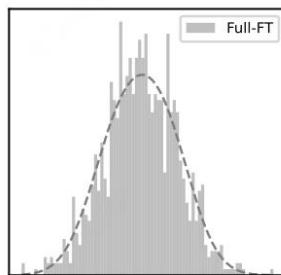


(b) On small-scale datasets.

G-Adapter

Method

- Exploration in this paper reveals the feature distribution shift issue due to the absence of graph structure in the fine-tuning process.
- To alleviate these concerns, a novel structure-aware PEFT method, G-Adapter, is proposed, which leverages graph convolution operation to introduce graph structure as the inductive bias to guide the updating process.
- They apply the low-rank decomposition to the learnable weights, which makes G-Adapter highly lightweight.



Adapter: 11.12%

LoRA: 10.01%

BitFit: 8.22%

G-Adapter: 1.98%

AdapterGNN

Motivation

- Delta tuning improves the traditional fine-tuning in the catastrophic forgetting of pre-trained knowledge problem and overfitting problem.

How to effectively utilize the advantages of delta tuning while preserving the expressivity of GNNs?

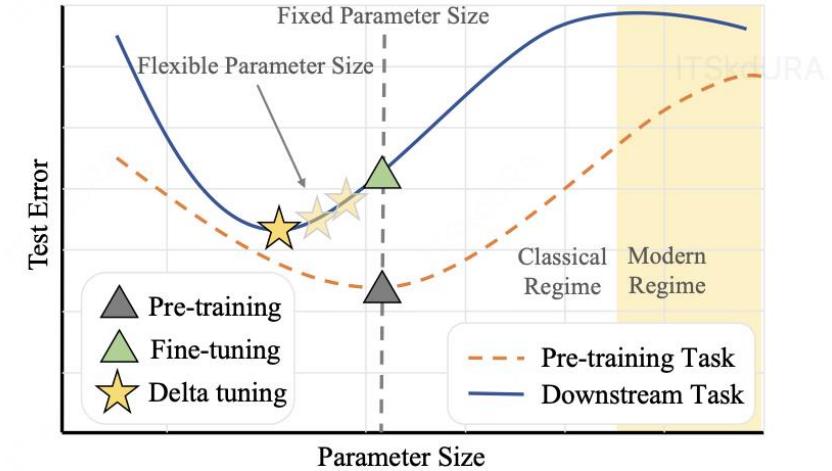
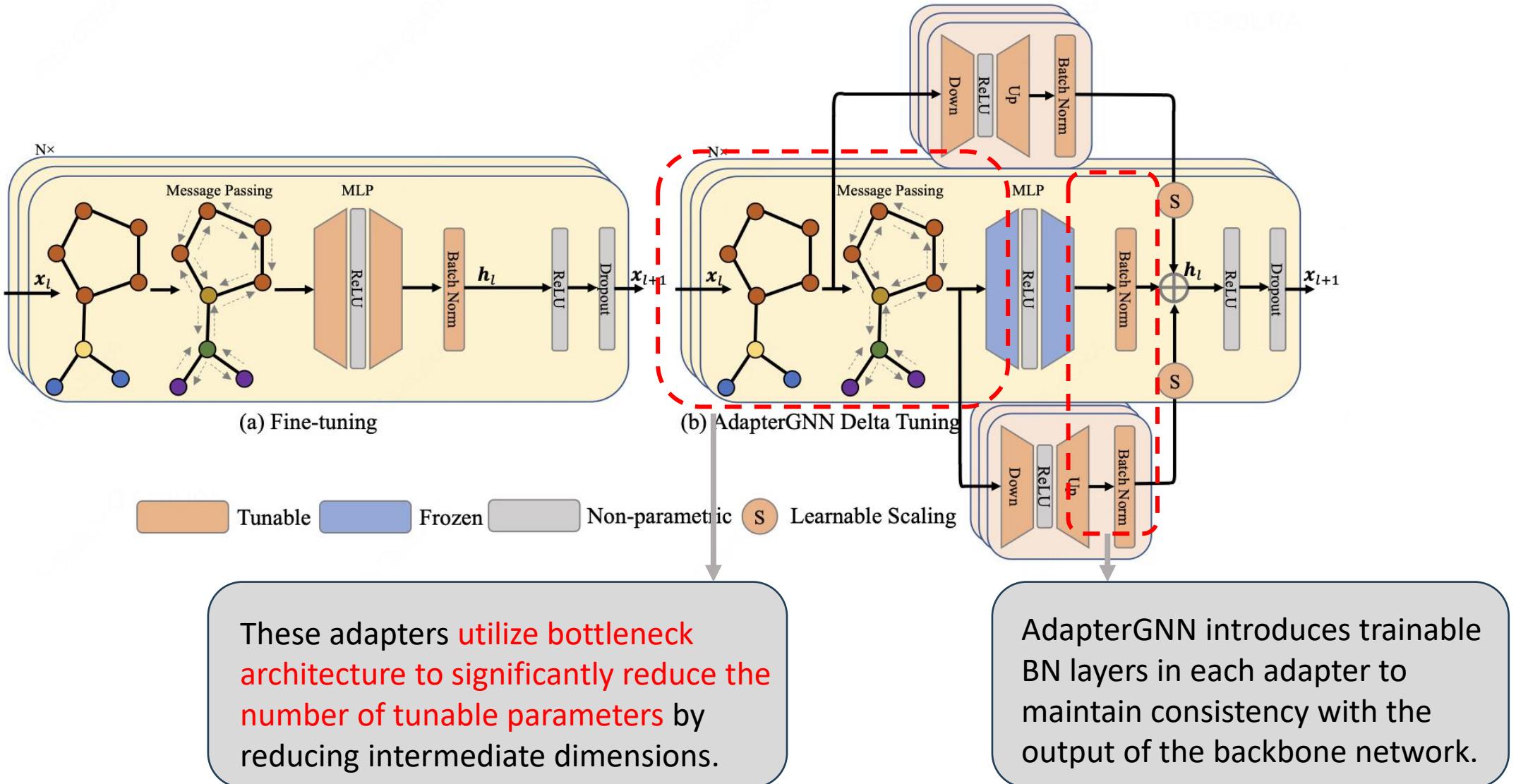


Figure 1: A large model is often employed for pre-training \blacktriangle when sufficient data is available. However, for downstream tasks with limited data, a smaller model is optimal in the classical regime. Compared with fine-tuning \blacktriangle , delta tuning \star preserves expressivity while reducing the size of parameter space, leading to lower test error.

AdapterGNN

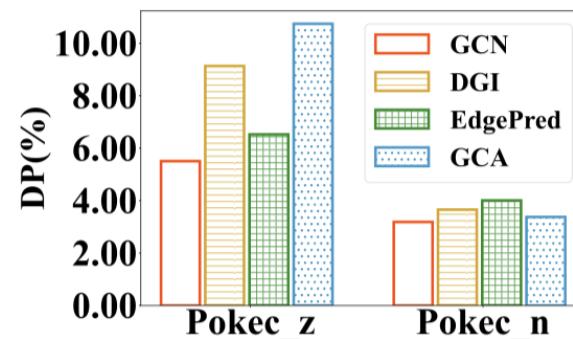


GraphPAR

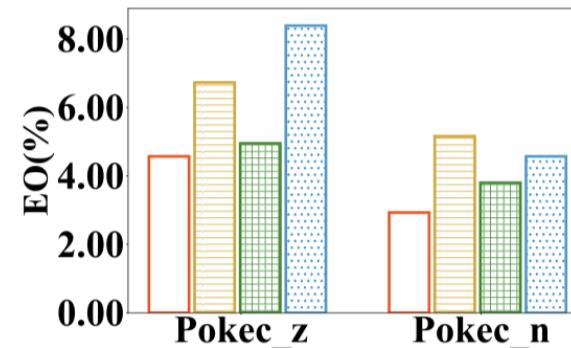
Background

- Recent works have demonstrated that pre-trained language models tend to inherit bias from pre-training corpora.
- Pre-trained Graph Models(PGMs) can well capture semantic information on graphs during the pre-training phase, which inevitably contains sensitive attribute semantics.

How to improve the fairness of PGMs?



(a) Demographic Parity (DP).



(b) Equality Opportunity (EO).

GraphPAR

Existing fair methods is inflexible and inefficient.

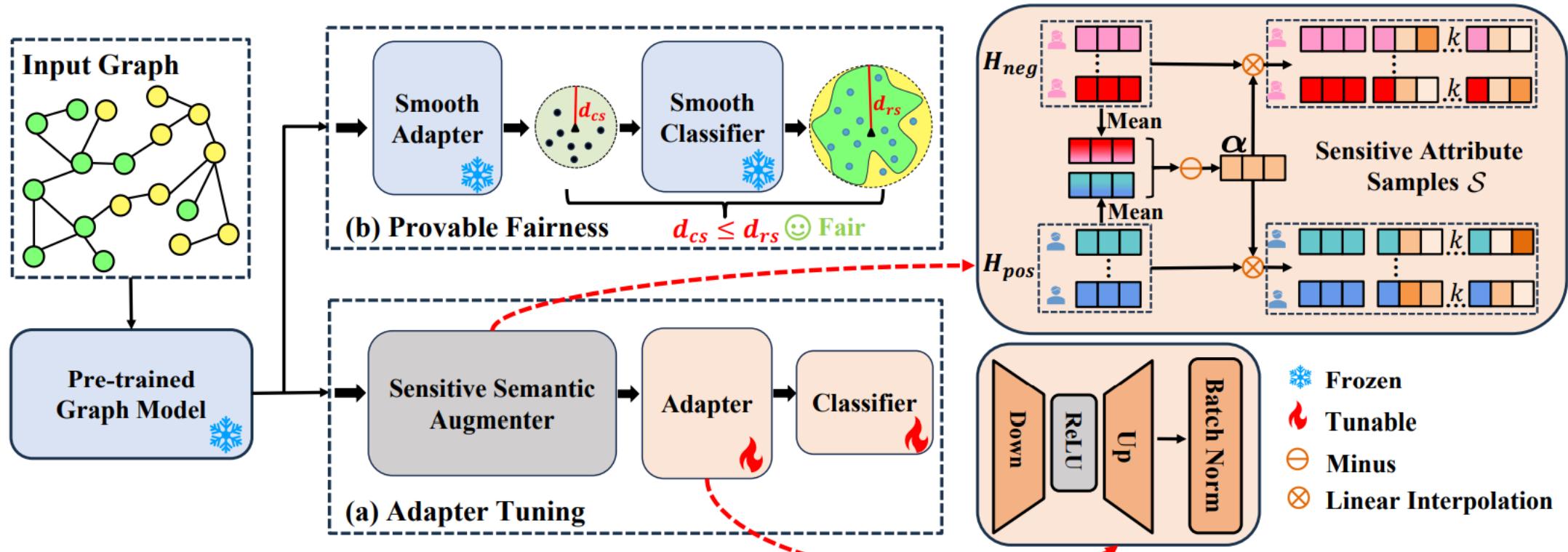
- Existing works generally train a fair GNN for a specific task.
- Debiasing for a specific task in the pre-training phase is inflexible, and maintaining a specific PGM for each task is inefficient.

Existing fair methods lack theoretical guarantees.

- Provable lower bounds on the fairness of model prediction.

How to efficiently and flexibly endow PGMs fairness with practical guarantee?

GraphPAR



Training adapter for PGMs fairness

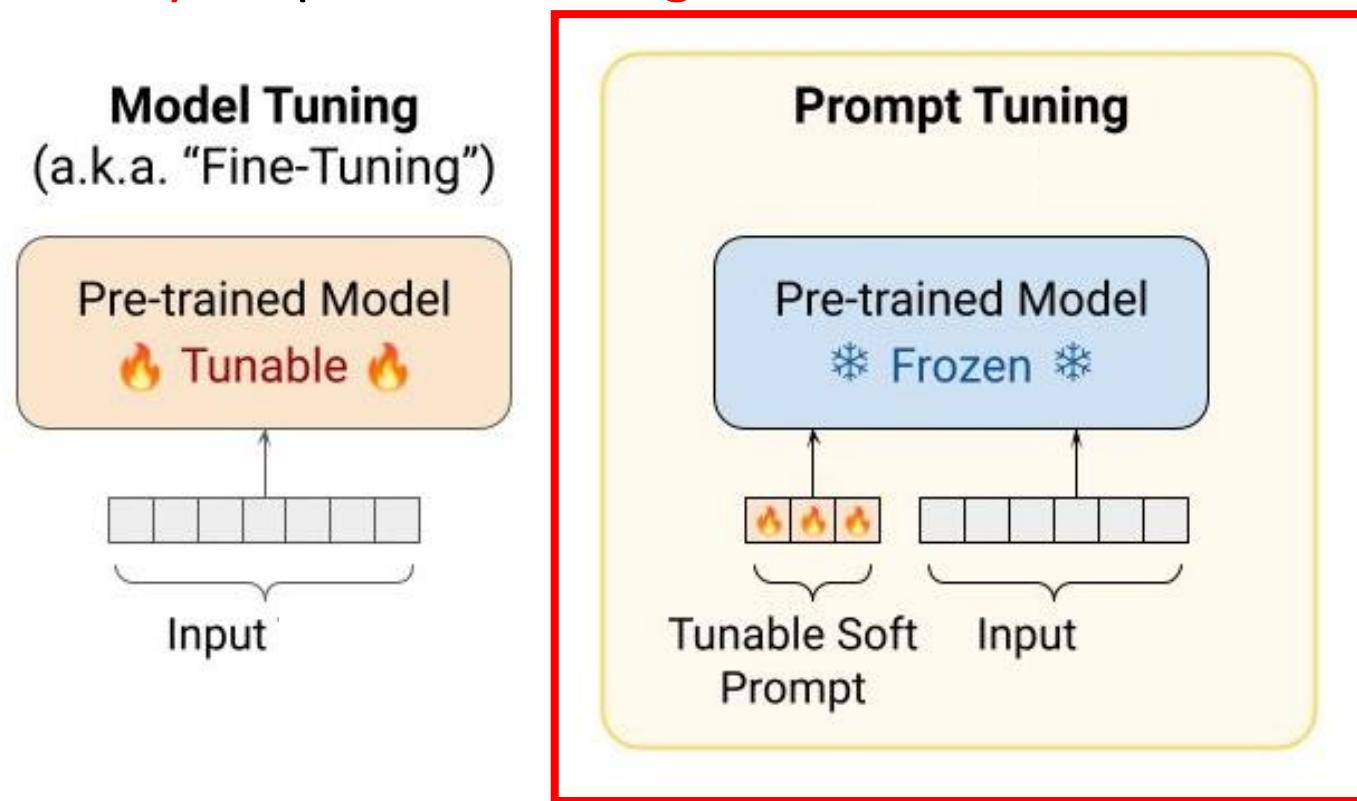
$$\mathcal{L}_{\text{RandAT}} = \mathbb{E}_{i \in \mathcal{V}_L} \left[\mathbb{E}_{\mathbf{h}'_i \in \hat{\mathcal{S}}_i} [\ell(d \circ g(\mathbf{h}'_i), y_i)] \right],$$

$$\mathcal{L}_{\text{MinMax}} (\mathbf{h}_i) \approx \max_{\mathbf{h}'_i \in \hat{\mathcal{S}}_i} \|g(\mathbf{h}_i) - g(\mathbf{h}'_i)\|_2.$$

Adaptation

Downstream Adaptation

- Prompt-tuning: keep pre-trained model intact
 - pre-prompt: **modify** input **graph**
 - post-prompt: **modify** output **embedding**



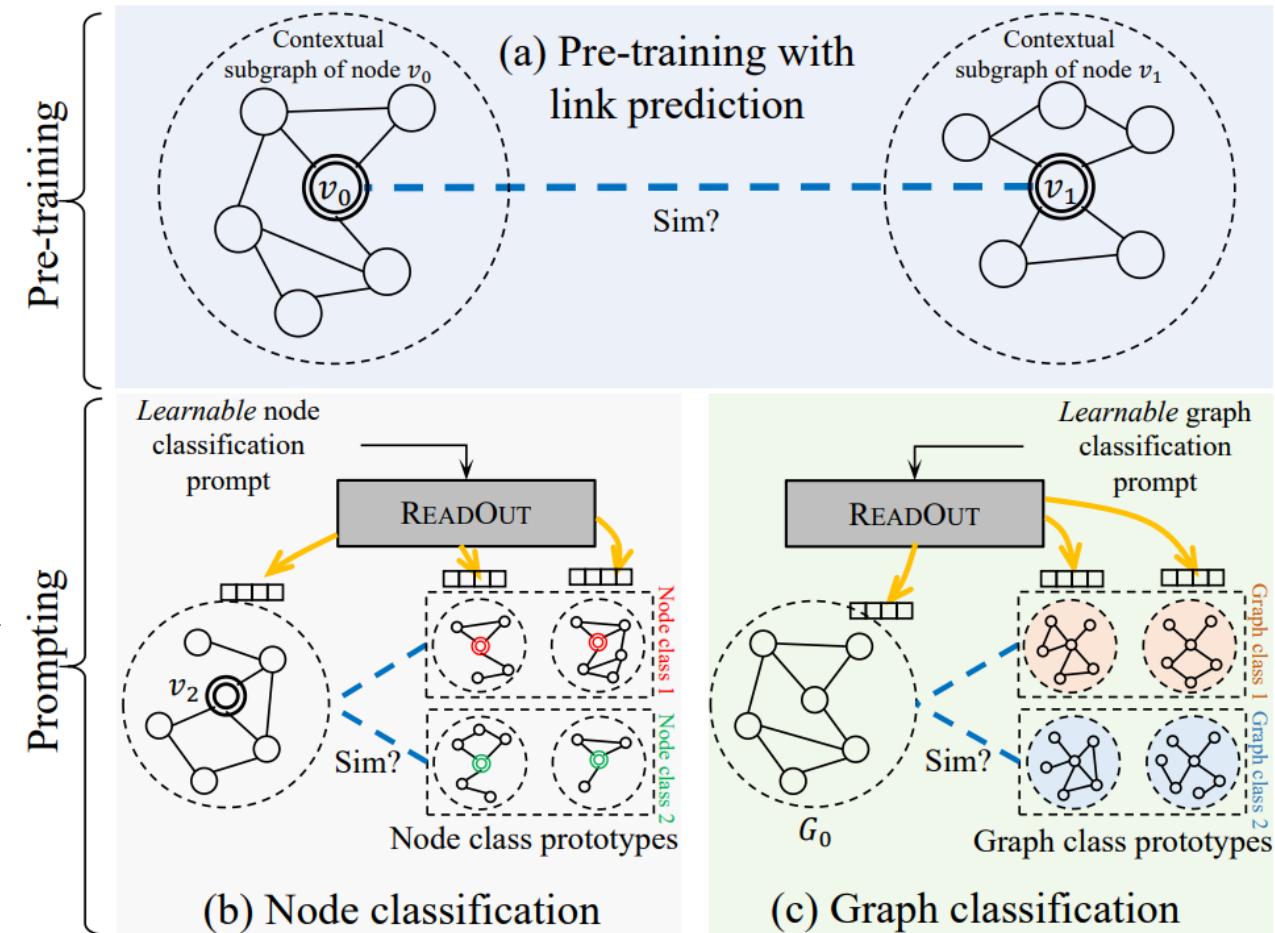
GraphPrompt

Problem:

- How to unify various pre-training and downstream tasks on graph?
- How to design prompts on graph?

Insights:

- A unified task template based on subgraph similarity computation
- Use a learnable prompt to guide graph readout for different tasks



GraphPrompt

Prompt design:

Different downstream tasks require different subgraph readout
→ Use task-specific learnable prompts

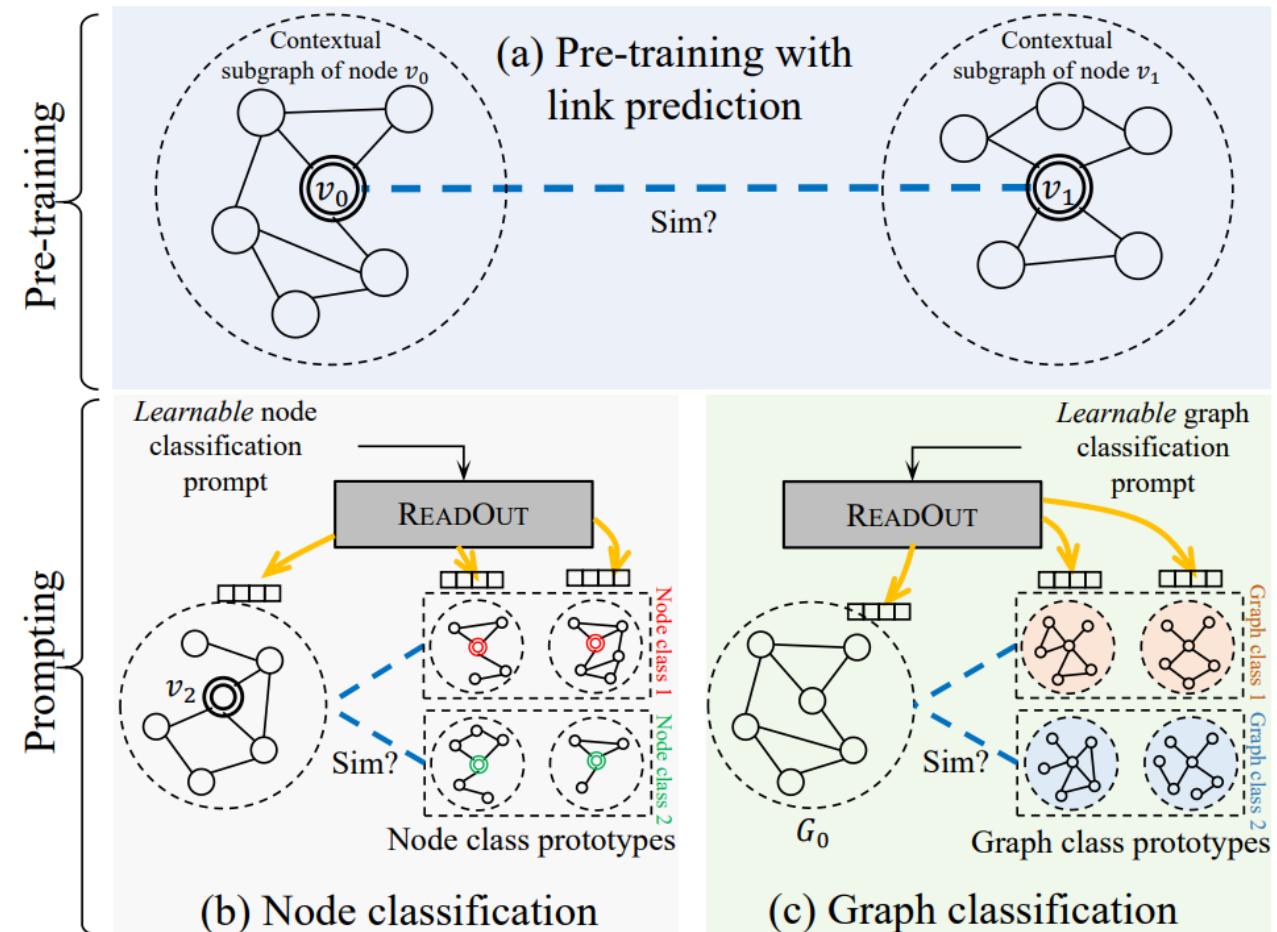
Prompt vector added to the readout layer of the pre-trained GNN

$$\mathbf{s}_{t,x} = \text{READOUT}(\{\mathbf{p}_t \odot \mathbf{h}_v : v \in V(S_x)\})$$

$\mathbf{s}_{t,x}$: (sub)graph embedding of x for a task t

\mathbf{h}_v : node v 's embedding vector

\mathbf{p}_t or \mathbf{P}_t : learnable prompt vector or matrix for task t

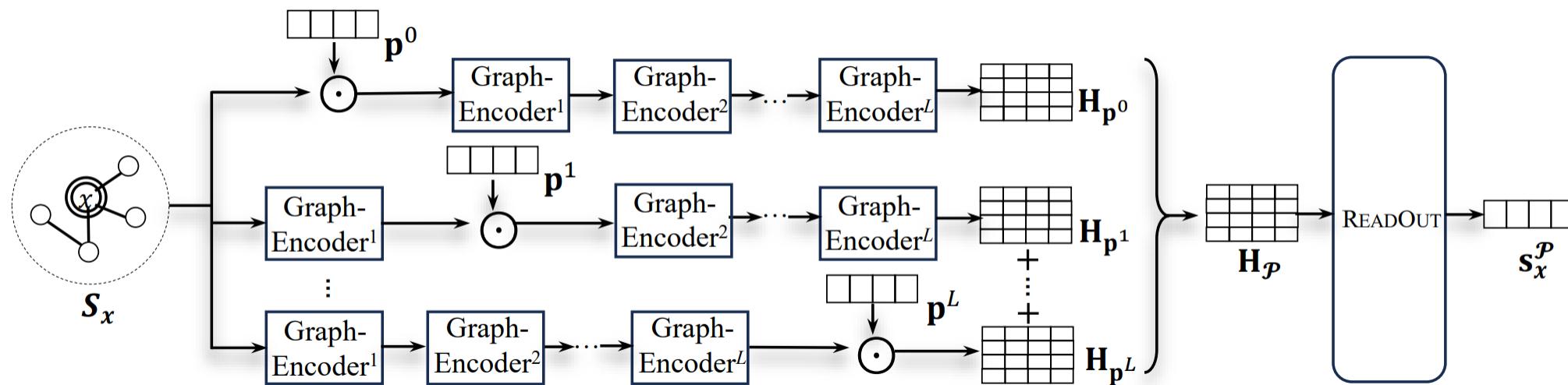


Generalized Graph Prompt

Support more pre-training tasks beyond link prediction:

- DGI, InfoGraph, GraphCL, GCC, ...

Layer-wise prompts



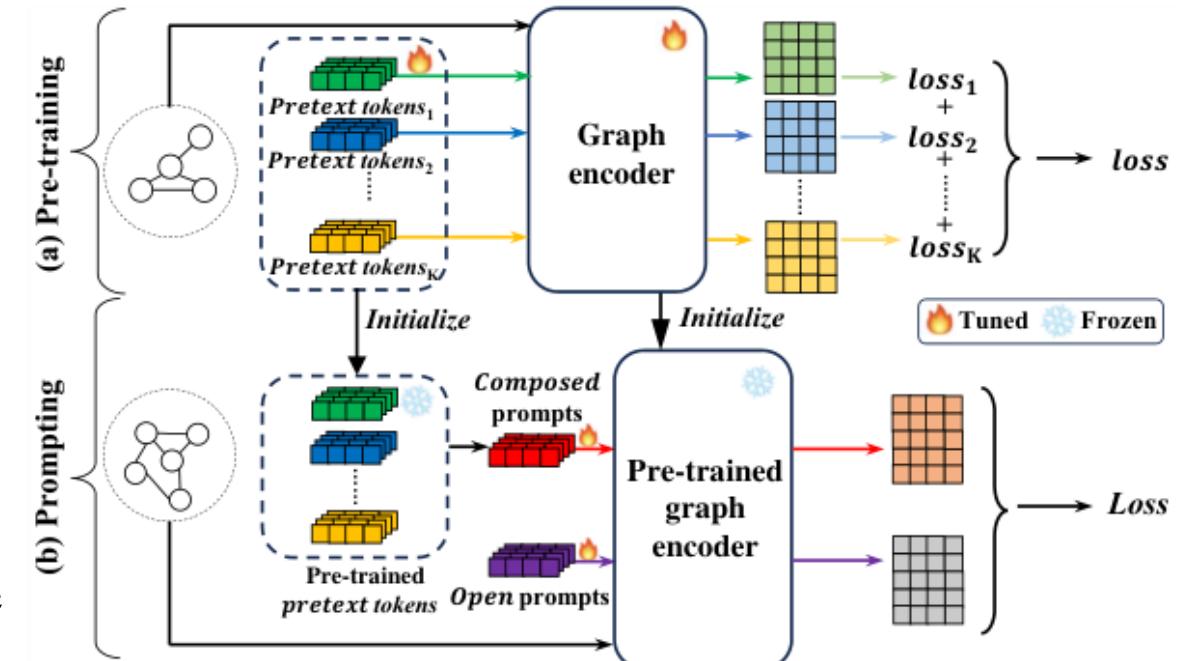
MultiGPrompt

Problem:

- To cater to diverse downstream tasks, pre-training should broadly extract knowledge from various aspects.

Challenges:

- Different pretext tasks often have different objectives, directly combining them lead to task interference.
- Multiple pretext tasks further complicates the alignment of downstream objectives with the pre-trained model.



C1: How can we leverage diverse pre-text tasks for graph models in a synergistic manner?

C2: How can we transfer both task-specific and global pre-trained knowledge

MultiGPrompt

Multitask pre-training

Pretext tokens

$$\mathcal{T}_{(k)} = \{t_{(k),0}, t_{(k),1}, \dots, t_{(k),L}\}$$

Add token to each layer of graph encoder

$$\mathbf{H}^{l+1} = \text{MP}(\mathbf{t}_{(k),l} \odot \mathbf{H}^l, \mathbf{A}; \theta^l)$$

Graph encoder output embedding

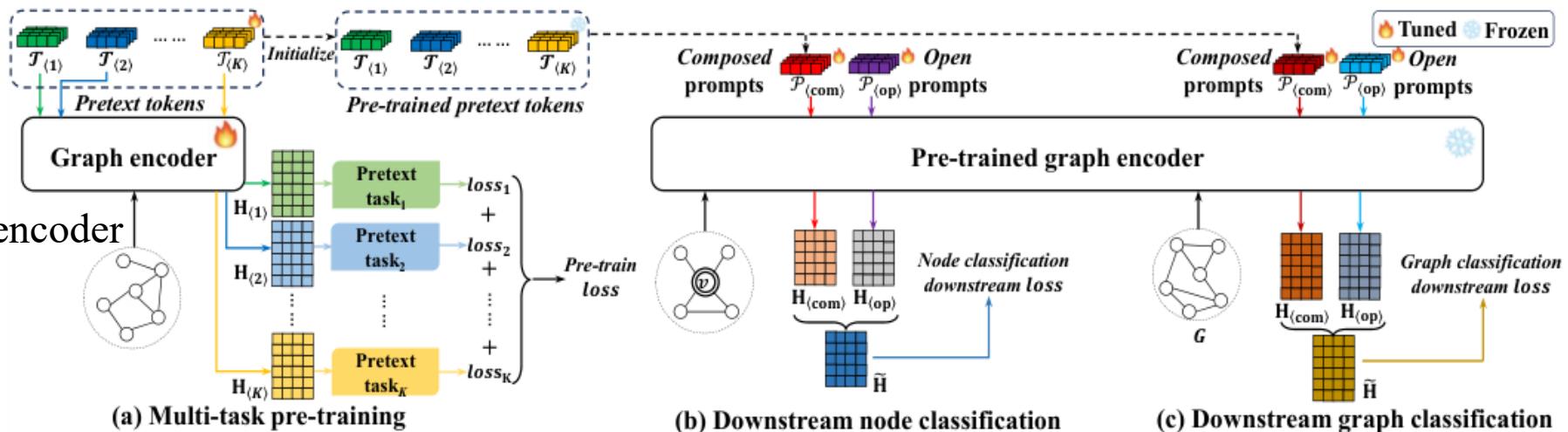
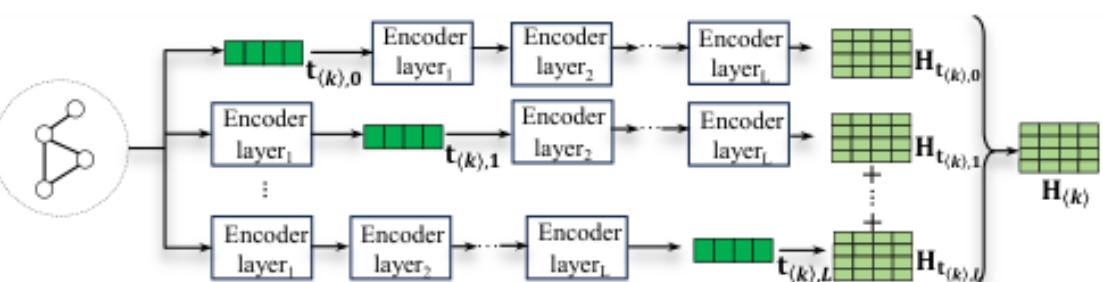
$$\mathbf{H}_t = \text{GRAPHENCODER}_t(\mathbf{X}, \mathbf{A}; \Theta)$$

Overall embedding

$$\mathbf{H}_{(k)} = \sum_{l=0}^L \alpha_l \mathbf{H}_{t_{(k),l}}$$

Pre-Training Objective

$$\mathcal{L}_{\text{pre}}(\mathcal{H}; \mathcal{T}, \Theta) = \sum_{k=1}^K \beta_k \mathcal{L}_{\text{pre}_{(k)}}(\mathbf{H}_{(k)}; \mathcal{T}_{(k)}, \Theta),$$



Prompt tuning

Composed prompt

$$\mathcal{P}_{\text{com}} = \{p_{\text{com},0}, p_{\text{com},1}, \dots, p_{\text{com},L}\}$$

$$p_{\text{com},l} = \text{COMPOSE}(t_{(1),l}, t_{(2),l}, \dots, t_{(K),l}; \Gamma)$$

Open prompt

$$\mathcal{P}_{\text{op}} = \{p_{\text{op},0}, p_{\text{op},1}, \dots, p_{\text{op},L}\}$$

Add prompt to each layer of graph encoder

$$\mathbf{H}_p = \text{GRAPHENCODER}_p(\mathbf{X}, \mathbf{A}; \Theta_{\text{pre}})$$

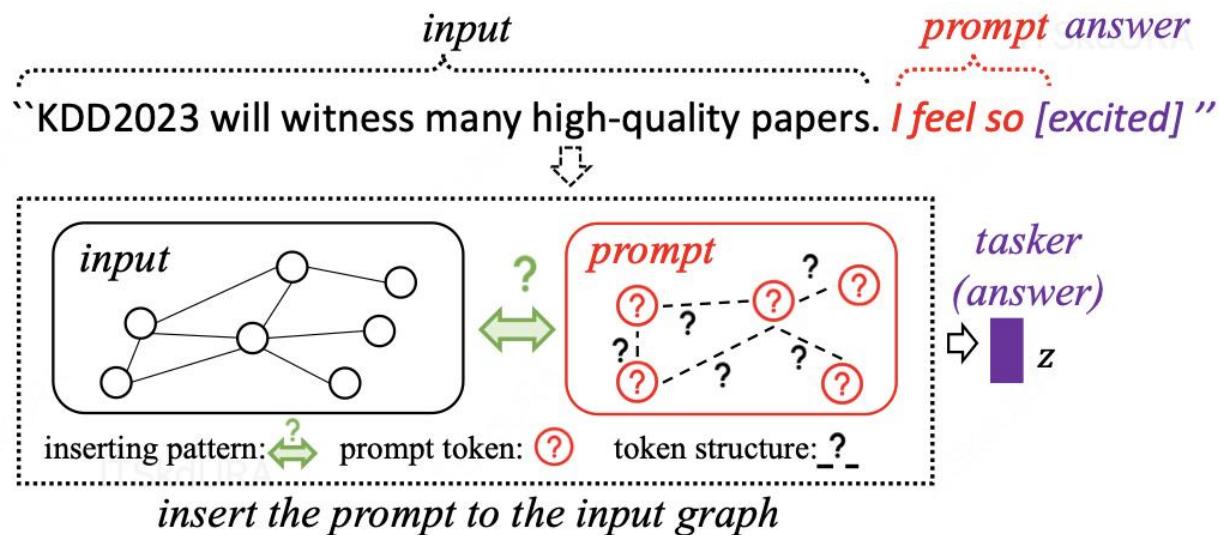
Aggregate dual prompt

$$\tilde{\mathbf{H}} = \text{AGGR}(\mathbf{H}_{\text{com}}, \mathbf{H}_{\text{op}}; \Delta)$$

All in One

Challenges:

- Graph prompt not only requires the prompt “content” but also needs to know how to organize these tokens and how to insert the prompt into the original graph.
- There is a huge difficulty in reconciling downstream problems to the pre-training task.
- Learning a reliable prompt needs huge manpower and is more sensitive in multi-task setting.



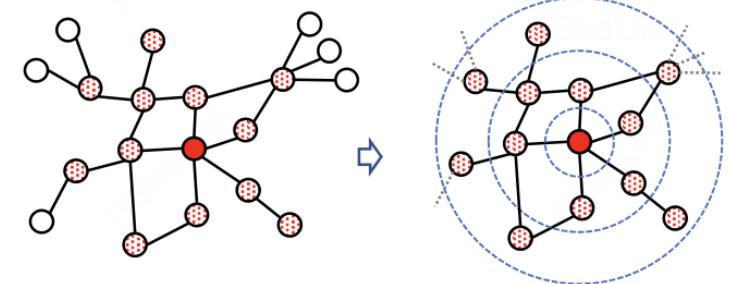
All in One

Reformulate Downstream Tasks :

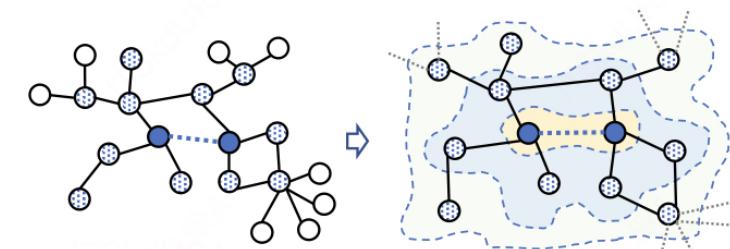
- This work reformulates node-level and edge-level tasks to graph-level tasks by building induced graphs for nodes and edges, respectively.

Prompt Graph Design:

- This work introduces some prompt nodes with unique connection relationships between them and adaptively insert them into the original input graph, in order to obtain a prompt graph.



(a) Induced graphs for nodes



(b) Induced graphs for edges

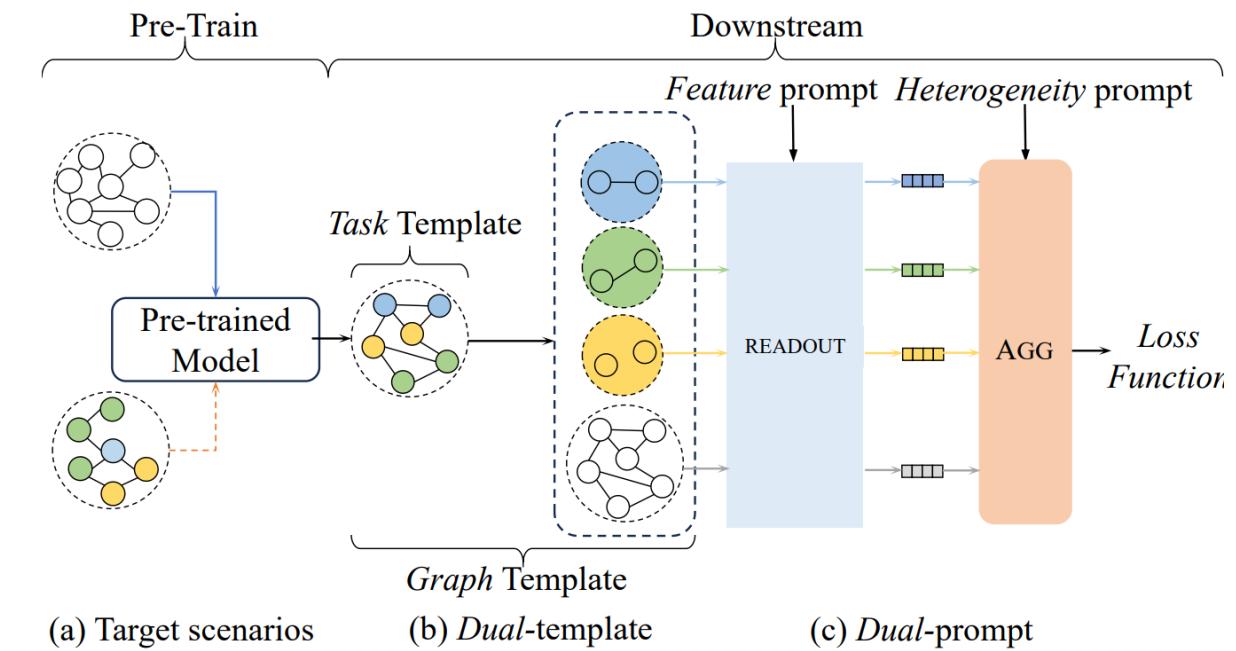
HGPrompt: Extending to heterogeneous graphs

Problem:

- Gap between homogeneous and heterogeneous graph.
- Different downstream tasks focus on heterogeneous aspect.

Insights :

- Dual-template:
Task + Graph template
- Dual-prompt:
Feature + Heterogeneity prompt

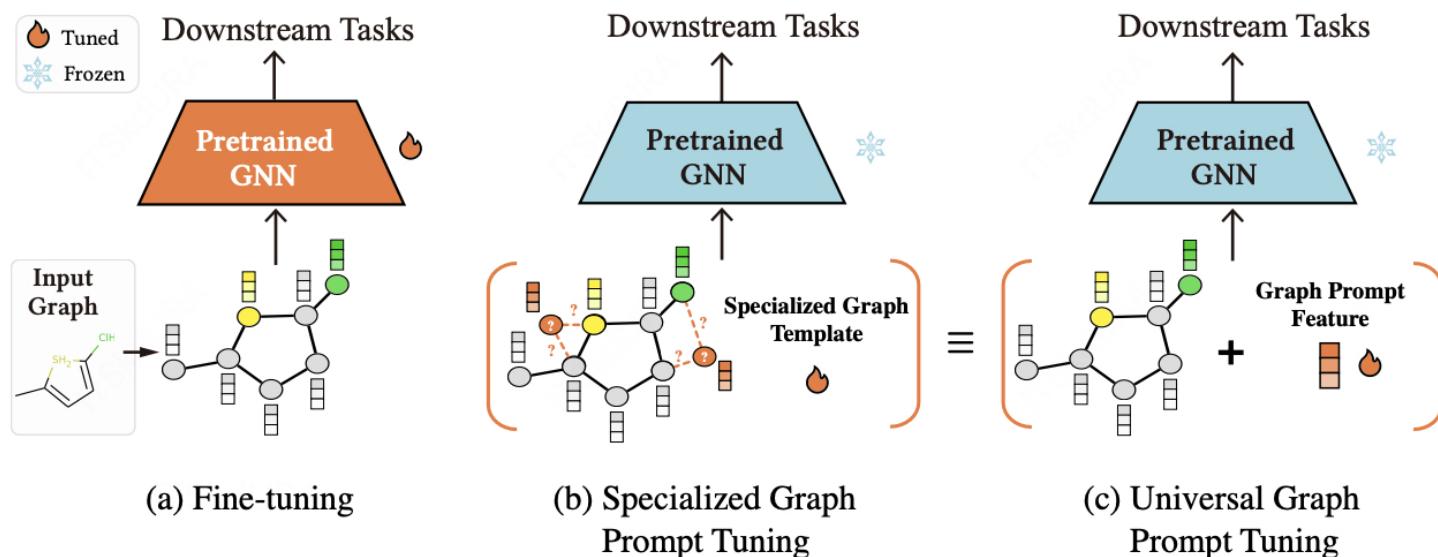


Challenges:

- Diverse pre-training strategies employed on graphs make it difficult to design suitable prompting functions.
- Existing prompt-based tuning methods for GNN models are predominantly designed based on intuition, lacking theoretical guarantees for their effectiveness.

Method:

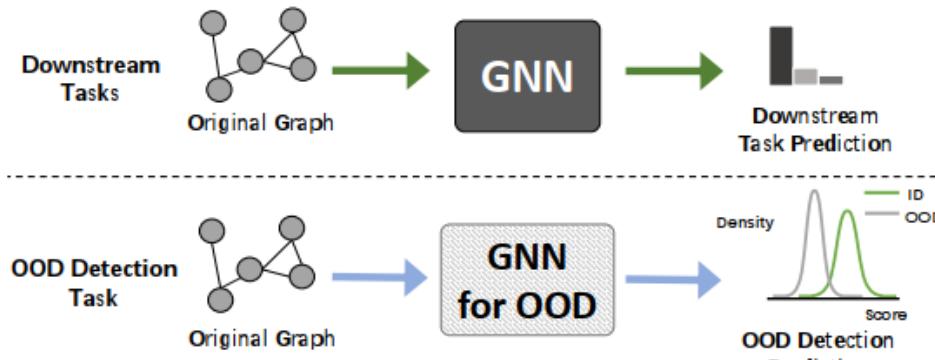
- This work proposes a universal prompt-based tuning method that can be applied to the pre-trained GNN models that employ any pre-training strategy.
- **GPF operates on the input graph's feature space** and involves adding a shared learnable vector to all node features in the graph.
- GPF-plus is a theoretically stronger variant of GPF, for practical application, which incorporates different prompted features for different nodes in the graph.



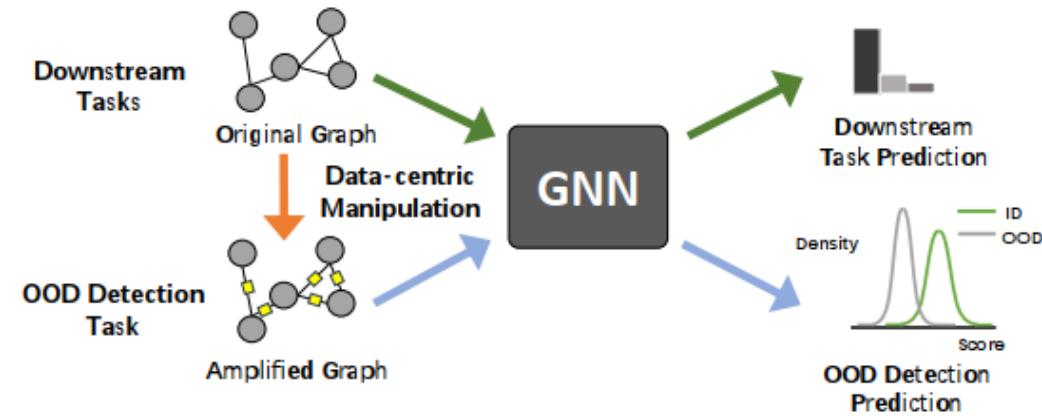
Motivation

- A reliable GNN should not only perform well on known samples (ID) but also identify graphs it has not been exposed to before (OOD).
- Existing works propose to train a neural network specialized for the OOD detection task.

Can we build a graph prompt that can solve OOD detection given a well-trained GNN?



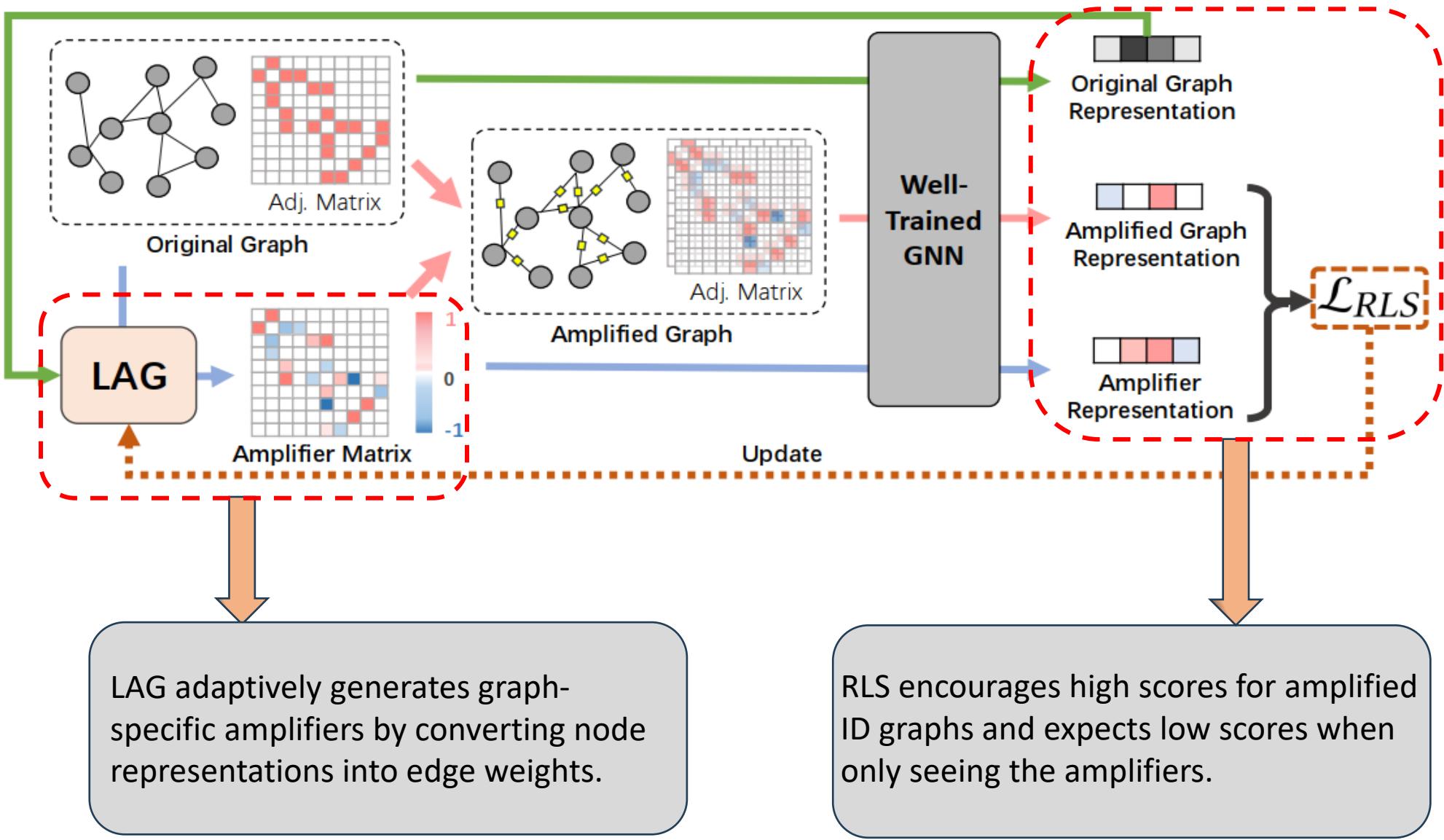
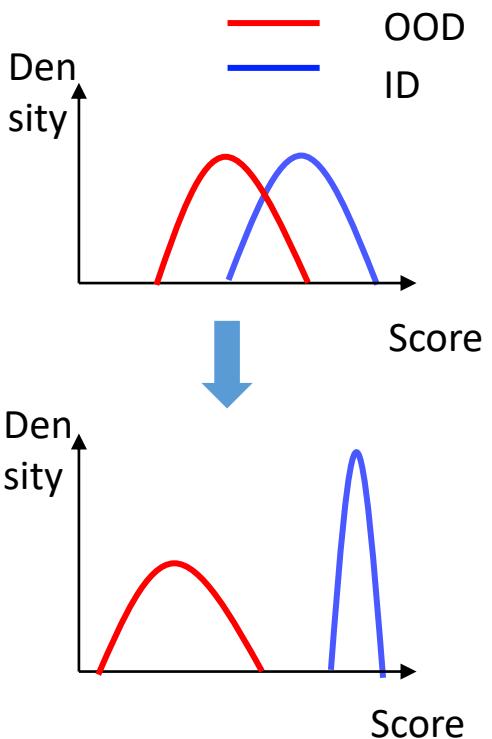
(1) Traditional works



(2) Our proposed framework

AAGOD

We modify edge weights as prompts to highlight the latent pattern of ID graphs, and thus enlarge the score gap between OOD and ID graphs.

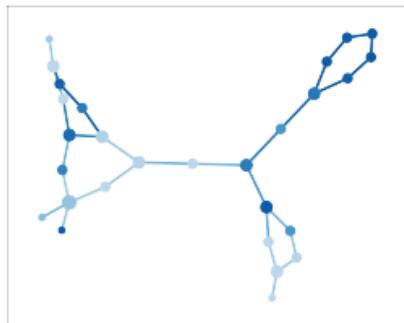


AAGOD

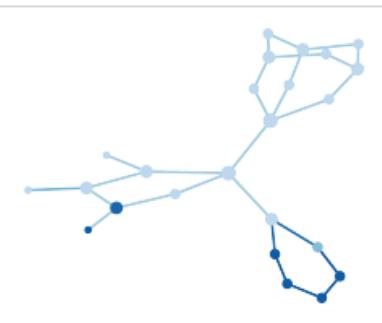
We conducted experiments on five dataset pairs over four GNNs to verify performance.

ID	OOD	Metric	GCL _S	GCL _{S+}	Improv.	GCL _L	GCL _{L+}	Improv.	JOAO _S	JOAO _{S+}	Improv.	JOAO _L	JOAO _{L+}	Improv.
ENZYMES	PROTEIN	AUC ↑	62.97	73.76	+17.14%	62.56	67.15	+7.34%	61.20	74.19	+21.23%	59.68	65.11	+9.10%
		AUPR ↑	62.47	75.27	+20.49%	65.45	65.18	-0.41%	61.30	77.10	+25.77%	64.16	64.49	+0.51%
		FPR95 ↓	93.33	88.33	-5.36%	93.30	85.00	-8.90%	90.00	81.67	-9.26%	96.67	85.00	-12.07%
IMDBM	IMDBB	AUC ↑	80.52	83.84	+4.12%	61.08	68.64	+12.38%	80.40	82.80	+2.99%	48.25	64.32	+33.31%
		AUPR ↑	74.43	80.16	+7.70%	59.52	68.03	+14.30%	74.70	77.77	+4.11%	47.88	61.62	+28.70%
		FPR95 ↓	38.67	38.33	-0.88%	96.67	91.33	-5.52%	44.70	42.00	-6.04%	98.00	94.00	-4.08%
BZR	COX2	AUC ↑	75.00	97.31	+29.75%	34.69	65.00	+87.37%	80.00	95.25	+19.06%	41.80	65.62	+56.99%
		AUPR ↑	62.41	97.17	+55.70%	39.07	62.89	+60.97%	67.10	94.34	+40.60%	56.70	67.22	+18.55%
		FPR95 ↓	47.50	15.00	-68.42%	92.50	80.00	-13.51%	37.50	12.50	-66.67%	97.50	97.50	0.00%
TOX21	SIDER	AUC ↑	68.04	71.27	+4.75%	53.44	58.25	+9.00%	53.46	69.39	+29.80%	53.64	55.67	+3.78%
		AUPR ↑	69.28	73.52	+6.12%	56.81	59.58	+4.88%	56.02	71.01	+26.76%	56.02	56.02	0.00%
		FPR95 ↓	90.42	89.53	-0.98%	94.25	92.72	-1.62%	95.66	90.55	-5.34%	95.66	89.66	-6.27%
BBBP	BACE	AUC ↑	77.07	80.64	+4.63%	46.74	50.53	+8.11%	75.48	78.54	+4.05%	43.96	51.28	+16.65%
		AUPR ↑	68.41	72.60	+6.12%	45.35	46.49	+2.51%	69.32	74.06	+6.84%	44.77	48.32	+7.93%
		FPR95 ↓	71.92	60.59	-15.75%	92.12	86.70	-5.88%	76.85	69.46	-9.62%	94.09	92.61	-1.57%

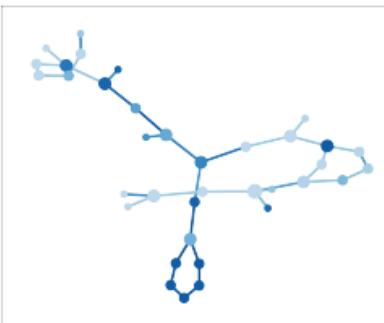
Case study: We visualize the learned graph prompts (i.e., amplifiers) for interpretability analysis.



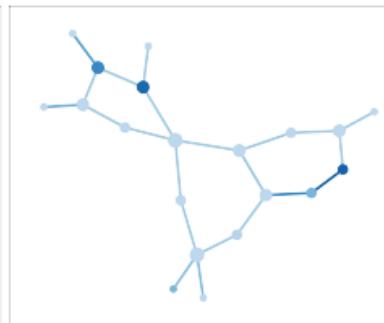
(a) ID



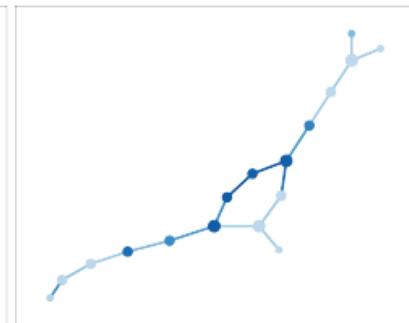
(b) ID



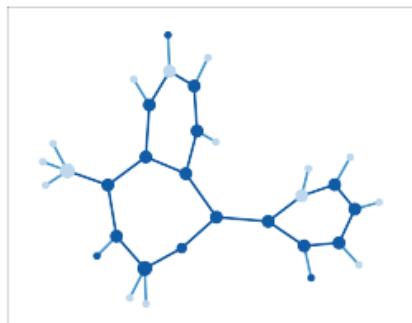
(c) ID



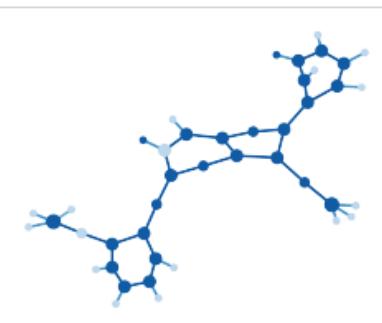
(d) OOD



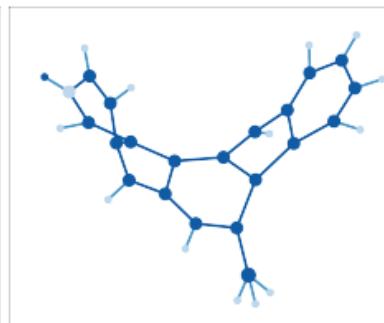
(e) OOD



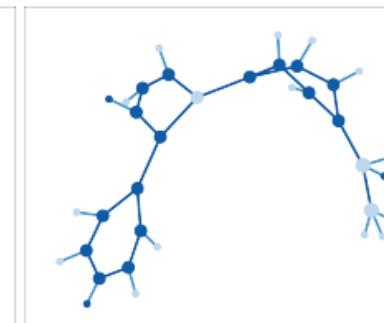
(a) ID



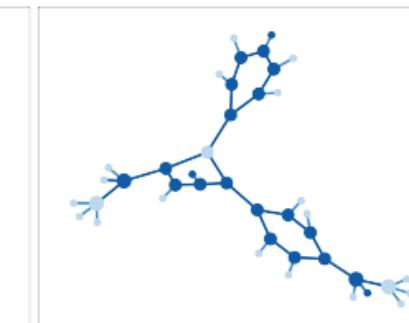
(b) ID



(c) ID

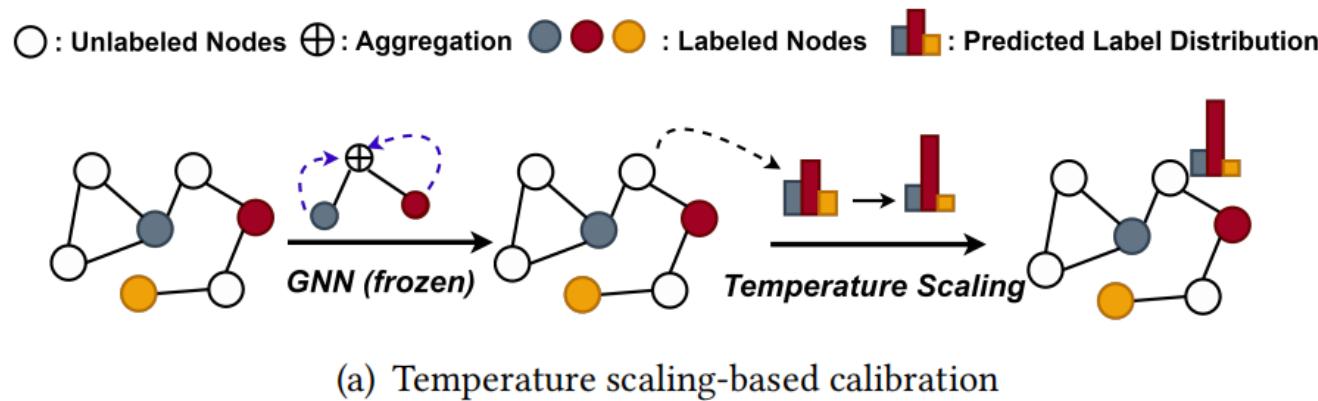


(d) OOD



(e) OOD

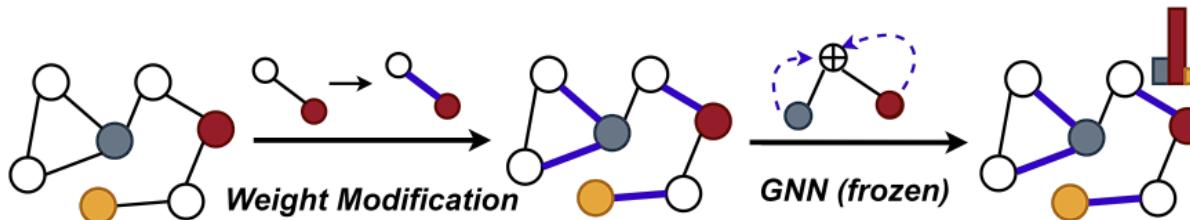
- Existing calibration methods focus on improving GNN models. Recent work has shown that the post-hoc methods, such as temperature scaling-based calibration, can achieve a better trade-off between accuracy and calibration.



- Through evaluating the expected calibration error (ECE) on Cora and Photo datasets with five different GNNs, we find that the ECEs on Cora (10.25%-18.02%) are always larger than those on Photo (4.38%-8.27%), indicating that **the calibration performance depends more on the datasets instead of GNN model**.

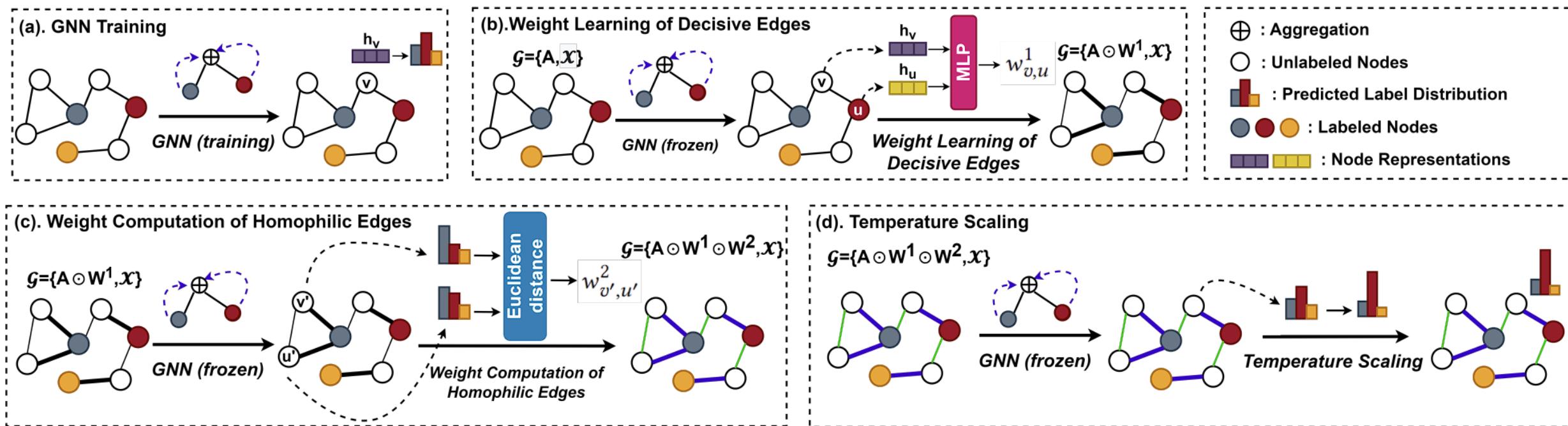
- Inspired by this phenomenon, we innovatively propose to calibrate GNNs from a data-centric perspective:

Can we modify the graph data instead for better calibration performance without losing accuracy?



(b) Data-centric calibration

- We propose Data-centric Graph Calibration (DCGC) with two edge weighting modules to adjust the input graph.



Summary

GNN-based models compares to foundation models with LLMs

- Advantage:
 - small parameter size, resulting in **low-cost training**
 - possess essential properties like **permutation invariance**
 - exhibit **strong performance** in scenarios without textual attributes
- Disadvantage:
 - **limited capacity** to harness **extensive knowledge** and can struggle to manifest emergent abilities
 - **underutilize** the information stored in **textual data**

Thanks
Q&A