ADC-GNN: Adaptive Dual-level Collaborative Graph Neural Networks for Graph Classification

Wan Tang[1], **Lu Bai**[1*], **Lixin Cui**[2], **Ming Li**[3], **Hangyuan Du**[4], **Jing Li**[5]
[1]School of Artificial Intelligence, Beijing Normal University, Beijing, China
[2]School of Information, Central University of Finance and Economics, Beijing, China
[3]Zhejiang Key Laboratory of Intelligent Education Technology and Application, Zhejiang Normal University, Jinhua, China
[4]School of Computer and Information Technology, Shanxi University, Taiyuan, China
[5]School of Geospatial Artificial Intelligence, East China Normal University, Shanghai, China
wan_tang@mail.bnu.edu.cn, bailu@bnu.edu.cn, cuilixin@cufe.edu.cn

Abstract

Most existing Graph Neural Networks (GNNs) rely on the node-level message passing or attention mechanisms to propagate and extract useful information. Although recent advances attempt to move beyond purely the node-level propagation by constructing high-level representations, these approaches are often constrained by pre-computed substructures or unidirectional bottom-up aggregations. Consequently, high-level structural semantics cannot effectively feed back to guide node representation learning, limiting the collaborative optimization between fine-grained features and macroscopic structural semantics. To address these limitations, we propose a novel Adaptive Dual-level Collaborative GNN (ADC-GNN) associated with an adaptive dual-level collaborative mechanism. We commence by introducing a set of global, learnable latent prototypes as high-level semantic references, and then employ a relaxed Sinkhorn algorithm to establish differentiable, non-collapsing assignments between nodes and prototypes. Based on these assignments, the ADC-GNN constructs high-level representations and enables interactions among them. We show that the ADC-GNN can inject the learned high-level information back into the node level, forming a closed-loop, bidirectional optimization process. Experiments demonstrate the superior performance of the proposed ADC-GNN on graph classification.

1 Introduction

Graph-based machine learning methods such as Graph Neural Networks (GNNs) and Graph Transformers (GTs) have been demonstrated as powerful tools for analyzing graph structured data associated with complex interactions. Typical instances include molecular property prediction [Stokes *et al.*, 2020; Fout *et al.*, 2017], link prediction of physical systems [Sanchez-Gonzalez *et al.*, 2018], social network analysis [Hamilton *et al.*, 2017; Wu *et al.*, 2020], etc. Most existing methods take nodes as fundamental computational units and capture local information through iterative node-wise message passing [Kipf and Welling, 2017; Bai *et al.*, 2023] or long-range dependencies through pairwise attention [Ying *et al.*, 2021; Rampášek *et al.*, 2022].

However, real-world graphs often exhibit higher-level structural semantics beyond individual nodes and edges. For instance, in molecular graphs, functional groups represent stable and reusable chemical substructures that largely determine molecular properties; in social networks, recurring community structures convey macroscopic functionalities. Such high-level structures can be viewed as certain forms of graph structural patterns characterized by larger receptive fields, stronger semantic stability, and reusability across graph instances, which are difficult to explicitly and efficiently capture and exploit using node-level interactions alone.

Existing methods that incorporate high-level structural information can be roughly divided into two main categories. The first category relies on pre-computed structural rules to construct high-level structures [Morris *et al.*, 2019; Gao *et al.*, 2022], such as manually defined substructures [Bevilacqua *et al.*, 2022; Ai *et al.*, 2024] or graph partitions [He *et al.*, 2023]. The resulting high-level structures are then encoded as high-level representations for downstream prediction. While these approaches enhance structural modeling to some extent, the resulting high-level structures remain fixed once constructed and are independent of downstream tasks and node features. As a result, such approaches struggle to adaptively capture optimal high-level structures relevant to the task, which in turn limits flexibility and generalization.

The second category induces high-level structures from node representations through learnable aggregation mechanisms. These include pooling or coarsening approaches [Ying *et al.*, 2018; Ranjan *et al.*, 2020; Song *et al.*, 2024; Bai *et al.*, 2024], which form coarse-grained graph structures by learned node clustering or selection, as well as methods that introduce learnable global latent units or prototypes [Baek *et al.*, 2021; Vincent-Cuaz *et al.*, 2022], associating nodes with high-level

*Corresponding Authors: Lu Bai

representations based on similarity measures such as attention, distance metrics, or optimal transport, etc. Although more flexible, these methods typically follow an unidirectional paradigm in which high-level representations are constructed bottom-up from node features [Bianchi and Lachi, 2023]. In this setting, high-level structures act as compressed summaries of node representations for pooling or readout, rather than an independent representation level with its own semantics that can provide global contextual information to inform or constrain node representation learning. Consequently, the interaction between fine-grained node-level representations and high-level structural semantics remains limited, hindering effective collaborative optimization across representation levels.

Based on the above observations, we argue that high-level structural semantics should be modeled as an explicit and learnable intermediate representation level that not only aggregates node information but also feeds back to guide node representation learning. This calls for a bidirectional information flow between node-level and high-level representations, enabling adaptive construction of high-level semantics while providing global contextual guidance to node updates.

To this end, we propose a novel Adaptive Dual-level Collaborative Graph Neural Network (ADC-GNN), a collaborative framework that learns graph information at both the node level and the high-level in a parallel and interactive manner. Specifically, at the node level, the ADC-GNN employs standard message passing to preserve the sparse topology and fine-grained structural information of the original graph. At the high level, it introduces a set of globally learnable latent prototypes that serve as stable semantic references shared across graph instances. To bridge the two representation levels, the ADC-GNN adopts a relaxed Sinkhorn algorithm [Chizat *et al.*, 2018] to compute dynamic soft assignment relationships between node representations and global prototypes. The resulting assignment matrix is used to construct adaptive high-level representations. This optimal-transport-based assignment mechanism provides a differentiable and non-collapsing way to model node-prototype interactions, enabling effective dual-level collaborative modeling.

Moreover, instead of constructing a high-level adjacency matrix, we directly apply a self-attention encoder to model interactions among high-level representations. Because the number of high-level units is much smaller than the number of nodes, this design significantly reduces computational and storage costs while maintaining strong expressive capacity. More importantly, after high-level interactions, the learned assignment relationships are used to project high-level information back to node representations. Consequently, each node integrates both local topological information and global semantic context. These enriched node representations then guide subsequent rounds of node-level propagation and high-level structure construction,forming a closed-loop, bidirectional optimization process between the two representation levels. In summary, our contributions are as follows.

- We propose a dual-level collaborative graph neural network framework that models high-level structures as explicit, learnable, and shared latent prototypes, establishing a closed-loop interaction between node-level and high-level representations.
- We introduce a relaxed Sinkhorn-based node-prototype assignment mechanism to alleviate assignment collapse, together with an efficient high-level interaction scheme, enabling adaptive and computationally scalable dual-level modeling in an end-to-end manner.
- Extensive experiments on multiple graph classification datasets demonstrate that the proposed ADC-GNN consistently outperforms existing methods with favorable computational efficiency and robustness.

2 Related Works

2.1 Modeling and Utilizing High-level Structure

Extracting pre-computed substructures from the original graph is a commom way to incorporate high-level structural information. For instance, ESAN [Bevilacqua *et al.*, 2022] aggregates predefined equivariant subgraphs and Graph ViT-Mixer [He *et al.*, 2023] adopts the METIS algorithm to partition graphs into substructures that are treated as patches for high-level representation modeling. A more flexible class of methods learns high-level structures directly from node representations through pooling or coarsening mechanisms. DiffPool [Ying *et al.*, 2018] introduces a learnable soft assignment matrix to cluster nodes into coarse groups and progressively build hierarchical graph representations. Other approaches, such as GMT [Baek *et al.*, 2021], leverages attention mechanisms to aggregate node features into global multi-set representations.

Remarks. Although these GNNs can effectively incorporate high-level information, they still suffer from some theoretical drawbacks. First, the fixed modeling approaches depend on the pre-detected substructures. Therefore, they cannot adaptively identify the optimal substructures for specific downstream tasks. Second, the modeling process of adaptive approaches is unstable and prone to the collapse problems, where the learned assignment matrix degenerates into trivial solutions [Ying *et al.*, 2018; Bianchi *et al.*, 2020]. Moreover, The bottom-up paradigm they follow hinders the reverse guidance of information interaction among these nodes, which does not fully utilize high-level information.

2.2 Sinkhorn Algorithm and Balanced Assignment

The Sinkhorn algorithm was originally proposed to approximate optimal transport solutions [Genevay *et al.*, 2018; Mena *et al.*, 2018; Chizat *et al.*, 2018]. In graph representation learning, Sinkhorn has been used to compute transport plans as alternatives to the attention mechanisms [Chen *et al.*, 2020], to measure distances between graphs and templates, also as feature descriptors [Vincent-Cuaz *et al.*, 2022], or to define losses for graph coarsening [Ma and Chen, 2021].

Remarks. Not for distance computation or explicit graph matching, our approach adopts Sinkhorn as an intra-layer normalization strategy. By leveraging its doubly stochastic constraints, Sinkhorn regularizes the node–prototype assignment matrix and prevents collapse to a few prototypes, thereby enabling stable bidirectional information flow between node-level and high-level semantic representations.

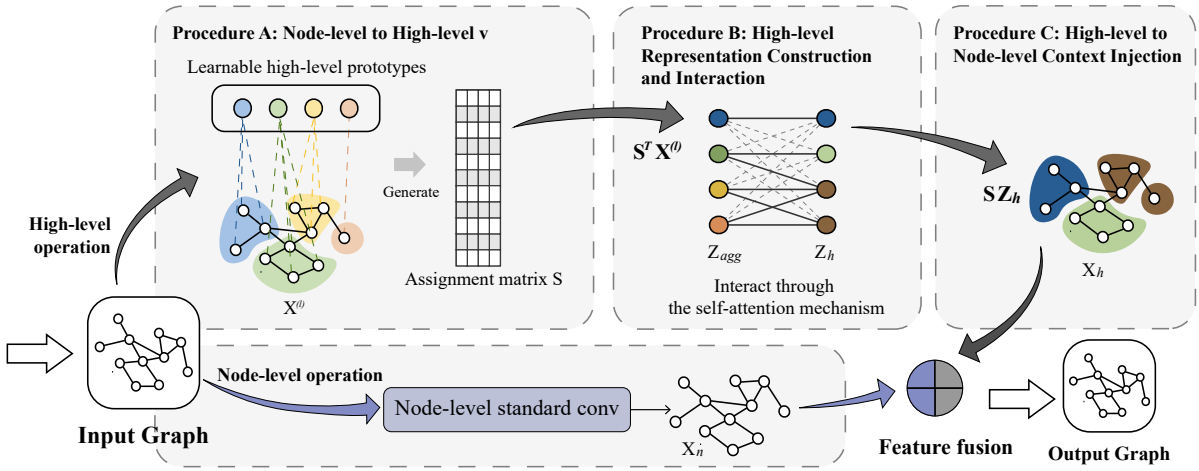


Figure 1: Illustration of the Dual-level Collaborative Convolution (DCC) Module.

3 Definitions of the Proposed ADC-GNN

3.1 Preliminaries

Graph Notations Let $G = (\mathcal{V}, \mathcal{E}) \in \mathcal{G}$ be a graph with node set $\mathcal{V}$ and edge set $\mathcal{E}$, where $|\mathcal{V}| = N$ and $|\mathcal{E}| = M$. Let $\mathbf{A} \in R^{N\times N}$ denote the adjacency matrix of G, and $\mathbf{X}^{(0)} \in R^{N\times c_0}$ denote its initial node feature matrix.

High-level Structural Information High-level structural information refers to graph patterns beyond individual nodes and edges, such as coarse-grained groups or substructures. In this paper, it is instantiated as learnable latent units, which derive their representation from the set of nodes composed to them. For a graph G at layer l,we define it as

$$\mathcal{H}^{(l)}(G) \triangleq \left(\mathbf{S}^{(l)}, \mathbf{Z}_h^{(l)}\right), \tag{1}$$

where $\mathbf{S}^{(l)} \in R^{N\times K}$ denotes the node-to-unit assignment matrix, capturing the composition of high-level structures, and $\mathbf{Z}_h^{(l)} \in R^{K\times d}$ denotes the interacted high-level representations, capturing their content and dependencies.

3.2 Dual-level Collaborative Convolution Module

The Dual-level Collaborative Convolution (DCC) module is the core component of ADC-GNN, designed to enable bidirectional and collaborative information exchange between fine-grained node-level representations and high-level representations. The whole stage has three procedures exhibited by Figure 1. First, an adaptive assignment between nodes and a set of learnable prototypes is learned, mapping the node-level representations and the high-level representations. Second, high-level representations are constructed based on the learned assignments and they can interact globally to capture macroscopic dependencies. Third, the interacted high-level contexts are injected back into node representations guided by the same assignments. These are fused with node representations obtained through node-level message passing to produce the final result. Together, these stages form a closed-loop collaborative update mechanism between node-level and high-level representations.

Procedure A: Node-level to High-level Assignment

Given the input graph at the l-th layer $G^{(l)} = (\mathbf{X}^{(l)}, \mathbf{A})$, where $\mathbf{X}^{(l)} \in R^{N\times d}$ denotes the node feature matrix and d is the feature dimension. We introduce a set of prototype vectors $\mathbf{P} \in R^{K\times d}$, where K denotes the number of prototypes. Each prototype p_k is a globally learnable parameter vector, rather than being computed from pre-defined graph rules or obtained by node clustering algorithms. It serves as a potential high-level semantic reference that adaptively aggregates nodes with similar attributes.

To measure the similarity between nodes and prototypes, we first compute a scaled cosine similarity matrix $\mathbf{M} \in R^{N\times K}$ as

$$\mathbf{M}_{ik} = \frac{1}{\epsilon}\bar{x}_i\bar{p}_k^\top, \tag{2}$$

where $\bar{x}_i$ and $\bar{p}_k$ denote the ℓ_2-normalized node feature $x_i^{(l)}$ and prototype vector p_k, respectively. The value $\mathbf{M}_{ik}$ represents their logit score. The temperature parameter ϵ controls the smoothness of the assignment. As $\epsilon \to 0$, the assignment approaches a hard assignment, while a larger ϵ leads to a more uniform distribution.

Our goal is to learn an assignment matrix $\mathbf{S} \in R^{N\times K}$ that satisfies two requirements, i.e., a) each node should be reasonably matched to its most similar prototypes; and b) assignment collapse should be avoided, where most nodes are assigned to only a few prototypes, resulting in redundant or degenerated high-level semantics. To this end, we adopt a Sinkhorn algorithm based on log-domain iterations and incorporate soft load-balancing constraints to enhance numerical stability and encourage balanced prototype utilization.

Specifically, we initialize the log-assignment matrix as $\mathbf{H}^{(0)} = \mathbf{M}$. At the t-th iteration, we first perform row normalization over the prototype dimension of $\mathbf{H}^{(t)} \in R^{N\times K}$, ensuring that the assignment weights of each node across all prototypes sum to one as follows

$$\mathbf{H}_{i:}^{(t)} \leftarrow \mathbf{H}_{i:}^{(t)} - \mathrm{LogSumExp}_j(\mathbf{H}_{ij}^{(t)}), \tag{3}$$

where $\mathrm{LogSumExp}_j(z_j) = \log\sum_j \exp(z_j)$.

Next, we employ a soft load-balancing constraint along the prototype dimension. We define the current log mass of the k-th prototype $\mu_k^{(t)}$ and its target log mass $\hat{\mu}^{(t)}$ as

$$\mu_k^{(t)} = \mathrm{LogSumExp}_i(\mathbf{H}_{ik}^{(t)}), \tag{4}$$

$$\hat{\mu}^{(t)} = \mathrm{LogSumExp}_{j,k}(\mathbf{H}_{jk}^{(t)}) - \log K. \tag{5}$$

Here, $\hat{\mu}^{(t)}$ corresponds to the target assignment under the assumption of approximately balanced prototype masses. The assignment matrix is then smoothly adjusted according to the discrepancy between the current and target masses

$$\mathbf{H}_{:k}^{(t+1)} \leftarrow \mathbf{H}_{:k}^{(t)} + \tau \cdot (\hat{\mu}^{(t)} - \mu_k^{(t)}), \tag{6}$$

where $\tau \in (0, 1]$ controls the strength of the load-balancing constraint. This update does not enforce strict equality of prototype masses; instead, it softly encourages balanced assignments across prototypes, alleviating assignment collapse.

After T iterations, the final log-assignment matrix is mapped back to the probability space, yielding the soft node-to-prototype assignment matrix as

$$\mathbf{S}_{ik} = \exp\Big(\mathbf{H}_{ik}^{(T)} - \mathrm{LogSumExp}_j(\mathbf{H}_{ij}^{(T)})\Big). \tag{7}$$

The resulting matrix $\mathbf{S}$ encodes the affinity between each node and each prototype, thereby providing the basis for subsequent high-level representation construction and node-level context injection.

Procedure B: High-level Representation Construction and Interaction

After obtaining the soft assignment matrix $\mathbf{S}$, we construct concrete high-level representations and conduct their global interactions. Specifically, the initial representation of each graph in high-level space is obtained by aggregating node features as

$$\mathbf{Z}_{agg} = \mathbf{S}^\top \mathbf{X}^{(l)} \in R^{K \times d}. \tag{8}$$

This operation can be regarded as an adaptive weighted pooling scheme that compresses node-level information into K compact high-level feature vectors.

However, the aggregated features $\mathbf{Z}_{agg}$ are isolated from each other at this stage and lack an understanding of dependencies among high-level units. Different from node-level representations whose interactions are constrained by the topology of the original graph, we do not explicitly construct an adjacency structure among high-level units. In fact, the adjacency relationship of these units is a dense graph that is almost fully connected. Therefore, we can model their pairwise interactions directly. We employ a standard Transformer block consisting of multi-head self-attention (MSA) and a feed-forward network (FFN) to perform global information exchange as follows

$$\mathbf{Z}_{in} = \mathbf{Z}_{agg}\mathbf{W}_{in}, \tag{9}$$

$$\tilde{\mathbf{Z}} = \mathrm{MSA}\left(\mathrm{LN}(\mathbf{Z}_{in})\right) + \mathbf{Z}_{in}, \tag{10}$$

$$\mathbf{Z}_h = \mathrm{FFN}\left(\mathrm{LN}(\tilde{\mathbf{Z}})\right) + \tilde{\mathbf{Z}}, \tag{11}$$

where $\mathbf{W}_{in} \in R^{d \times d'}$ is a learnable projection matrix and we set $d' = d$ in this work. $\mathrm{LN}(\cdot)$ denotes layer normalization, $\mathrm{MSA}(\cdot)$ computes semantic correlations among high-level units and performs global aggregation, $\tilde{\mathbf{Z}}$ is an intermediate representation, and $\mathrm{FFN}(\cdot)$ denotes a position-wise feed-forward network.

Each high-level unit can adaptively aggregate information from all other units according to semantic relevance. The resulting representation $\mathbf{Z}_h \in R^{K \times d}$ encodes global contextual information over the entire graph. Since $K \ll N$, the computational complexity of this interaction is $O(K^2)$, effectively avoiding the prohibitive cost of node-level self-attention.

Procedure C: High-level to Node-level Context Injection

After interacting at the high level, we further inject $\mathbf{Z}_h$ back into the node level as semantic context, enabling node representations to explicitly perceive the semantics of their associated high-level structures. This process relies on the learned soft node-prototype assignment matrix $\mathbf{S}$ from Section 3.2 procedure A, that is,

$$\mathbf{X}_h = \mathbf{S}\mathbf{Z}_h \in R^{N \times d}. \tag{12}$$

Since $\mathbf{S}_{ik}$ measures the degree to which the i-th node is associated with the k-th prototype, it also naturally defines the projection weights from the high-level semantic back to the node. As a result, $\mathbf{X}_h$ provides each node with global contextual information aligned with its role.

Meanwhile, a standard node-level message passing operation is performed to capture fine-grained local information, yielding node representations $\mathbf{X}_n \in R^{N \times d}$ as

$$\mathbf{X}_n = \mathrm{GNN}\left(\mathbf{A}, \mathbf{X}^{(l)}\right), \tag{13}$$

where $\mathrm{GNN}(\cdot)$ can be instantiated by any graph neural network convolution operation, such as GIN or GCN.

Considering that different nodes may rely on high-level context to different extents, for example, structurally central nodes may emphasize local connectivity. In contrast, bridging nodes may benefit more from global semantics. We introduce a learnable gating coefficient $\boldsymbol{\alpha}$ to dynamically regulate the fusion ratio as below

$$\boldsymbol{\alpha} = \sigma(\mathbf{W}_g[\mathbf{X}_n \parallel \mathbf{X}_h] + \mathbf{b}_g), \tag{14}$$

$$\mathbf{X}_{out} = \mathrm{Norm}(\mathbf{X}_n + \boldsymbol{\alpha} \odot \mathbf{X}_h), \tag{15}$$

where $\parallel$ denotes feature concatenation, $\mathbf{W}_g \in R^{2d \times d}$ and $\mathbf{b}_g \in R^d$ are parameters of the gating network, $\sigma(\cdot)$ is the Sigmoid activation function, and $\odot$ denotes element-wise multiplication. The fused representation $\mathbf{X}_{out}$ is then used as the input to the next convolutional layer.

Through this design, the ADC-GNN forms a complete closed loop. Specifically, node representations firstly construct globally interacted high-level structures, and the results are then fed back to refine node representations.

3.3 Overall Architecture

In this section, we present the overall architecture of the proposed ADC-GNN. As shown in Figure 2, the ADC-GNN consists of three sequential stages, i.e., initialization, collaborative encoding, and graph readout.

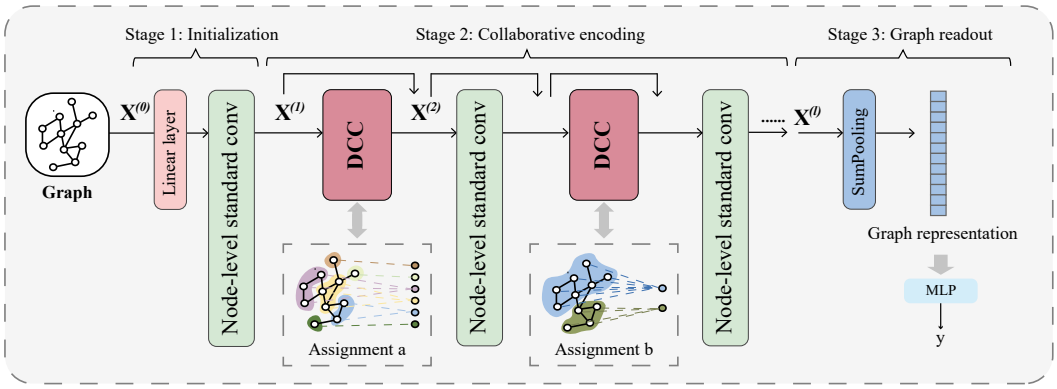


Figure 2: The architecture of the proposed ADC-GNN model. The assignment of a and b illustrates the configuration of prototype numbers and possible allocations across different DCC. By adjusting the number of prototypes, we control the semantic granularity of the induced high-level structures.

Given an input graph G with initial node features $\mathbf{X}^{(0)}$, we first map the node features into a continuous embedding space and apply a standard GNN layer for initialization,

$$\mathbf{X}^{(1)} = \mathrm{GNN}_{init}(\mathbf{X}^{(0)}, \mathbf{A}). \tag{16}$$

This initialization step injects basic local structural information into node representations, providing a stable foundation for subsequent feature based assignment and high-level semantic construction.

In the second stage, we alternately stack standard node-level GNN layers and Dual-level Collaborative Convolution (DCC) modules. Formally, the node representation update at the l-th layer is defined as

$$\mathbf{X}^{(l+1)} = \begin{cases} \mathrm{DCC}(\mathbf{X}^{(l)}, \mathbf{A}), & \text{if } l \bmod 2 \neq 0, \\ \mathrm{GNN}(\mathbf{X}^{(l)}, \mathbf{A}), & \text{otherwise.} \end{cases} \tag{17}$$

Each DCC module performs bidirectional refinement between node-level and high-level representations, serving as the core unit for dual-level collaborative evolution. The alternating stacking strategy enables this collaboration to be progressively propagated across network depth while maintaining computational efficiency. Residual connections [He *et al.*, 2016] are applied at each layer to stabilize deep training. In addition, an SE layer [Hu *et al.*, 2018] is employed after each DCC module to recalibrate feature channels and enhance discriminative capacity.

As the network deepens, node representations update from both local structural information and global semantic context, which in turn yields richer and appropriate node-prototype assignments in subsequent DCC modules.

Finally, after L layers of collaborative encoding, the final node representations $\mathbf{X}^{(L)}$ are aggregated via global sum pooling to obtain a graph-level representation $\mathbf{g} \in R^d$ as

$$\mathbf{g} = \mathrm{SumPooling}(\mathbf{X}^{(L)}) = \sum_{i=1}^{N} x_i^{(L)}, \tag{18}$$

which is then used for downstream graph classification tasks.

3.4 Complexity Analysis

We analyze the computational complexity of a single Dual-level Collaborative Convolution (DCC) module in the ADC-GNN. Let the input graph contain N nodes and E edges, with K high-level prototypes and Sinkhorn iteration count T. At the node level, the DCC module employs a standard GNN for local message passing, whose time complexity scales linearly with the number of edges as $O(E)$, consistent with conventional GNN models. In the assignment procedures, computing the similarity matrix requires $O(NK)$ operations, while T Sinkhorn iteration costs $O(TNK)$. Since T is typically set to a small constant (e.g., 2-3), the overall complexity of this stage is $O(NK)$. In the high-level interaction procedures, self-attention among the K high-level units incurs a time complexity of $O(K^2)$. Given that $K \ll N$, this design effectively avoids the quadratic complexity associated with node-level Transformers. Combining the above components, the overall time complexity of a single DCC module is $O\big(E + NK + K^2\big)$.

In terms of space complexity, the ADC-GNN adopts sparse storage at the node level and additionally maintains the node-prototype assignment matrix $\mathbf{S} \in R^{N \times K}$ and the high-level representations. The extra space complexity is therefore $O(NK + K)$. Consequently, the ADC-GNN introduces explicit global modeling capability while maintaining favorable computational and memory efficiency.

4 Experiments

In this section, we conduct extensive experiments to evaluate the performance of ADC-GNN. Specifically, we aim to: 1) verify its effectiveness on standard graph-classification benchmarks from TUDataset; 2) assess its generalization ability and long-range dependency modeling capability on more challenging OGB and LRGB datasets; 3) analyze its runtime efficiency; 4) perform ablation studies to justify the necessity of each module in the ADC-GNN; 5) analyze the performance of different assignment strategies; 6) discuss the selection of the number of prototypes and its implications and 7) provide visualizations to qualitatively demonstrate the ef-

	Biochemical Domain			Social Domain		
	MUTAG	PROTEINS	DD	IMDB-B	COLLAB	**AR**
# Graphs	188	1,113	1,178	1,000	5,000	–
# Classes	2	2	2	2	3	–
Avg. Nodes	17.93	39.06	284.32	19.77	74.49	–
GCN	69.50±1.78	73.24±0.73	72.05±0.55	73.26±0.46	80.59±0.27	9.0
GIN	81.39±1.53	71.46±1.66	70.79±1.17	72.78±0.86	78.19±0.63	10.6
GMT	83.44±1.33	75.09±0.59	78.72±0.59	73.48±0.76	80.74±0.54	5.4
GSAPool-MID	79.88±0.20	75.04±0.42	76.70±0.56	73.06±0.20	79.44±0.30	8.2
DiffPool	79.22±1.02	73.03±1.00	77.56±0.41	73.14±0.70	78.68±0.43	8.8
ASAP	77.83±1.49	73.92±0.63	76.58±1.04	72.81±0.50	78.64±0.50	9.8
SEP-G	85.56±1.09	76.42±0.39	77.98±0.57	74.12±0.56	81.28±0.15	4.0
Cluster-GT	87.11±1.37	76.48±0.86	79.15±0.63	75.10±0.84	80.43±0.52	3.6
GKNN GL	85.24±2.28	75.36±1.12	–	69.90±1.44	–	7.3
ENAHPool	–	**77.12±0.15**	79.91±0.25	74.54±0.42	81.09±0.51	2.3
AEGK(A)	89.11±0.40	75.11±0.28	77.82±0.29	–	–	4.7
UnionSNN	87.31±5.29	75.02±2.50	77.00±2.37	–	–	6.7
ADC-GNN (Ours)	**90.69±1.06**	76.78±0.81	**80.12±0.29**	**75.54±0.39**	**82.51±0.26**	**1.2**

Table 1: Classification accuracy (% ± standard error) on TUDataset benchmarks. The best results are highlighted in **bold**. "–" indicates the result is not reported. AR denotes Average Rank.

fectiveness of our design.

Datasets and Experimental Setups We evaluate our method on 1) Five publicly available graph-classification datasets from TUDataset, including three biochemica datasets and two social interaction datasets; 2) ogbg-molhiv from the Open Graph Benchmark (OGB) [Hu *et al.*, 2020]; 3) Peptides-func from the Long-Range Graph Benchmark (LRGB) [Dwivedi *et al.*, 2022b]. Basic dataset statistics are summarized in the headers of Tables 1 and 2.

For the proposed ADC-GNN, we perform hyperparameter tuning using a grid search strategy. Specifically, the number of layers is selected from the range of 1 to 7. The temperature parameter ϵ is searched in $[0.01, 0.2]$ with a step size of 0.02; the number of Sinkhorn iterations t is chosen from $[2, 3]$; and the load-balancing relaxation coefficient τ is tuned in $[0.05, 0.1, 0.2, 0.3, 0.6, 0.8]$. The number of high-level prototypes in each DCC module is selected from $[32, 16, 8]$ for all datasets, and we use orthogonal initialization to ensure its initial diversity. All experiments are conducted on NVIDIA RTX 3090 GPUs with 24GB of RAM. The additional implementation details are available in our project repository[1].

4.1 Comparisons on the TUDataset

We compare the proposed ADC-GNN with 12 graph classification methods, including 2 classic node-level message-passing models 1) GCN [Kipf and Welling, 2017], 2) GIN [Xu *et al.*, 2019]; 6 models related to our proposed method 1) GMT [Baek *et al.*, 2021], 2) GSAPool-MID [Liu *et al.*, 2023], 3) DiffPool [Ying *et al.*, 2018], 4) ASAP [Ranjan *et al.*, 2020], 5) SEP-G [Wu *et al.*, 2022], 6) Cluster-GT [Huang *et al.*, 2024]; and 4 the latest outstanding models 1) GKNN GL [Cosmo *et al.*, 2025], 2) ENAHPool [Zhao *et al.*, 2025], 3) AEGK(A) [Bai *et al.*, 2025], 4) UnionSNN [Xu *et al.*, 2024]. In our experiments, to ensure a fair comparison, we perform 10-fold cross-validation and repeat each experiment ten times, reporting the mean classification accuracy along with its standard error. For datasets with node labels, we adopt one-hot encodings of the node labels as initial node features; for datasets without node labels, we use node degrees as initial features. GIN [Xu *et al.*, 2019] is chosen as the node-level convolutional backbone of ADC-GNN.

[1]https://github.com/Volento/ADC-GNN

	ogbg-molhiv	Peptides-func	**AR**
# Graphs	41,127	15,535	–
Avg. Nodes	25.5	150.9	–
Metric	AUROC ↑	Avg. Precision ↑	–
GraphGPS	0.7880±0.0101	0.6535±0.0041	6.5
Graph-MIP-Mixer	0.7997±0.0102	0.6970±0.0080	3.0
GRIT	0.7835±0.0054	0.6988±0.0082	5.0
Subgraphormer	**0.8038±0.0192**	0.6415±0.0052	4.5
Graph-Mamba	–	0.6739±0.0087	6.0
GIN+	0.7928±0.0099	0.7059±0.0089	3.0
GatedGCN+PE+VNG	0.7910±0.0086	0.6822±0.0052	5.0
ADC-GNN (Ours)	0.7993±0.0124	**0.7113±0.0025**	**2.0**

Table 2: Results on OGB and LRGB benchmarks. Best results are highlighted in **bold**. "–" indicates the result is not reported. AR denotes Average Rank.

Results and Analyses. The experimental results on the TUDataset are reported in Table 1. As can be seen, the proposed ADC-GNN outperforms the remaining models significantly. Specifically, except for PROTEINS, our ADC-GNN achieved

Method	Time (s/epoch) ↓	Params.
GIN+	12.60	∼500k
GraphGPS	96.00	∼550k
Subgraphormer	67.82	∼300k
ADC-GNN (Ours)	30.20	∼500k

Table 3: Efficiency comparison on ogbg-molhiv.

	PROTEINS	DD
ADC-GNN (Full)	**76.78±0.81**	**80.12±0.29**
ADC-GNN-NoS	76.19±0.54	78.61±0.34
ADC-GNN-NoA	76.46±0.67	77.71±0.38
ADC-GNN-NoH	75.56±0.40	76.74±0.26
	IMDB-B	COLLAB
ADC-GNN (Full)	**75.54±0.39**	**82.51±0.26**
ADC-GNN-NoS	75.35±0.34	82.24±0.14
ADC-GNN-NoA	75.30±0.15	81.28±0.22
ADC-GNN-NoH	75.30±0.35	81.17±0.15

Table 4: Classification accuracy for ablation study.

the best classification performance. This validates that the proposed architecture can yield powerful multi-scale graph representations adaptively and is suitable for varying datasets.

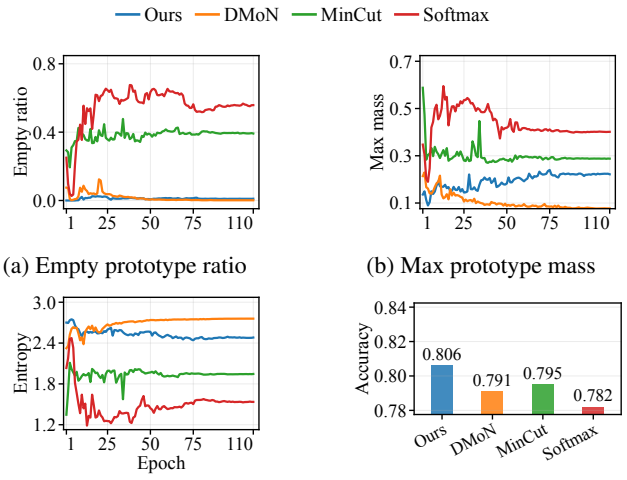
4.2 Comparisons on the OGB and LRGB

We compare our ADC-GNN with a set of recent representative methods, including 1) GraphGPS [Rampášek *et al.*, 2022], 2) Graph-MIP-Mixer [He *et al.*, 2023], 3) GRIT [Ma *et al.*, 2023], 4) Subgraphormer [Bar-Shalom *et al.*, 2024], 5) Graph-Mamba [Wang *et al.*, 2024], 6) GIN+ [Luo *et al.*, 2025], and 7) GatedGCN+PE+VNG [Southern *et al.*, 2025]. Due to the characteristics of the datasets, we follow the setup of GIN+ [Luo *et al.*, 2025], which adopts GINE, an edge-aware extension of GIN [Hu *et al.*, 2020], as the node-level convolutional backbone and uses the concatenation of atom features and Random Walk Structural Encoding (RWSE) [Dwivedi *et al.*, 2022a] as the initial node features. The model is constrained to approximately 500K parameters. We report the mean performance and standard deviation over five different runs.

Results and Analyses. As shown in Table 2, our model exhibits strong competitiveness, particularly on the dataset requiring long-range dependencies. This advantage stems from our high-level mechanism, where nodes that are distant in the original graph are compressed into high-level units and can exchange information directly. The refined high-level information is then injected back to the original nodes to guide subsequent message passing, thereby naturally enhancing long-range communication.

4.3 Runtime Efficiency Comparison

Table 3 reports the runtime results on ogbg-molhiv, where all models are evaluated on n NVIDIA A100 GPUs for fair comparison. Due to the additional high-level module, our model is slightly slower than plain GIN, but remains faster than other transformer-based models, indicating competitive efficiency.



(a) Empty prototype ratio (b) Max prototype mass

(c) Prototype usage entropy (d) Classification accuracy

Figure 3: Analysis of different assignment strategies.

4.4 The Ablation Study

To investigate the contribution of each component in our ADC-GNN, we design three ablation variants, namely the ADC-GNN-NoS, ADC-GNN-NoA, and ADC-GNN-NoH. In the ADC-GNN-NoS, we remove the Sinkhorn constraint and allow potential assignment collapse. In the ADC-GNN-NoA, we construct the high-level adjacency matrix $\mathbf{A_h} = \mathbf{S}^\top \mathbf{A}\mathbf{S}$ and perform standard message passing on the high-level graph instead of the original attention mechanism. In the ADC-GNN-NoH, we entirely remove the high-level module. We evaluate the full model and all variants under the same hyperparameter settings. As shown in Table 4, removing any component leads to performance degradation, indicating that all designs jointly contribute to the effectiveness of our proposed ADC-GNN.

4.5 The Analysis of Assignment Strategies

We further analyze the effect of the node-prototype assignment strategies on the collapse problem. We replace the assignment module in ADC-GNN with three representative alternatives, including 1) DMoN-style collapse regularization [Tsitsulin *et al.*, 2023], 2) MinCut-style orthogonality regularization [Bianchi *et al.*, 2020], and 3) unconstrained Softmax. For the assignment matrix $\mathbf{S} \in R^{N\times K}$, we define the normalized usage mass of the k-th prototype as $m_k = \frac{1}{N}\sum_{i=1}^{N} \mathbf{S}_{ik}$. Based on m_k, we use three diagnostic metrics to measure prototype utilization,

$$R_{\text{empty}} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{1}(m_k < \delta), \tag{19}$$

$$H_{\text{proto}} = -\sum_{k=1}^{K} m_k \log(m_k + \epsilon), \tag{20}$$

# Prototypes	8	16	32	64
PROTEINS	0.7610	0.7552	0.7601	0.7561
DD	0.7924	0.7805	0.7831	0.7971
IMDB-B	0.7475	0.7440	0.7470	0.7450
COLLAB	0.8137	0.8141	0.8155	0.8130

Table 5: Sensitivity analysis to the number of prototypes.

$$M_{\max} = \max_k m_k, \tag{21}$$

where R_{empty} measures the ratio of inactive prototypes, $\delta = 0.1/K$, H_{proto} characterizes the overall balance of prototype usage, and $M_{\max}$ indicates whether the assignment mass is overly concentrated on a small number of prototypes. An ideal assignment mechanism should avoid empty prototypes and excessive concentration, while maintaining sufficiently clear node-prototype preferences.

As shown in Figure 3, unconstrained Softmax leads to a higher empty-prototype ratio and a larger maximum prototype mass, suggesting that row-wise normalization alone is insufficient to regulate global prototype usage. DMoN and MinCut regularization mitigate collapse to some extent, but they may still suffer from skewed prototype usage or overly uniform assignments during training. In contrast, the relaxed Sinkhorn assignment maintains a lower empty-prototype ratio, a moderate maximum prototype mass, and stable prototype-usage entropy, while also achieving the best classification performance. These results indicate that the relaxed Sinkhorn assignment provides a more reliable routing mechanism for bidirectional information exchange and contributes to the final results.

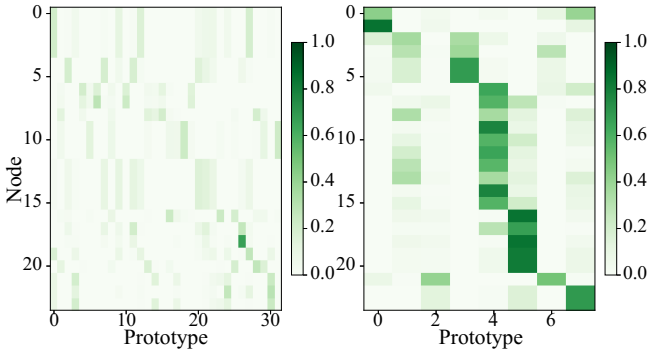
4.6 Sensitivity to the Number of Prototypes

To study the effect of high-level prototype numbers on ADC-GNN, we varied the number of prototypes while keeping other hyperparameters fixed. All experiments used four layers (two DCC layers) with consistent prototypes per layer. Table 5 shows the performance. Note that the results may differ from those reported in Table 1 due to different settings. Our main observation is that the model is robust to the prototype number within a practical range, with performance varying but not dramatically. Since prototypes are learned end-to-end, the model can adaptively redistribute semantic roles across them. We find that a moderate number of prototypes strikes a good balance between performance and efficiency, with values up to 64 sufficient for most datasets.

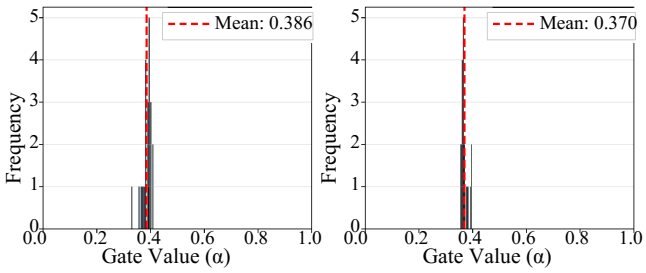
4.7 The Visualization Evaluation

In our visualization analysis, we focus on two aspects: 1) the node-to-prototype assignment results in each DCC layer, to understand how high-level units are composed of underlying nodes; 2) the fusion patterns between high-level information and node-level information across different layers. Figure 4 presents the visualization of a graph instance from the PROTEINS dataset after 50 training epochs, with two DCC layers, located on Layer 1 and Layer 3, respectively.

As shown in Figure 4(a), the color intensity indicates the assignment weight of each node to the high-level prototype.



(a) Assignments visualization. The x-axis corresponds to different prototypes, and the y-axis corresponds to graph nodes. Left: Layer 1; Right: Layer 3



(b) Fusion gates visualization. The x-axis denotes the learnable gating coefficient, while the y-axis denotes the number of nodes associated with each coefficient value. Left: Layer 1; Right: Layer 3

Figure 4: Visualization of the DCC module for a sample graph.

We observe that each node exhibits a relatively clear preference toward specific prototypes, and all prototypes are effectively utilized. It means assignment collapse or uniform distribution across prototypes does not happen, and the model has learned a good correspondence relationship.

Regarding the injection of high-level information into nodes, Figure 4(b) illustrates that different nodes rely on high-level information to varying degrees. For this graph, the importance ratio of high-level information to node information is approximately 4:6, which suggests that node-level features and high-level representations in our ADC-GNN collaboratively contribute to the final graph representation.

5 Conclusions

In this paper, we have proposed a GNN method, namely ADC-GNN, for graph classification. By introducing learnable high-level prototypes and a relaxed Sinkhorn-based assignment mechanism, we have shown that ADC-GNN derives rich high-level representations for each node and reintegrates them into the node-level features to generate enhanced node representations, thereby collaboratively capturing both global and local information across multiple levels. Moreover, the model enhances long-range information exchange while maintaining favorable computational efficiency. Experimental results demonstrate the effectiveness of ADC-GNN for graph classification.

Acknowledgments

This work is supported by National Natural Science Foundation of China (No. 62576371, 62576198 and 62536006), and the Open Project Foundation of Key Laboratory of Computation Intelligence and Chinese Information Processing of Ministry of Education and Key Laboratory of Data Intelligence and Cognitive Computing of Shanxi Province.

References

[Ai *et al.*, 2024] Xing Ai, Chengyu Sun, Zhihong Zhang, and Edwin R. Hancock. Two-level graph neural network. *IEEE Trans. Neural Networks Learn. Syst.*, 35(4):4593–4606, 2024.

[Baek *et al.*, 2021] Jinheon Baek, Minki Kang, and Sung Ju Hwang. Accurate learning of graph representations with graph multiset pooling. In *Proceedings of ICLR*, 2021.

[Bai *et al.*, 2023] Lu Bai, Yuhang Jiao, Lixin Cui, Luca Rossi, Yue Wang, Philip S. Yu, and Edwin R. Hancock. Learning graph convolutional networks based on quantum vertex information propagation. *IEEE Trans. Knowl. Data Eng.*, 35(2):1747–1760, 2023.

[Bai *et al.*, 2024] Lu Bai, Zhuo Xu, Lixin Cui, Ming Li, Yue Wang, and Edwin R. Hancock. HC-GAE: the hierarchical cluster-based graph auto-encoder for graph representation learning. In *Proceedings of NeurIPS*, 2024.

[Bai *et al.*, 2025] Lu Bai, Lixin Cui, Ming Li, Peng Ren, Yue Wang, Lichi Zhang, Philip S. Yu, and Edwin R. Hancock. AEGK: aligned entropic graph kernels through continuous-time quantum walks. *IEEE Trans. Knowl. Data Eng.*, 37(3):1064–1078, 2025.

[Bar-Shalom *et al.*, 2024] Guy Bar-Shalom, Beatrice Bevilacqua, and Haggai Maron. Subgraphormer: Unifying subgraph gnns and graph transformers via graph products. In *Proceedings of ICML*. OpenReview.net, 2024.

[Bevilacqua *et al.*, 2022] Beatrice Bevilacqua, Fabrizio Frasca, Derek Lim, Balasubramaniam Srinivasan, Chen Cai, Gopinath Balamurugan, Michael M. Bronstein, and Haggai Maron. Equivariant subgraph aggregation networks. In *Proceedings of ICLR*. OpenReview.net, 2022.

[Bianchi and Lachi, 2023] Filippo Maria Bianchi and Veronica Lachi. The expressive power of pooling in graph neural networks. In *Proceedings of NeurIPS*, pages 71603–71618, 2023.

[Bianchi *et al.*, 2020] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In *Proceedings of ICML*, volume 119, pages 874–883. PMLR, 2020.

[Chen *et al.*, 2020] Benson Chen, Gary Bécigneul, Octavian-Eugen Ganea, Regina Barzilay, and Tommi Jaakkola. Optimal transport graph neural networks. *arXiv preprint arXiv:2006.04804*, 2020.

[Chizat *et al.*, 2018] Lenaïc Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. Scaling algorithms for unbalanced optimal transport problems. *Math. Comput.*, 87(314):2563–2609, 2018.

[Cosmo *et al.*, 2025] Luca Cosmo, Giorgia Minello, Alessandro Bicciato, Michael M. Bronstein, Emanuele Rodolà, Luca Rossi, and Andrea Torsello. Graph kernel neural networks. *IEEE Trans. Neural Networks Learn. Syst.*, 36(4):6257–6270, 2025.

[Dwivedi *et al.*, 2022a] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *Proceedings of ICLR*. OpenReview.net, 2022.

[Dwivedi *et al.*, 2022b] Vijay Prakash Dwivedi, Ladislav Rampásek, Michael Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. In *Proceedings of NeurIPS*, 2022.

[Fout *et al.*, 2017] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. In *Proceedings of NIPS*, 2017.

[Gao *et al.*, 2022] Han Gao, Xu Han, Jiaoyang Huang, Jian-Xun Wang, and Liping Liu. Patchgt: Transformer over non-trainable clusters for learning graph representations. In *Proceedings of LoG*, pages 27–1. PMLR, 2022.

[Genevay *et al.*, 2018] Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences. In *Proceedings of AISTATS*, pages 1608–1617. PMLR, 2018.

[Hamilton *et al.*, 2017] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of NIPS*, page 1025–1035. Curran Associates Inc., 2017.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of CVPR*, pages 770–778, 2016.

[He *et al.*, 2023] Xiaoxin He, Bryan Hooi, Thomas Laurent, Adam Perold, Yann LeCun, and Xavier Bresson. A generalization of vit/mlp-mixer to graphs. In *Proceedings of ICML*, pages 12724–12745. PMLR, 2023.

[Hu *et al.*, 2018] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of CVPR*, pages 7132–7141, 2018.

[Hu *et al.*, 2020] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Proceedings of NeurIPS*, 2020.

[Huang *et al.*, 2024] Siyuan Huang, Yunchong Song, Jiayue Zhou, and Zhouhan Lin. Cluster-wise graph transformer with dual-granularity kernelized attention. In *Proceedings of NeurIPS*, 2024.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of ICLR*. OpenReview.net, 2017.

[Liu *et al.*, 2023] Chuang Liu, Yibing Zhan, Baosheng Yu, Liu Liu, Bo Du, Wenbin Hu, and Tongliang Liu. On exploring node-feature and graph-structure diversities for node drop graph pooling. *Neural Networks*, 167:559–571, 2023.

[Luo *et al.*, 2025] Yuankai Luo, Lei Shi, and Xiao-Ming Wu. Can classic gnns be strong baselines for graph-level tasks? simple architectures meet excellence. In *Proceedings of ICML*. OpenReview.net, 2025.

[Ma and Chen, 2021] Tengfei Ma and Jie Chen. Unsupervised learning of graph hierarchical abstractions with differentiable coarsening and optimal transport. In *Proceedings of AAAI*, pages 8856–8864, 2021.

[Ma *et al.*, 2023] Liheng Ma, Chen Lin, Derek Lim, Adriana Romero-Soriano, Puneet K. Dokania, Mark Coates, Philip H. S. Torr, and Ser-Nam Lim. Graph inductive biases in transformers without message passing. In *Proceedings of ICML*, volume 202, pages 23321–23337. PMLR, 2023.

[Mena *et al.*, 2018] Gonzalo E. Mena, David Belanger, Scott W. Linderman, and Jasper Snoek. Learning latent permutations with gumbel-sinkhorn networks. In *Proceedings of ICLR*. OpenReview.net, 2018.

[Morris *et al.*, 2019] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of AAAI*, volume 33, pages 4602–4609, 2019.

[Rampášek *et al.*, 2022] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In *Proceedings of NeurIPS*, pages 14501–14515, 2022.

[Ranjan *et al.*, 2020] Ekagra Ranjan, Soumya Sanyal, and Partha Talukdar. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In *Proceedings of AAAI*, volume 34, pages 5470–5477, 2020.

[Sanchez-Gonzalez *et al.*, 2018] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *Proceedings of ICML*, pages 4470–4479. PMLR, 2018.

[Song *et al.*, 2024] Yunchong Song, Siyuan Huang, Xinbing Wang, Chenghu Zhou, and Zhouhan Lin. Graph parsing networks. In *Proceedings of ICLR*. OpenReview.net, 2024.

[Southern *et al.*, 2025] Joshua Southern, Francesco Di Giovanni, Michael M. Bronstein, and Johannes F. Lutzeyer. Understanding virtual nodes: Oversquashing and node heterogeneity. In *Proceedings of ICLR*. OpenReview.net, 2025.

[Stokes *et al.*, 2020] Jonathan M Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M Donghia, Craig R MacNair, Shawn French, Lindsey A Carfrae, Zohar Bloom-Ackermann, et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702, 2020.

[Tsitsulin *et al.*, 2023] Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. Graph clustering with graph neural networks. *J. Mach. Learn. Res.*, 24:127:1–127:21, 2023.

[Vincent-Cuaz *et al.*, 2022] Cédric Vincent-Cuaz, Rémi Flamary, Marco Corneli, Titouan Vayer, and Nicolas Courty. Template based graph neural network with optimal transport distances. In *Proceedings of NeurIPS*, pages 11800–11814, 2022.

[Wang *et al.*, 2024] Chloe Wang, Oleksii Tsepa, Jun Ma, and Bo Wang. Graph-mamba: Towards long-range graph sequence modeling with selective state spaces. *CoRR*, abs/2402.00789, 2024.

[Wu *et al.*, 2020] Yongji Wu, Defu Lian, Yiheng Xu, Le Wu, and Enhong Chen. Graph convolutional networks with markov random field reasoning for social spammer detection. In *Proceedings of AAAI*, volume 34, pages 1054–1061, 2020.

[Wu *et al.*, 2022] Junran Wu, Xueyuan Chen, Ke Xu, and Shangzhe Li. Structural entropy guided graph hierarchical pooling. In *Proceedings of ICML*, volume 162, pages 24017–24030. PMLR, 2022.

[Xu *et al.*, 2019] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *Proceedings of ICLR*. OpenReview.net, 2019.

[Xu *et al.*, 2024] Jiaxing Xu, Aihu Zhang, Qingtian Bian, Vijay Prakash Dwivedi, and Yiping Ke. Union subgraph neural networks. In *Proceedings of AAAI*, pages 16173–16183, 2024.

[Ying *et al.*, 2018] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of NeurIPS*, 2018.

[Ying *et al.*, 2021] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In *Proceedings of NeurIPS*, pages 28877–28888, 2021.

[Zhao *et al.*, 2025] Zhehan Zhao, Lu Bai, Lixin Cui, Ming Li, Ziyu Lyu, Lixiang Xu, Yue Wang, and Edwin R. Hancock. Enahpool: The edge-node attention-based hierarchical pooling for graph neural networks. In *Proceedings of ICML*. OpenReview.net, 2025.