Electronic Research Archive

https://www.aimspress.com/journal/era

ERA, 33(12): 7360–7384.
DOI: 10.3934/era.2025325
Received: 29 September 2025
Revised: 18 November 2025
Accepted: 21 November 2025
Published: 05 December 2025

Research article

GraphDAFI: A graph representation learning framework with degree-aware feature interaction for node classification

Yiming Chen, Ying Zhang, Wenrui Guan and Wengang Jiang*

School of Automation, Jiangsu University of Science and Technology, Zhenjiang 212100, China

* **Correspondence:** Email: a_1_2_3@163.com.

Abstract: Graph neural networks (GNNs) have been widely studied to handle graph-structured data due to their superior learning capability. Despite the successful applications of GNNs in many areas, their performance suffers heavily from the imbalanced degree distribution (long-tail issue). Most prior studies tackle this issue by graph augmentation, which explicitly increases the communication among nodes by optimizing original topology. In this paper, we employed the perspective of taylor interaction to explore the long-tail issue, and analyzed that there is insufficient interaction between low-degree nodes and their neighbors. In detail, we proposed a novel GNN framework named with degree-aware feature interaction (GraphDAFI), in order to bridge the gap of neighborhood aggregation between head-node embeddings and tail-node embeddings. GraphDAFI comprises two collaborative modules: adaptive feature interaction and degree-aware neighborhood transfer. Adaptive feature interaction leverages node embeddings of the current layer and interactions of the historical layer to perceive potential local information. Then, a unified feature encoder was designed that enhances the interaction to increase the model's generalization ability. To inject relevant information into low-degree nodes, a degree-aware neighborhood transfer was developed, which updates the node-edge adjacency matrix through a degree-aware strategy to achieve knowledge transfer. Experimental results demonstrate that GraphDAFI achieves excellent performance in semi-supervised node classification compared with the state-of-the-art models.

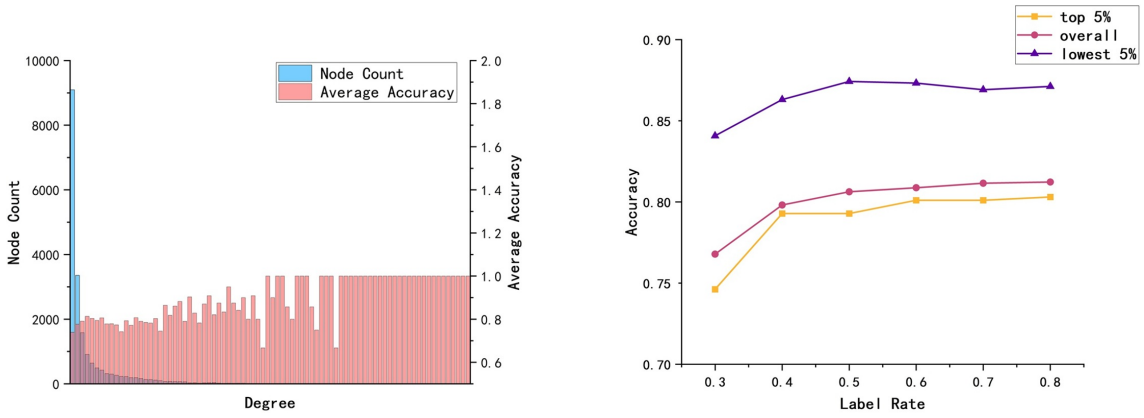
Keywords: semi-supervised learning; graph neural network; imbalance learning; knowledge transfer; feature interaction

1. Introduction

In recent years, there has been a surge in approaches that handle graph-structured data via graph neural networks (GNNs) [1], which have achieved astonishing success in many real-world scenarios such as recommendation systems [2], bioinformatics [3] and traffic prediction [4]. Under the

assumption of homogeneity [5], most GNNs hinge on a neighborhood-aware mechanism, where each node recursively receives and aggregates features from its neighbors [6]. As a representative downstream task of GNNs, semi-supervised node classification aims to correctly classify the unlabeled nodes with partly labeled nodes and a given topology [7]. The critical challenge of this task is to reveal and explore the potential information for enhancing the learning performance of node embeddings [8, 9].

Despite the remarkable performance of node classification achieved by emerging GNNs, most of them assume that graph-structured data follows a balanced situation where the number of neighbors (i.e., node degree) are uniform [10]. Unfortunately, real-world graphs often suffer from the long-tailed issue (i.e., imbalance issue), where the node degree approximately obeys the power-law distribution [11]. For example, in the citation network (Pubmed) where each node represents a scientific paper and the links indicate citation relationships among these papers, groundbreaking research may have thousands of followers while most papers receive only a few citations. This means that a significant fraction of tail nodes with low degree can only perceive limited information in the recursive neighborhood-aware process [12, 13]. As a consequence, GNNs trained on a dataset with a long-tailed distribution are likely to fail in detecting such significant tail cases, as depicted in Figure 1(a). Compared with the head nodes that have abundant structural connectivity, these tail nodes struggle to access information from labeled nodes, resulting in performance degradation, as shown in Figure 1(b).



(a) (b)

Figure 1. Long-tail issue in Pubmed. (a) Degree frequency and corresponding accuracy statistics; (b) comparison of head and tail nodes with different label rates. All reported with balanced accuracy (bAcc) (%) for node classification.

A few recent studies have attempted to mitigate this issue through the graph structure learning strategy. Specifically, graph structure learning aims to obtain an optimized topology via the targeted loss function, so as to transfer the ideal information of head nodes to tail nodes with few links. For instance, long-tailed graph neural networks via graph structure learning (LTSL-GNN) [14] allows information-rich head nodes to optimize the graph structure through multi-metric learning and further enhancing the embeddings of the tail nodes with the learned graph structure, and Tail-GNN [15] constructs the variable ties between a target node and its neighbors to achieve a neighborhood

translation from the structurally rich head nodes and obtain the robust tail node embeddings.

Essentially, these methods rely on a fundamental assumption, i.e., the optimized topology by graph structure learning is reliable for downstream tasks. However, since graph-structured datasets are extracted from complex interactive systems through manual predefined rules, the above assumption cannot be always satisfied. For head nodes, the increased links make their receptive field further expanded and accept redundant information, which reduces the discrimination of the node embeddings [16]. For tail nodes that have sparse message propagation, they depend on the information of high-degree nodes given by neighborhood translation. While the information from head nodes provides valuable insights for enhancing the representation learning of tail nodes, it also introduces potential noise [17]. These unintended influences can compromise the stability of representation learning for tail nodes, leading to a compromise on the robustness and reliability of tail-node embeddings.

To alleviate this limitation, we employ the perspective of taylor interaction to explore the long-tail issue. [18] proves that the network output can be mathematically decomposed as the sum of two typical types of effects caused by input variables, i.e., the independent effect and interaction effect. Furthermore, [19] shows that coefficients of the neighborhood interaction effect are relatively small in most GNN models. On this basis, we argue that insufficient neighborhood interaction of tail nodes results in deteriorating the performance of their embeddings. Specifically, the complex characteristics of graphs require the approaches to carry powerful nonlinear modeling capability [20, 21]. However, the majority of message propagation depends on the general aggregation strategy, which does not comprehensively utilize the relationship information [22]. Such simple combination of neighbors without considering the interactions between nodes limits the capability of capturing intricate relational dependencies in graphs [23]. Furthermore, compared to the head nodes, a notable limitation of tail-node embeddings is that their sparse neighbors exacerbate the attenuation of interaction effects. This naturally leads us to consider one practical question: How can the interaction effect for tail nodes be supplemented without disrupting the original topology?

In this study, we propose a graph representation learning framework with degree-aware feature interaction (GraphDAFI), in order to bridge the gap of neighborhood aggregation between head-node embeddings and tail-node embeddings. The core idea is to inject additional interaction information from head nodes into tail nodes for enhancing their raw suboptimal embeddings. Specifically, we first construct the adaptive feature interaction of pairwise nodes to reveal the potential dependencies in the graph. As a novel explicit interaction model, it effectively fuses node embeddings of neighbors and interaction information which are originally derived by a factorization machine to extract the adaptive interactions. Then, a universal feature encoder is designed to enhance both node embeddings and adaptive interactions through feature augmentation, which further improves the generalization of our model. In order to transmit the constructed adaptive interactions to the tail nodes, we propose a degree-aware neighborhood transfer with interaction. Specifically, to preserve the original topology, we extend the node-edge adjacency matrix onto the original adjacency matrix as the initialization for neighborhood transfer. Immediately, the node-edge adjacency matrix is applied to a tail-oriented learning strategy, aiming to deliver the interaction information (i.e., edge index) of paired nodes to tail nodes with similar embeddings. The main contributions can be summarized as follows.

- Considering that the manual neighborhood transfer lacks an interaction effect, a novel explicit adaptive interaction is designed by fusing node embeddings and feature interaction constructed

by a factorization machine, which can be automatically learned for various real-life scenarios.
- To conduct the derived interactions into tail nodes, a degree-aware neighborhood transfer is devised that adopts the node-edge adjacency relationship to expand the existing message propagation. Through the tail-oriented learning strategy, the node-edge adjacency matrix is updated to build channels for head-tail knowledge transfer.
- We conduct extensive experiments on six datasets in semi-supervised node classification, and the experimental results demonstrate that GraphDAFI achieves state-of-the-art performance. We apply GraphDAFI to the recommendation system, and segment the head and tail nodes for statistical analysis and visualization to show the effectiveness at alleviating the long-tail issue.

2. Related work

In line with the focus of our work, we briefly review the previous work in the two following areas: 1) GNNs with imbalanced learning, and 2) feature interaction.

2.1. Graph neural networks with imbalanced learning

There are currently different groups of methods for reducing the bias caused by graph-structured data imbalance [17]. Specifically, real-world graphs often exhibit a highly skewed distribution, where some segments contain an abundance of objects while others are significantly insufficient [24]. Due to the complex dependencies among nodes in graphs (i.e., non-independent and identically distributed), it is challenging to directly apply traditional approaches to address imbalance issues on graphs. As a result, there has been a surge in approaches that alleviate graph imbalance issues via graph structure learning [25]. Graph structure learning is to construct an optimized adjacency matrix via random or adaptive methods, which can directly increase the message propagation between the head and tail nodes. For instance, LTSL-GNN iteratively learns the graph structure and tail node embedding enhancement parameters, allowing information-rich head nodes to optimize the graph structure through multi-metric learning and further enhancing the embeddings of the tail nodes with the learned graph structure. CensNet [26] quantifies the similarity between hub vertices and their neighbors, and applies graph transformations through edge weight adjustments and self-connections, effectively mitigating this issue. Self-supervised-learning degree-specific GCN (SL-DSGCN) [27] leverages degree-specific graph convolution network (GCN) layers and the self-supervised-learning with the Bayesian teacher network to introduce more labeled neighbors for low-degree nodes. Intuitively, the key insight of the approaches is to improve communications between high-resource nodes and low-resource nodes, facilitating distribution alignment in training.

2.2. Feature interaction

As a technology for data augmentation, feature interaction aims to capture nonlinear distributions of node features, revealing the complex interaction patterns among nodes in graph-structured data [18]. More precisely, it refers to combining two or more features such that the model can capture higher-order interaction terms and learn more complex nonlinear relationships [22, 23]. It is worth noting that Hu et al. [19] mathematically demonstrated that the output of existing GNNs lacks interactive effects. Therefore, the neighborhood aggregation with feature interaction is introduced, and becomes a simple and promising approach that can effectively enhance the node embedding. For

example, dual feature interaction-based graph convolutional network (DFI-GCN) [28] extracts the arbitrary-order interactions between different features via Newton's identities, and integrates interactions into node embeddings using an attention mechanism; Adaptive factorization network (AFN) [29] transforms feature embeddings into a logarithmic space and constructs arbitrary-order feature interactions through feed forward architecture, and GraphAIR [19] takes into account the interactions between nodes in different channels, thus it represents the interactions as the inner product of node embeddings between two channels. However, the above interactions are manually designed and thus lack adaptive capabilities, which cannot be applied to unfamiliar interaction patterns. Furthermore, there are few GCNs that consider feature interactions in addressing the imbalance issue of graph-structured data.

3. Preliminaries

3.1. Notations

Let $G = (V, E)$ be an original graph, and $V = \{v_1, v_2, ..., v_N\}$ and $E = \{e_1, e_2, ..., e_M\}$ represent the node set and the edge set, respectively. $\mathcal{N}(i)$ is the set containing the neighbors of node v_i as well as node v_i itself. We denote the adjacency matrix of G as $A \in \{0, 1\}^{N\times N}$, where each element $A_{(i,j)} = 1$ if and only if (iff) $(v_i, v_j) \in E$. $X \in \mathbb{R}^{N\times F}$ is the feature matrix, where $x_i \in \mathbb{R}^F$ is the feature of v_i. $Y \in \mathbb{R}^{N\times C}$ denotes the label indicator matrix, where C is the number of classes and $y_i \in \mathbb{R}^C$ is the ground-truth label of node v_i. In Table 1, notations commonly used in the article are defined.

Table 1. Notation description.

Notation	Description
G	Original graph
A	Node adjacency matrix of G
V	Node set of G
E	Edge set of G
X	Feature matrix
Y	Label indicator matrix
C	Number of classes
$\mathcal{N}$	Neighbor node set
α	Aggregation weight

3.2. Node aggregation methods of neighborhood awareness

The basic idea of node aggregation is to learn a parameter-sharing aggregator, which takes features of node v_i and its neighbors $v_j \in \mathcal{N}(i)$ as inputs and outputs a new embedding for node v_i. As mentioned above, we can classify node aggregation methods into two types based on neighborhood-aware pattern: greedy neighborhood neighborhood-aware (GNA) and node-level neighborhood-aware (NLNA) [30]. In the following, we discard the subscripts of our notations for a moment, assuming A is the node adjacency matrix.

In the GNA methods, as one of the most popular approaches, GCN defines the aggregation coefficients as the symmetrically normalized adjacency matrix $\hat{A}$ with $\hat{A} = \tilde{D}^{-\frac{1}{2}}(A + I)\tilde{D}^{-\frac{1}{2}}$, where I is

the identity matrix and $\tilde{D}_{(i,i)} = \sum_j A_{(i,j)}$. The updated embedding of node v_i based on GCN can be formulated as:

$$h_i^{(l+1)} = \sigma(\sum_{v_j \in \mathcal{N}(i)} \hat{A}_{(i,j)} h_j^{(l)} W^{(l)}), \tag{3.1}$$

where $h_i^{(l)}$ is the embedding of the node v_i from the l-th convolutional layer. σ is a nonlinear activation function which we used in this paper as *sigmoid*, and $W^{(l)}$ denotes the weight matrix at the l-th layer.

In the NLNA methods, sampling and attention mechanisms are usually adopted to selectively aggregate information from relevant neighbors. The updated embedding of node v_i based on sampling mechanisms can be expressed as:

$$h_i^{(l+1)} = \sigma(W^{(l)} \cdot AGGREGATE(\{h_i^l\} \cup \{h_j^l, v_j \in \mathcal{N}_s(i)\})), \tag{3.2}$$

where $AGGREGATE$ is a aggregation function, which commonly includes mean, pooling, and LSTM. $\mathcal{N}_s(i)$ represents the partial neighbors of node v_i, which are obtained by different sampling mechanisms. In addition, the updated embedding of node v_i based on attention mechanisms can be formulated as:

$$h_i^{(l+1)} = \sigma(\sum_{v_j \in \mathcal{N}(i)} a_{ij}^{(l)} h_j^{(l)} W^{(l)}), \tag{3.3}$$

where $a_{ij}^{(l)}$ is the attention coefficient between nodes v_i and v_j at the l-th layer.

4. Method

4.1. Overview of the framework

The overall structure of GraphDAFI is illustrated in Figure 2. GraphDAFI differs from the traditional GNN framework for a long-tail issue in two main aspects. 1) GraphDAFI considers the importance of the interactive effect, incorporating a novel adaptive interaction strategy to explore local latent patterns. 2) GraphDAFI includes a degree-aware neighborhood transfer, where the devised interaction can be injected into the node representation by continuously updating the tail-oriented transfer matrix.

The following sections are structured to present the details and implementation of GraphDAFI. In Section 3.2, we present the construction of adaptive interaction. In Section 3.3, we provide details on the degree-aware neighborhood transfer. In Section 3.4, we introduce our proposed loss function and the process of model training.

4.2. The construction of adaptive interaction

As mentioned above, we design a novel adaptive interaction construction method, as shown in Figure 2. By extracting node interactions of the historical layer and node embeddings of the current layer, adaptive node interaction can be learned. In the following, we provide a detailed procedure of the adaptive interaction.

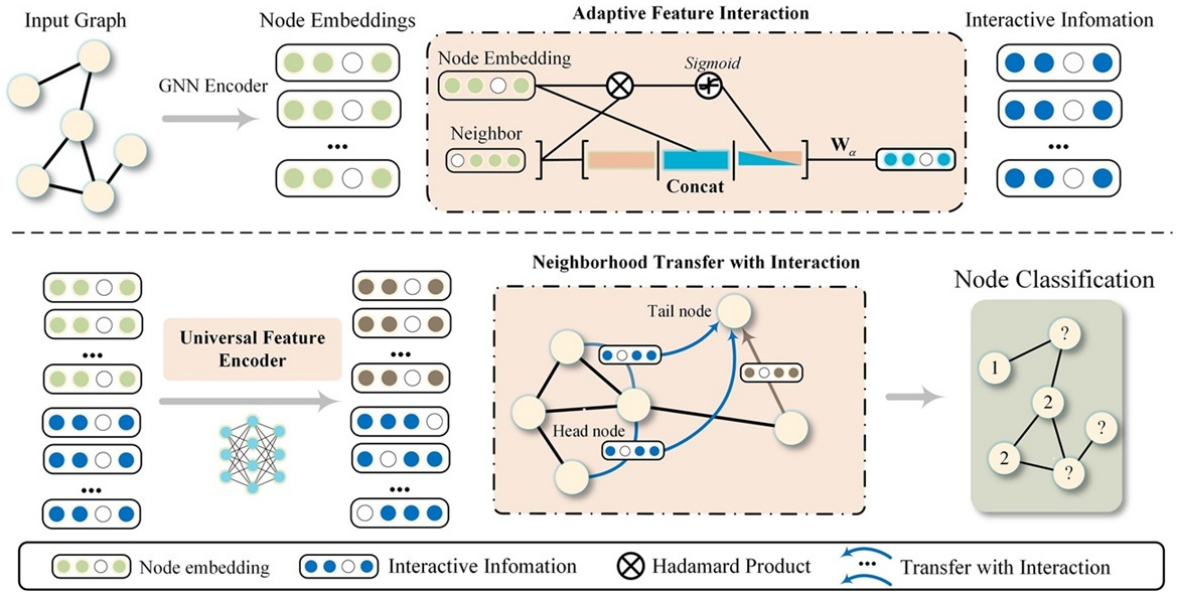


Figure 2. The framework of GraphDAFI.

As discussed in Section 3.2, GraphDAFI uses the node aggregation method of GNA. Thus, the updated process of node embedding H can be expressed as follows:

$$h_i^{l+1} = \sigma\Big(\sum_{v_j \in \mathcal{N}(i)} \hat{A}_{(i,j)} h_j^l W_v^{(l)}\Big), \tag{4.1}$$

where h_i^l is the node embedding of the node v_i from the l-th layer with $h_i^0 = x_i$. $\hat{A}$ is the symmetrically normalized adjacency matrix, and $W_v^{(l)}$ denotes the weight matrix of the l-th layer.

Taylor interaction effects point out that the output of a linear learner $f(\cdot)$ (i.e., GCN learner) can be decomposed into K-order Taylor expansions, which are expanded at a baseline point $b = [b_1, ..., b_n]^T$.

$$f(\mathbf{x}) = f(\mathbf{b}) + \underbrace{\sum_{i=1}^{n} \frac{1}{1!} \cdot \frac{\partial f(\mathbf{b})}{\partial x_i} \cdot (x_i - b_i)}_{Independent \quad effects} + \underbrace{\sum_{i=1}^{n} \sum_{j=1}^{n} \frac{1}{2!} \cdot \frac{\partial^2 f(\mathbf{b})}{\partial x_i \partial x_j} \cdot (x_i - b_i)(x_j - b_j) + \cdots}_{Interaction \quad effects} \tag{4.2}$$

It can be concluded that the interaction term of the learner is the key to improve the nonlinear fitting ability of the model. Unfortunately, the interaction term coefficient decreases rapidly with the increase of the order, and the existing literature [19] has proved that the third-order interaction term coefficient is less than 1/48, which limits the model learning ability. Therefore, a straightforward strategy is to explicitly construct interaction items and add them to the learning process. On this basis, a natural idea to model neighborhood interaction of pairwise nodes (v_i, v_j) is formulated as:

$$z_{(i,j)} = \left(\sum_{v_k \in \mathcal{N}(i)} \hat{A}_{(i,k)} h_j^l W_v^{(l)}\right) \odot \left(\sum_{v_r \in \mathcal{N}(j)} \hat{A}_{(j,r)} h_j^l W_v^{(l)}\right), \tag{4.3}$$

where $z_{(i,j)}$ is the neighborhood interaction representation between v_i and v_j, and $\odot$ is the element-wise multiplication operator. It should be pointed out that this approximation method lacks consideration

of the adaptive coefficient, and it is easy to cause performance degradation if it is blindly added to the forward propagation.

GraphDAFI first extracts the embedding of pairwise nodes, and constructs the interactive information $(h_i^{l+1} \odot h_j^{l+1}) \in \mathbb{R}^{|E| \times d'}$ of the current layer through the factorization machine form. In order to comprehensively perceive node relationships, GraphDAFI additionally considers the interaction information of the previous layer and merges it into the current layer through residual connection. Then, we use the attention mechanism to adaptively fuse the interaction information from different layers into the embedded dimension through a layer of MLP learners (the learnable feature matrix $S \in \mathbb{R}^{d \times d'}$). Therefore, the learning of adaptive interaction Z is defined as follows:

$$z_{(i,j)}^{l+1} = \sigma\left(\left[z_{(i,j)}^{l} \oplus (h_i^{l+1} \odot h_j^{l+1})\right] S^T\right), \tag{4.4}$$

where $z_{(i,j)}^{l}$ is the adaptive interaction between nodes v_i and v_j from the l-th layer, and $z_{(i,j)}^{0} = \sigma(h_i^0 \odot h_j^0)$. We denote $\oplus$ as the concatenation operation, respectively. $S \in \mathbb{R}^{d \times d'}$ is a learnable feature matrix, where d is the dimension of the current layer and d' is the sum of interaction dimensions. Thus, we explicitly model the interaction between pairwise nodes, where any interaction $z_{(i,j)}$ points to an inherent edge, and $Z = \{z_{(i,j)}, \cdots\}, i, j \in V$, can also be represented as $Z = \{z_{(k)}, \cdots\}, k \in E$.

4.3. The degree-aware neighborhood transfer

As head nodes are structurally rich, we assume their observed neighborhoods are representative enough to be regarded as the ideal neighborhoods. Therefore, the interaction between the head nodes and their neighbors often provides the knowledge that the tail node desires. Thus, we exploit and learn the feature interaction from the neighborhood of head nodes, and then locally transfer the knowledge of interaction to tail nodes in order to predict the missing neighborhood information.

Considering the inevitable existence of redundant information in the constructed interaction information, we first compress the adaptive interaction through a unified feature encoder to predict the missing neighborhood information of low-level nodes as much as possible. Its purpose is to enhance the distinguishability of the node embeddings by introducing the difference information between the node and its neighbors. If the node differs greatly from its neighbors (high AvgDist), its characteristics will be adjusted to reflect this local heterogeneity; otherwise, more original information will be retained. The unified feature encoder can be represented as:

$$\tilde{z}_{(i)} = \left(z_{(i)} + \sqrt{\sum_{v_k \in \mathcal{N}(i)} (z_{(i)} - z_{(k)})^2} / |\mathcal{N}(i)|\right) W_u, \tag{4.5}$$

where $|\mathcal{N}(i)|$ is the number of neighbors of node v_i. $(z_{(j)} - z_{(k)})^2$ represents the difference information (high-level information) between nodes, explaining implicit high-level relations. W_u is the weight matrix obtained by MLP for the new embedding.

Our tail-oriented transfer strategy aims to enable tail nodes to receive interaction from high-resource nodes of the same class but with significantly different degrees as much as possible. Based on the node adjacency matrix $A \in \mathbb{R}^{N \times N}$, we initialize the tail-oriented transition matrix $T \in \mathbb{R}^{N \times M}$ as the node edge

adjacency matrix. The transfer process can be expressed as:

$$h'_i = h_i + \sum_{k=1}^{N} T_{(i,k)} z_{(k)}, \tag{4.6}$$

where h'_i is the enhanced node embedding of node v_i, and $T_{(i,k)} = 1$ indicates that node v_i receives adaptive interaction $z_{(k)}$. When T is the node-edge adjacency matrix, any node can only accept the interaction information of its nearest neighbors. Therefore, we need to update the transfer matrix A_t according to the goal of knowledge transfer. The process of updating the transfer matrix A_t can be expressed as follows:

$$P_{(i,j)} = \begin{cases} 1, if \left| \sum_{k=1}^{N} A_{(i,k)} - \sum_{k=1}^{N} A_{(j,k)} \right| > \frac{1}{N} \sum_{k=1}^{N} \sum_{r=1}^{N} A_{(k,r)} \\ 0, otherwise \end{cases} \tag{4.7}$$

$$A_t = HH^T + P, \tag{4.8}$$

where $\left| \sum_{k=1}^{N} A_{(i,k)} - \sum_{k=1}^{N} A_{(j,k)} \right|$ represents the degree difference between nodes v_i and v_j. $\frac{1}{N} \sum_{k=1}^{N} \sum_{r=1}^{N} A_{(k,r)}$ denotes the average degree, and P is used to connect paired nodes with significantly different degrees. The transfer process toward the tail node can be expressed as:

$$H' = H + A_t T Z, \tag{4.9}$$

where $H' = \{h'_1, \cdots, h'_N\}$ is the final node embedding.

4.4. Task-dependent loss functions

For semi-supervised classification tasks, the loss fuction $\mathscr{L}$ can be represented as follows:

$$\mathscr{L}(\Theta) = -\sum_{v_i \in V} \sum_{j}^{C} Y_{ij} \log(\widetilde{H'}_{ij}), \tag{4.10}$$

where $\widetilde{H'}$ is the softmax result of H', and $\Theta = (W_v, W_e, W_t, S)$ is the parameter set, which is also used in link prediction.

To obtain more accurate node embeddings, we leverage adaptive interaction Z to construct the auxiliary classifiers. Specifically, we employ an additional graph convolutional layer that encodes adaptive interaction Z into embeddings H_z. Eventually, the overall objective function is the weighted sum of the three losses:

$$\mathscr{L}_{node} = \beta_1 \mathscr{L}(H', Y) + \beta_2 \mathscr{L}(H_z, Y), \tag{4.11}$$

where β_1, β_2 are the hyperparameters that control the proportion of loss functions to minimize the total loss $\mathscr{L}_{node}$.

Finally, we optimize the model parameters with respect to the objective using stochastic gradient descent. Without loss of generality, the algorithm of GraphDAFI is elaborated as follows (Algorithm 1).

Algorithm 1: GraphDAFI

Input: Original graph $G = (V, E)$; Feature matrix X; Iterations *epochs*; Hyperparameters β_1, β_2;

Output: The node embedding H';

Obtain the transfer matrix T via the original graph G;

for *epochs* **do**

Encode feature matrix X to obtain node embedding H via Eq (4.1);

Construct adaptive interaction of pairwise nodes Z via Eq (4.4);

Obtain enhanced interaction $\tilde{Z}$ by unified feature encoder via Eq. (4.5);

Aggregate adaptive interaction $\tilde{Z}$ into node embedding H' via Eq (4.6);

Update transfer matrix T via Eq (4.7) and Eq (4.8);

Calculate the corresponding loss and update the parameter set via Eq (4.11);

end

Return The node embedding H';

4.5. Complexity analysis

Let $G = (V, E)$ be the input graph, $|V| = N$ denotes the number of nodes, and $|E| = M$ is the number of edges. For each node v_i, d_0 represents the dimension of node features, and $\{d_1, ..., d_l\}$ denotes the hidden layer dimension of the GCN learner with l layers. Consequently, the complexity of ith layer in GCN is $O(Nd_id_{i-1} + Md_i)$. The complexity of ith layer in Demo-Net is $O(Nd_id_{i-1} + Td_i + NHd_{i-1} + Md_i)$, where T is the number of tasks (degree values) and H is the hashing dimension in the graph. The complexity of ith layer in UMGCN is $O(Nd_id_{i-1} + d'd'' + N^3 + (Nd_i + N^2d_{i-1}) + |V_L| \cdot |V_U|)$, where $|V_L|, |V_U|$ represent the number of labeled and unlabeled nodes and $\{d', d''\}$ are the hidden layer dimensions of this attention mechanism. For our proposed model GraphDAFI, its complexity can be composed of the following parts:

- In the adaptive feature interaction, the complexity of this module is $O(Md_i^2 d_{i-1})$, where M denotes we construct the interaction information $Z \in \mathbb{R}^{M \times d'}$ of pairwise nodes.
- In the degree-aware neighborhood transfer module, it involves the unified feature encoder, transfer process, and adjacency matrix update. Therefore, their time complexities are as follows: $O(Md_i + Nd_i + Nd_id_{i-1}), O(N^2), O(N^2 + Md')$.

Therefore, the computational complexity of the entire GraphDAFI model is $O((M + N)d_i + (Md_i{}^2 + Nd_i)d_{i-1} + 2N^2)$.

4.6. Scalability implementation

To address the computational challenges posed by pairwise interactions in large-scale graphs, we propose a scalable solution using METIS-based graph partitioning acceleration. METIS is a well-established graph partitioning algorithm that decomposes a graph into smaller subgraphs while minimizing the edge cut, thereby reducing inter-partition communication overhead.

Graph partitioning formulation. Given a graph $G = (V, E)$ with $|V| = N$ nodes and $|E| = M$

edges, we employ METIS to partition the graph into K disjoint subgraphs $\{G_1, G_2, \ldots, G_K\}$, where:

$$G_i = (V_i, E_i), \quad \bigcup_{i=1}^{K} V_i = V, \quad V_i \cap V_j = \emptyset \text{ for } i \neq j, \tag{4.12}$$

where V_i and E_i denote the node set and edge set of partition i, respectively. The partitioning objective is to minimize the edge cut:

$$\text{EdgeCut} = \left|\{(v_i, v_j) \in E : v_i \in V_p, v_j \in V_q, p \neq q\}\right|, \tag{4.13}$$

while maintaining balanced partition sizes: $|V_i| \approx \frac{N}{K}$ for all $i \in \{1, \ldots, K\}$.

Intra-partition interaction computation. For each partition G_i, we compute the adaptive node interactions locally using Eq 4.4. The interaction representation within partition i is:

$$Z_i = \{z^{l+1}_{(u,v)} : (u, v) \in E_i\}, \quad z^{l+1}_{(u,v)} = \sigma\left(\left[z^{l}_{(u,v)} \oplus (h^{l+1}_u \odot h^{l+1}_v)\right] S^T\right), \tag{4.14}$$

where all nodes $u, v \in V_i$ belong to the same partition, ensuring that strongly connected nodes are processed together.

Inter-partition interaction handling. For cross-partition edges (u, v) where $u \in V_p$ and $v \in V_q$ with $p \neq q$, we approximate the interaction by:

$$z^{l+1}_{(u,v)} = \sigma\left((h^{l+1}_u \odot h^{l+1}_v) S^T_{cross}\right), \tag{4.15}$$

where $S_{cross} \in \mathbb{R}^{d \times d'}$ is a simplified learnable matrix that avoids the residual connection from previous layers, thereby reducing communication overhead between partitions.

Complexity analysis. Without partitioning, computing adaptive interactions for all edges requires $O(Md^2)$ operations, where M is the total number of edges and d is the embedding dimension. With METIS partitioning into K subgraphs, the complexity becomes:

$$O\left(\sum_{i=1}^{K} (M_i d^2 + N_i \log N_i)\right), \tag{4.16}$$

where $M_i = |E_i|$ and $N_i = |V_i|$ represent the number of edges and nodes in partition i, respectively. The $N_i \log N_i$ term accounts for the partitioning overhead. Since $\sum_{i=1}^{K} M_i \ll M$ due to minimized edge cuts, and partitions can be processed in parallel, this approach achieves significant speedup:

$$\text{Speedup} \approx \frac{Md^2}{\max_i (M_i d^2 + N_i \log N_i)}, \tag{4.17}$$

where the denominator represents the bottleneck partition's computational cost under parallel processing.

5. Experiments

In this section, we evaluate GraphDAFI against state-of-the-art semi-supervised node classification models, focusing on its ability to address the long-tail distribution problem. Additionally, we conduct auxiliary experiments to assess the performance of the individual components of GraphDAFI.

5.1. Datasets

The experiments are conducted over four real-world datasets which are summarized in Table 2.

1) Citeseer: It is a literature citation network dataset, which can automatically extract citation information of the literature and establish citation networks among the literature.

2) Cora: It is an academic paper citation network, mainly used to provide a testing ground for algorithm research. This dataset contains approximately 2708 machine learning papers, each of which has been manually labeled as one of seven categories: Case Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Rule Learning, and Theory.

3) ACM: It is a dataset mainly used in heterogeneous GNNs, which constructs a heterogeneous information network including various nodes such as Paper, Author, Affiliation, Venue, and multiple relationships.

4) Pubmed: It is a dataset widely used in the biomedical field, containing the abstract texts of approximately 19,717 research papers related to diabetes as graph nodes.

Table 2. Data details.

Datasets	Nodes	Edges	Classes	Features	Training	Test
Cora	2708	5429	7	1433	14/28/140	1000
Citeseer	3327	4732	6	3703	15/30/160	1000
Pubmed	19,717	44,338	3	500	15/30/60	1000
ACM	3025	26,256	3	1870	15/30/60	1000

5.2. Baselines

To comprehensively evaluate the performance of our model, we conducted a comparison with several different types of models as detailed below:

1) Representative models: GCN [31], GAT [32]. The former treats all nodes as equally important without distinguishing between head and tail nodes.The latter assigns different weights to nodes using an attention mechanism, but the information propagation limitations for tail nodes hinder sufficient feature learning.

2) Feature interaction models: CensNet [26] and GraphAIR [19]. They are generally capable of optimizing graph structures and feature representations; however, they do not specifically address the issue of long-tail nodes. In scenarios characterized by significant long-tail distributions, models tend to pay insufficient attention to low-degree long-tail nodes, resulting in suboptimal learning outcomes.

3) Degree-based models: Demo-Net [25], Tail-GNN [15], and SL-DSGCN [27]. They address the issue of long-tail distribution with the aim of enhancing the representational capability of low-degree nodes. By employing a recalibration mechanism to mitigate bias, learning and transferring neighborhood relationships of high-degree nodes, and designing degree-specific layers, they effectively optimize the feature representation of long-tail nodes. This approach significantly improves performance in sparse and imbalanced graph data.

4) SOTA models: UMGCN [33], Logo-GNN [34], and FTCP [35] are the latest topology augmentation-related models in graph representation learning. Among them, Logo-GNN and UMGCN both adopt the multi-channel framework to merge the information of the augmented graph with the original topology.

5.3. Parameter setting

Table 2 gives the basic information of relevant datasets, and explains the allocation ratio of the training, verification, and test set. In the proposed GraphDAFI, we employed full batch processing to train our model and optimized it using the Adam algorithm. The parameter settings are as follows: the learning rate is set between 0.001 and 0.005; hyperparameters β_1 and β_2 are searched within the range $\{0, 0.1, ..., 1\}$; the number of training epochs ranges from 100 to 500; weight decay is set at 0.0005. For the baselines, the results of GCN, GAT, Dome-net, SL-DSGCN, and UMGCN were copied from previous publications [33]. Tail-GCN, CensNet, and GraphAIR adopt the two-layer GCN architecture and the embedding dimension is selected from [512, 256, 128]. In particular, the GraphAIR's parameters of the overall objective function are set to [1, 1, 1]. Tail-GCN sets the default degree threshold to $K = 5$, i.e., nodes with degree no greater than 5 are regarded as tail nodes. The experiments are run 10 times in the same partition, and then the average results are reported for comparison and validation.

5.4. Semi-supervised node classification

Table 3 presents the accuracy values (ACC) and F1 score obtained after training, following a detailed discussion and comparison of various algorithms in semi-supervised node classification. The results that performed best are highlighted in bold. From the Table 3, it can be observed that:

1) Compared to the baselines, GraphDAFI achieves superior performance on most datasets. This can be attributed to the effective collaboration between adaptive feature interaction and degree-aware neighborhood transfer. The devised modules not only extract discriminative information but also integrate this into the final node embedding to enhance the overall classification performance.

2) Compared with long-tail-related models, GraphDAFI exhibits strong competitiveness over most datasets. It is worth noting that GraphDAFI adopts the interaction information as the local perspective to capture auxiliary knowledge. In addition, the improvements indicate that degree-aware neighborhood transfer perceives the nonlinear dependencies between source and target domains, facilitating the exchange of high-level information.

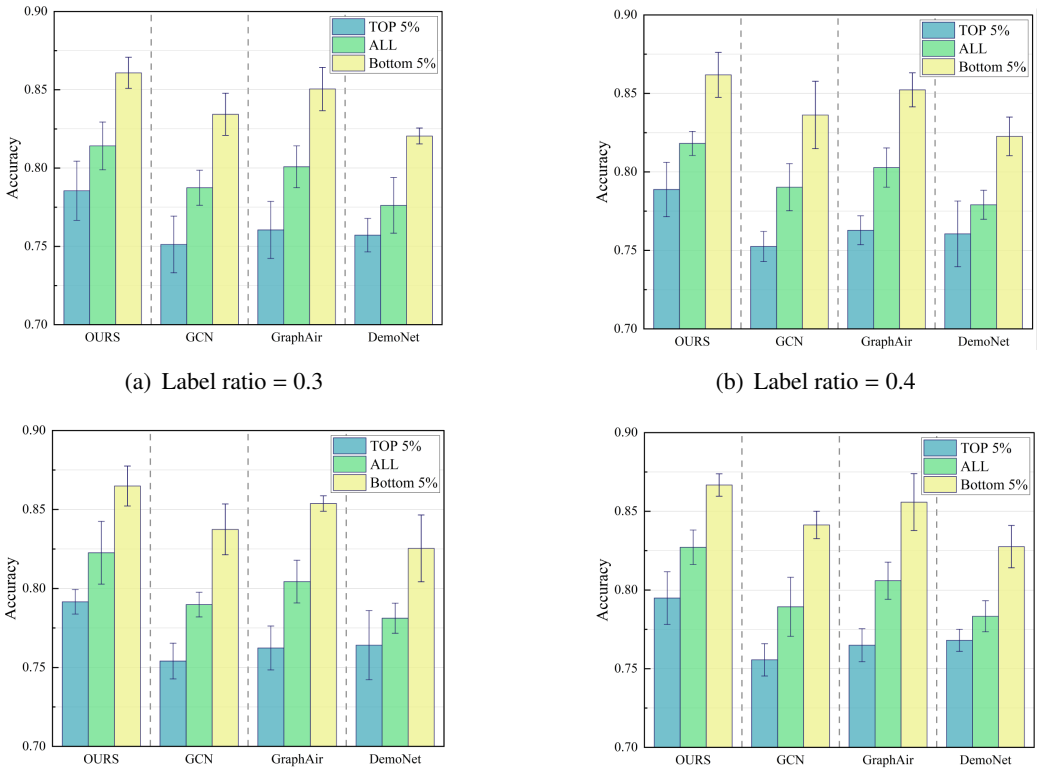
3) From the failure cases presented in Table 3, using the Citeseer dataset as an example, it is evident that the performance of GraphDAFI was inferior to that of UMGCN. This discrepancy can be attributed to UMGCN's ability to extract high-quality node representations, effectively compensate for information loss associated with low-degree nodes, and maintain stable model performance in complex networks. Consequently, it is essential to enhance GraphDAFI's feature aggregation capabilities within scenarios characterized by sparse graph structures and intricate original topological relationships.

Table 3. Comparison of models on different datasets.

Datasets	Cora		Citeseer		ACM		Pubmed	
Metrics	ACC↑	F1↑	ACC↑	F1↑	ACC↑	F1↑	ACC↑	F1↑
GCN [31]	74.87	71.50	67.80	63.63	78.50	76.08	76.51	75.60
GAT [32]	78.14	77.88	69.24	65.58	85.60	84.23	77.11	75.82
Tail-GCN [15]	83.12	81.32	72.22	69.34	88.03	87.08	79.48	77.87
Demo-net [25]	78.24	77.60	61.44	59.02	90.13	90.02	75.96	74.28
SL-DSGCN [27]	83.43	83.03	65.49	62.47	89.09	87.23	80.93	80.02
CensNet [26]	79.14	78.45	67.51	66.23	89.72	87.45	69.94	69.24
GraphAIR [19]	84.22	81.78	71.19	68.73	90.24	89.66	80.12	77.45
Logo-GNN [34]	84.16	81.80	71.88	69.56	90.64	88.78	80.62	78.31
FTCP [35]	83.31	80.80	72.65	69.43	91.24	89.13	81.67	79.45
UMGCN [33]	82.46	81.80	**72.93**	69.34	90.44	89.69	80.27	78.24
GraphDAFI	**85.33**	**82.39**	70.23	**69.85**	**91.56**	**90.49**	**81.40**	**80.82**

5.5. Comparison of the ability to promote the long-tail issue

In order to verify the ability of the model to alleviate the long-tail distribution, we sample the top 5% and the bottom 5% of the dataset, respectively, as the head and tail nodes. We select the train environment with different label ratios for full implementation, and the results are shown in Figure 3.



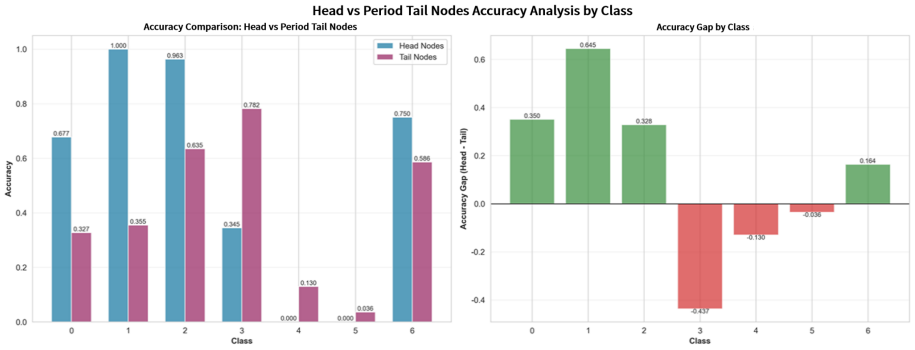
(a) Label ratio = 0.3

(b) Label ratio = 0.4

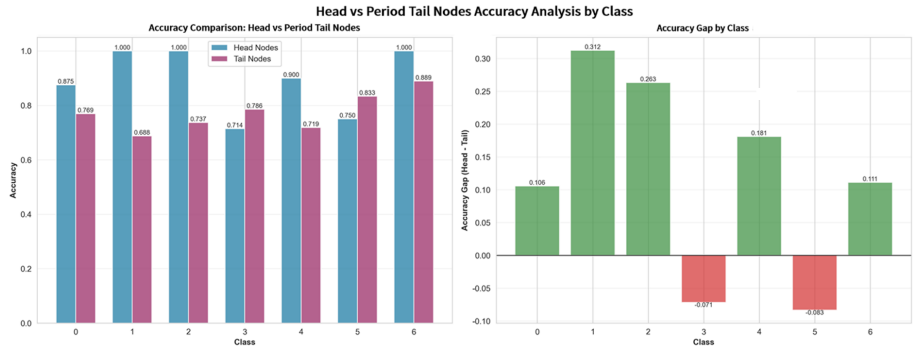
(c) Label ratio = 0.5

(d) Label ratio = 0.6

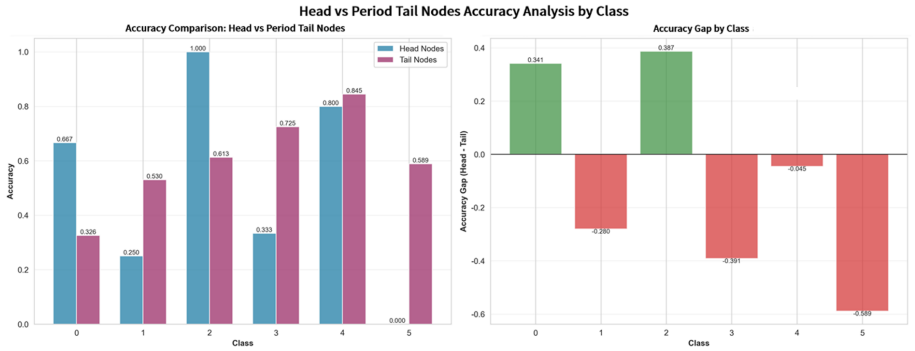
Figure 3. Visualization comparison of head vs. tail nodes on Cora.



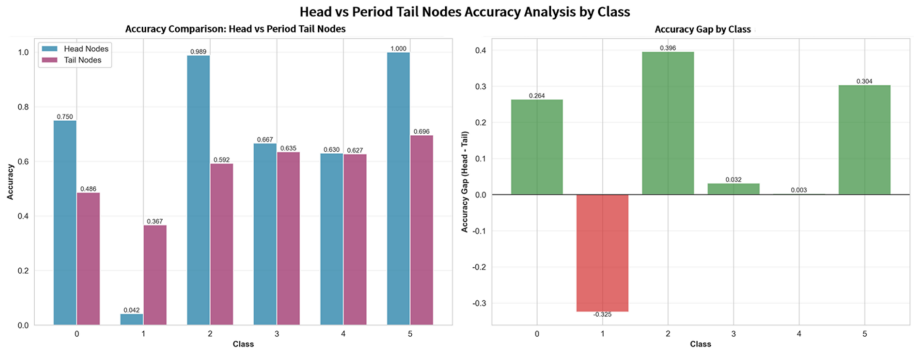
(a) Per-class accuracy of GCN on the Cora



(b) Per-class accuracy of GraphDAFI on the Cora



(c) Per-class accuracy of GCN on the Citeseer



(d) Per-class accuracy of GraphDAFI on the Citeseer

Figure 4. Visualization comparison of per-class or tail vs. head accuracy.

It is evident from empirical results that GraphDAFI demonstrates significant advantages in tail node classification. This suggests that the degree-aware neighborhood transfer effectively leverages auxiliary knowledge from head nodes and utilizes constructed interactive information. Comparing DemoNet with GCN and GarphAir, we can find that although DemoNet's strategy for long-tail distribution is effective, it has lost the overall classification performance to some extent.

From Figure 4, the proposed GraphDAFI model demonstrates superior performance over the baseline GCN in long-tail node classification. It achieves a more balanced accuracy distribution across head and tail classes: for example, maintaining high accuracy in head classes (e.g., 1.0 in Class 1) while significantly improving tail class recognition, such as increasing tail accuracy in Class 1 from 0.6875 to 0.8352. Additionally, the model generalizes well across classes of varying sizes, sustaining robust accuracy in both large and small classes. These results confirm that GraphDAFI effectively mitigates class imbalance in GNNs, delivering more balanced, stable, and accurate classification.

5.6. Scalability on the large graph

To evaluate the scalability of our GraphDAFI framework, we conducted extensive experiments on the ogbn-arxiv dataset, a substantial citation network comprising 169,343 nodes and 1,166,243 edges. The core objective was to assess the computational efficiency of our adaptive interaction mechanism when handling large-scale graphs. Specifically, we measured the additional processing time required to compute the pairwise node interactions and the corresponding GPU memory allocation during this preprocessing phase.

As shown in Table 4 and Figure 5, memory allocation refers to the GPU memory required to store the preprocessed graph, while partition rate denotes the proportion of subgraphs modified by GraphDAFI during preprocessing. GraphDAFI achieves an effective balance between performance enhancement and computational overhead. The adaptive interaction module introduces a manageable increase in processing time while significantly improving the model's representation capability. More importantly, as demonstrated in our analysis, the implemented optimization strategies effectively control memory consumption, enabling GraphDAFI to scale efficiently to graphs of this magnitude. This demonstrates the practical viability of our approach for real-world, large-scale graph learning tasks.

Table 4. Scalability on ogbn-arxiv.

Partition rate	Remove edges	Acc	Preprocessing time(s)	Training & testing time(s)
0	–	OOM	–	–
0.10%	89,455	0.66	176	492
0.50%	187,221	0.67	191	511
1.00%	298,777	0.71	263	531
5.00%	423,105	0.72	350	543

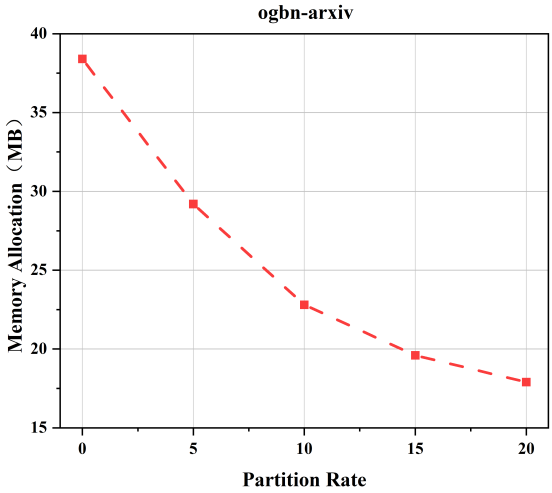


Figure 5. Memory allocation for graphs after preprocessing with different partition rates.

5.7. Ablation study

This section presents an ablation study focused on validating the contributions of various components of GraphDAFI to the model performance, as well as their synergistic interactions. To this end, we systematically remove model components either individually or in combination and evaluate the performance of GraphDAFI and its variants. Specifically, we analyze how the node representation in GraphDAFI is integrated from three components: adaptive feature interaction (AFI), universal feature encoding (UFE), and degree-aware transfer integration (NTI). The variants are represented by removing single or paired components; for instance, GraphDAFI-UN indicates that AFI has been removed from GraphDAFI, while GraphDAFI-A signifies that UFE and NTI have been excluded.

The final classification results are presented in Table 5 and Figure 6, revealing several key observations:

1) Contribution of individual components: It is evident that GraphDAFI-UN, GraphDAFI-AN, and GraphDAFI-AU perform exceptionally well on the ACM dataset. Notably, the performance of GraphDAFI-AU surpasses that of GraphDAFI-UN; however, the performance of GraphDAFI-AN significantly declines compared to both of the other variants. Analyzing these outcomes reveals that NTI effectively leverages node degree information for deep modeling in graph data with pronounced domain distribution differences, thereby reducing feature bias across domains. AFI enhances global feature learning capabilities when dynamically capturing edge-node feature interactions which improves model robustness. UFE contributes to the foundational classification ability by providing universal feature representations.

2) Performance of paired components. It can be observed that the combination of AFI, UFE, and NTI can complement each other and improve the performance of the model in the graph classification task. Specifically, GraphDAFI-A is suitable for datasets with complex distribution and dynamic relationships. GraphDAFI-U has strong domain adaptability and is very suitable for graphs with significant distribution differences.

Table 5. ACC (%) of node classification with different components.

	AFI	UFE	NTI	Cora	Citeseer	Pubmed	ACM
GraphDAFI-UN	✓			82.2	67.5	79.1	88.2
GraphDAFI-AN		✓		75.3	59.4	70.2	79.3
GraphDAFI-AU			✓	82.6	68.6	80.3	89.4
GraphDAFI-A		✓	✓	83.1	69.3	80.6	89.7
GraphDAFI-U	✓		✓	83.5	69.2	80.7	87.5
GraphDAFI-N	✓	✓		82.9	69.5	80.4	87.8
GraphDAFI	✓	✓	✓	**84.3**	**70.2**	**81.4**	**90.5**

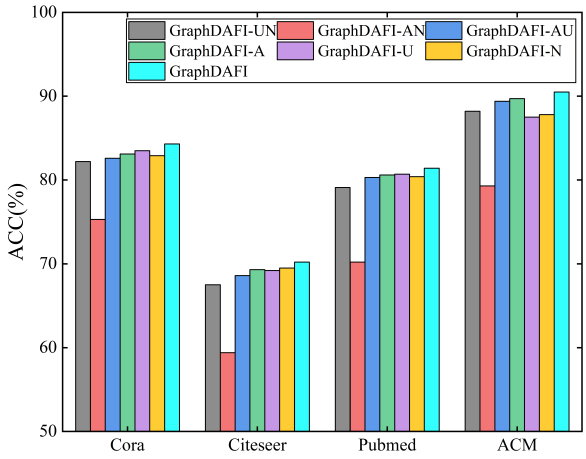


Figure 6. ACC (%) of GraphDAFI and its variants.

5.8. Parameter sensitivity

The analysis of the hyperparameters in the loss function reveals the operational mechanism of our model. In this section, we further investigate the regularities regarding how the hyperparameters β_1 and β_2 influence node classification on the Cora dataset. Specifically, the hyperparameters β_1 and β_2 take values from $\{0.6, 0.7, ..., 1.4\}$. We present the corresponding bar charts illustrating these effects from the perspective of node classification accuracy (ACC), as shown in Figure 7.

In the Figure 7, it is evident that the two hyperparameters exhibit a significant nonlinear relationship with classification accuracy, characterized by a certain degree of fluctuation. The variation range lies between 0.7 and 0.8. From an overall distribution perspective, higher classification accuracy predominantly concentrates within the mid-to-high value range, particularly in the region where β_1 and β_2 are approximately between 0.8 and 1.2. This phenomenon can largely be attributed to the model's ability to effectively capture feature interactions among nodes while maintaining a balanced optimization of loss terms, thereby avoiding both overfitting and underfitting. The results indicate that moderate parameter settings can better balance the weights of various components within the loss function, thus significantly enhancing model performance.

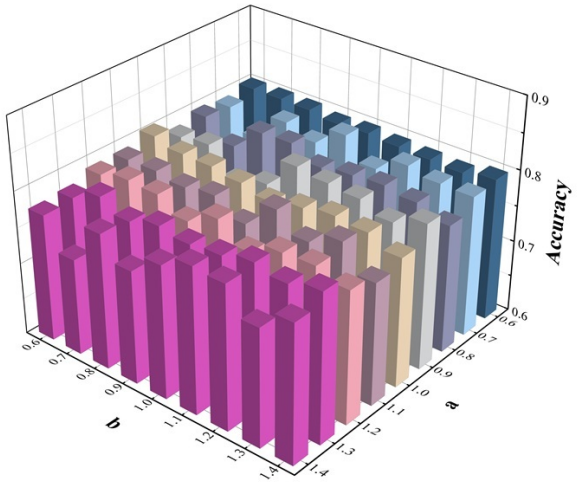
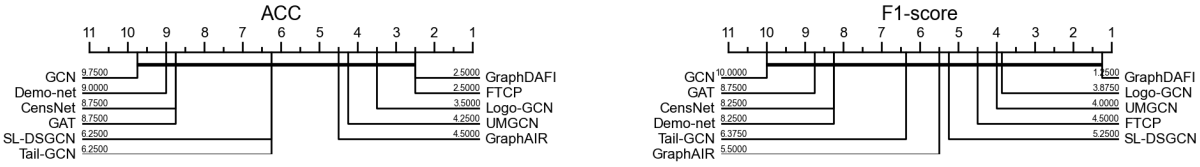


Figure 7. ACC of node classification on Cora with varying hyperparameters.

Furthermore, when there is a substantial disparity in the values of β_1 and β_2, such as when $\beta_1 = 0.7$ and $\beta_2 = 1.4$, there is a marked decline in classification accuracy. This outcome suggests that these two hyperparameters may possess some degree of coupling within the model; an imbalance in their weight distribution could hinder effective coordination between node feature learning and structural information processing. Further analysis reveals that larger parameter values may introduce additional noise elements, diminishing modeling capability for graph structure information; conversely, smaller parameter values might result in insufficient weighting, failing to adequately optimize specific loss components.

5.9. Statistical analysis

In order to systematically analyze the performance of the above comparison methods, it is necessary to consider using statistical tests for analysis. Specifically, we use the Bonferroni-Dunn test to generate CD (critical difference) diagrams of the two performance indicators, as shown in the Figure 8.



(a) CD diagram of accuracy

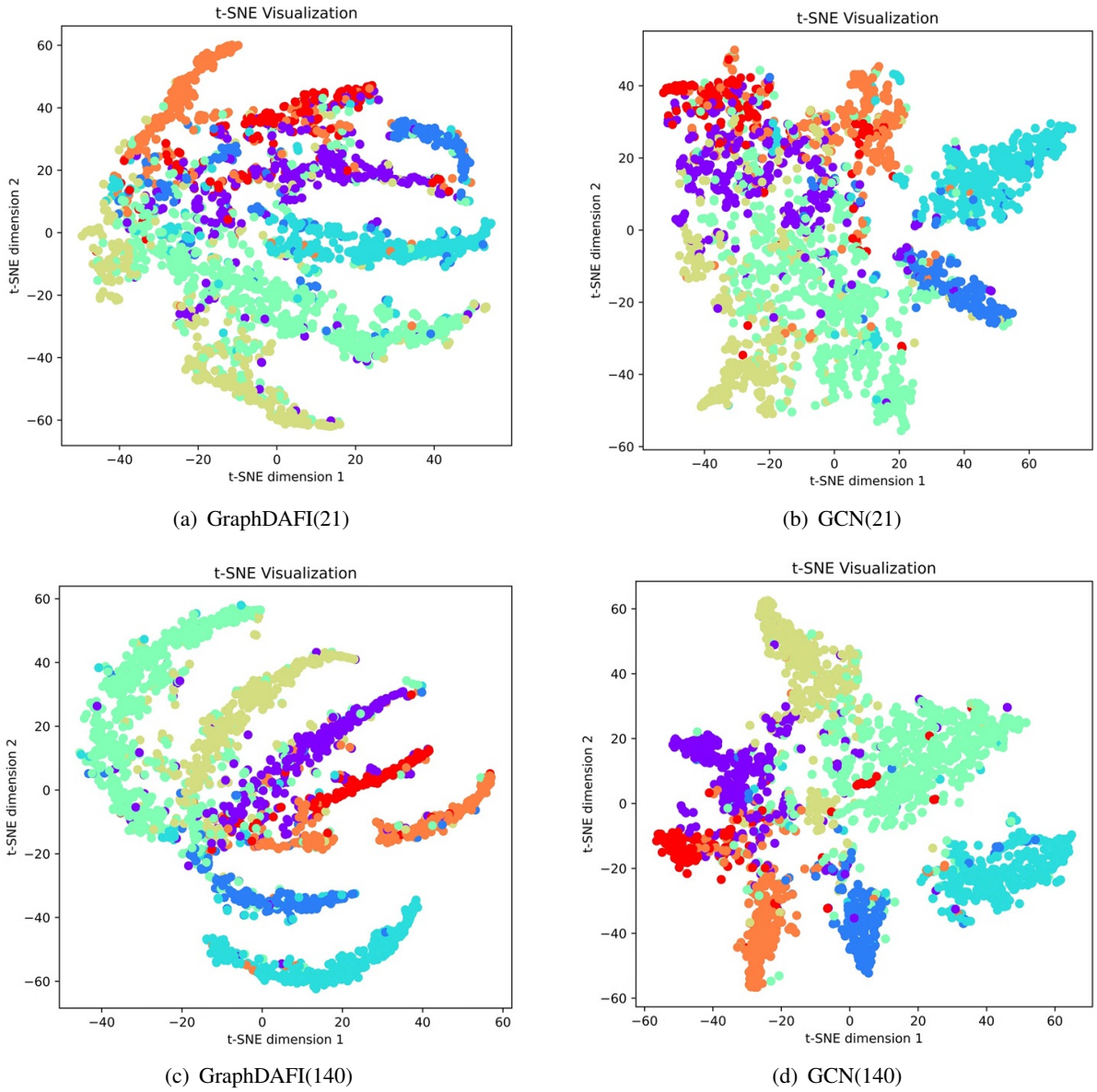
(b) CD diagram of F1-score

Figure 8. The CD diagrams on all datasets. (The line of the order number indicates the ranking of method performance. Bold lines indicate within the CD threshold.)

Figure 8 shows that GraphDAFI achieved the top ranking in both ACC and F1 indicators, owing to its superior performance. Compared with models related to long-tail distribution, GraphDAFI outperforms Demo-net, SL-DSGCN, and Tail-GCN, indicating that GraphDAFI not only solves the problem of tail nodes' poor performance but also improves the overall performance.

5.10. Visualization

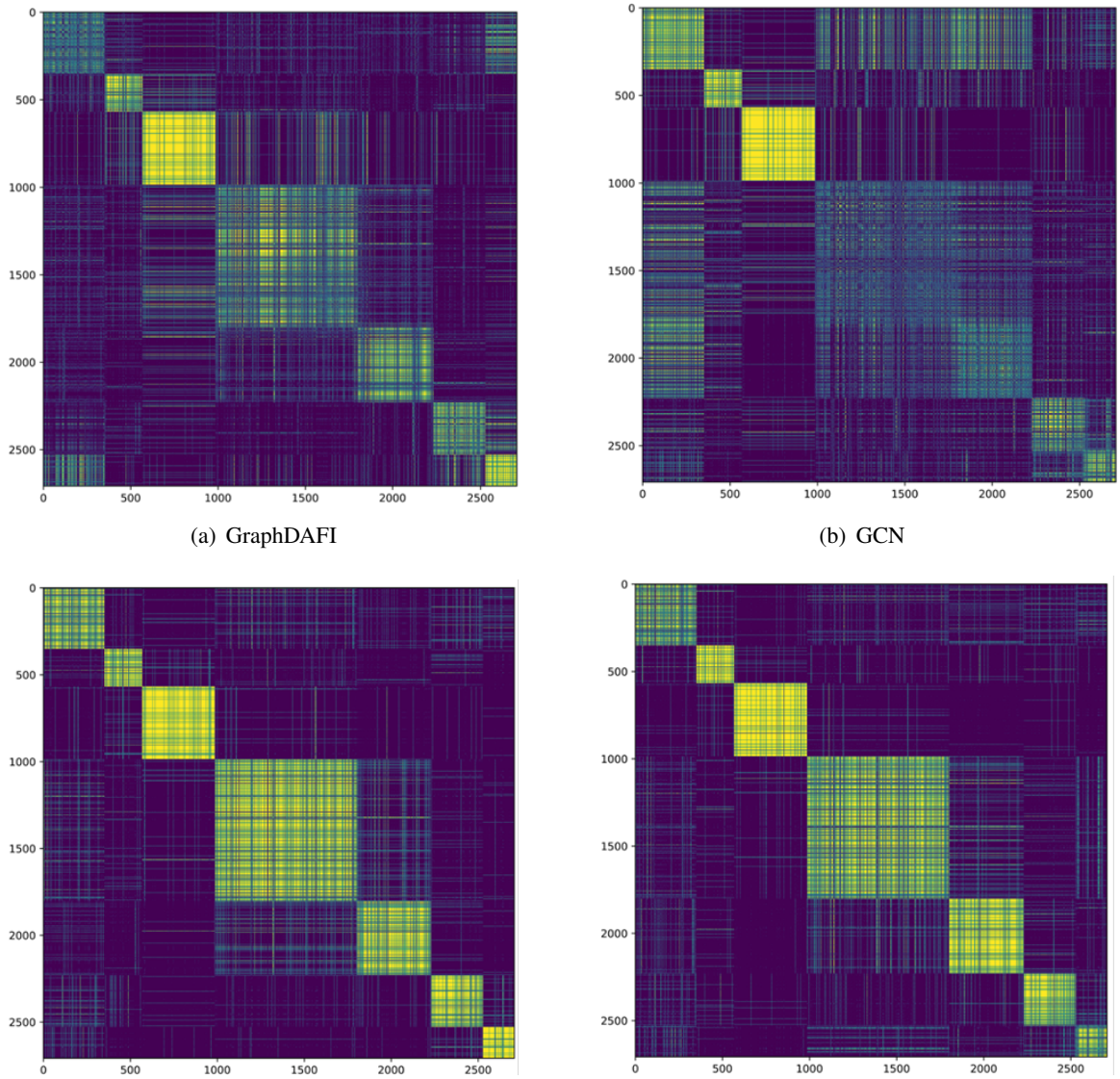
In this subsection, we conduct a comparative analysis of the performance of GraphDAFI and GCN models on the Cora dataset through visualization techniques. The choice to compare with GCN is motivated by the differing approaches both models employ in node representation learning and feature interaction modeling. Specifically, while GraphDAFI dynamically captures the interactions between nodes and edges to enhance intra-class feature consistency and inter-class feature separability, IGCN relies on static convolution that aggregates features based on a fixed graph structure. Therefore, we employed t-SNE and similarity matrix $\widetilde{Z}^{agg}\widetilde{Z}^{agg^T}$ visualizations to comprehensively illustrate the distribution characteristics and quality of node representations $\widetilde{Z}^{agg}$, as shown in Figures 9 and 10.



(a) GraphDAFI(21) (b) GCN(21)

(c) GraphDAFI(140) (d) GCN(140)

Figure 9. Visualization comparison of node representation distribution on Cora.



(a) GraphDAFI (b) GCN

(c) GraphDAFI (d) GCN

Figure 10. Visualization comparison of node representation quality on Cora.

From Figure 9, it can be observed that under the low-sample condition with a training sample size of 21 (e.g., Figure 9(a)), our model effectively clusters similar samples together and demonstrates clearer boundaries compared to GCN. In the high-sample condition with a training sample size of 140 (e.g., Figure 9(c)), both models show an overall improvement in node distribution; however, our model exhibits superior embeddings and higher intra-class similarity.

The similarity matrix presented in Figure 10 further corroborates this conclusion. Under low-sample conditions, our model displays complete diagonal matrix blocks, indicating high intra-class node similarity and consistent distribution of embedding features across categories. In contrast, the diagonal block distribution for GCN is more scattered and noticeably darker, suggesting weaker intra-class similarity and difficulty in forming accurate class representations. Under

high-sample conditions, both models demonstrate enhanced quality in overall node representations; however, GraphDAFI further reinforces the diagonal block structure, resulting in brighter corresponding category regions and stronger intra-class consistency.

6. Conclusions and future work

In this study, we introduce GraphDAFI, a novel framework addressing the long-tail degree distribution problem in GNNs via Taylor interaction theory. Its key innovations are: 1) an adaptive feature interaction module that automatically learns potential graph dependencies by fusing node embeddings with factorization machine-based interaction features; 2) a degree-aware neighborhood transfer mechanism that updates the node-edge adjacency matrix through a tail-focused strategy to transfer interaction information to low-degree nodes without altering the original topology; and 3) a direct enhancement of interaction effects for tail nodes, eliminating noise injection from high-degree nodes and significantly improving the robustness of tail-node representations. Unlike graph augmentation approaches, GraphDAFI preserves structural integrity while effectively bridging the embedding gap between head and tail nodes.

Although this study has made significant progress in different downstream tasks, there are still many issues worthy of further investigation. For example, in the process of injecting auxiliary information into low-degree nodes, although knowledge transfer effectively enhances the expression ability of low-degree node embeddings, the overall robustness of the GNN framework cannot be guaranteed. In future work, we will further explore the robustness of models in imbalanced learning based on knowledge transfer. In addition, due to the need to construct interaction information between pairwise nodes, the model is limited by computational complexity when dealing with large-scale dense graphs. The lower performance improvement on dense graphs in recommendation systems also reflects the difficulty of GraphDAFI in handling dense graphs. Therefore, in future work, we will attempt to use acceleration methods such as vector quantization to simplify the model and improve its scalability.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work is supported by National Natural Science Foundation of China (62076111).

Conflict of interest

The authors declare there are no conflicts of interest.

References

1. S. Khoshraftar, A. An, A survey on graph representation learning methods, *ACM Trans. Intell. Syst. Technol.*, **15** (2024), 1–55. https://doi.org/10.1145/3633518

2. K. Sharma, Y. Lee, S. Nambi, A. Salian, S. Shah, S. Kim, et al., A survey of graph neural networks for social recommender systems, *ACM Comput. Surv.*, **56** (2024), 1–34. https://doi.org/10.1145/3661821

3. M. Réau, N. Renaud, L. C. Xue, A. M. J. J. Bonvin, DeepRank-GNN: A graph neural network framework to learn patterns in protein–protein interfaces, *Bioinformatics*, **39** (2023), 759. https://doi.org/10.1093/bioinformatics/btac759

4. F. Li, J. Feng, H. Yan, G. Jin, F. Yang, F. Sun, et al., Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution, *ACM Trans. Knowl. Discovery Data*, **17** (2023), 1–21. https://doi.org/10.1145/3532611

5. X. Pei, X. Deng, N. N. Xiong, S. Mumtaz, J. Wu, Complex graph analysis and representation learning: Problems, techniques, and applications, *IEEE Trans. Network Sci. Eng.*, **11** (2024), 4990–5007. https://doi.org/10.1109/TNSE.2024.3417850

6. Q. Guo, X. Yang, F. Zhang, T. Xu, Perturbation-augmented graph convolutional networks: A graph contrastive learning architecture for effective node classification tasks, *Eng. Appl. Artif. Intell.* , **129** (2024), 107616. https://doi.org/10.1016/j.engappai.2023.107616

7. Y. Ye, S. Ji, Sparse graph attention networks, *IEEE Trans. Knowl. Data Eng.*, **35** (2023), 905–916. https://doi.org/10.1109/TKDE.2021.3072345

8. F. Wu, T. Zhang, A. H. de Souza Jr., C. Fifty, T. Yu, K. Q. Weinberger, Simplifying graph convolutional networks, preprint, arXiv:1902.07153.

9. S. Yi, Z. Mao, W. Ju, Y. Zhou, L. Liu, X. Luo, Towards long-tailed recognition for graph classification via collaborative experts, *IEEE Trans. Big Data*, **9** (2023), 1683–1696. https://doi.org/10.1109/TBDATA.2023.3313029

10. X. Wang, X. Yang, P. Wang, H. Yu, T. Xu, SSGCN: A sampling sequential guided graph convolutional network, *Int. J. Mach. Learn. Cybern.*, **15** (2024), 2023–2038. https://doi.org/10.1007/s13042-023-02013-2

11. Z. Liu, W. Zhang, Y. Fang, X. Zhang, S. C. H. Hoi, Towards locality-aware meta-learning of tail node embeddings on networks, in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, (2020), 975–984. https://doi.org/10.1145/3340531.3411910

12. E. M. Hamedani, M. Kaedi, Recommending the long tail items through personalized diversification, *Knowl.-Based Syst.*, **164** (2019), 348–357. https://doi.org/10.1016/j.knosys.2018.11.004

13. K. Yao, J. Liang, J. Liang, M. Li, F. Cao, Multi-view graph convolutional networks with attention mechanism, *Artif. Intell.*, **307** (2022), 103708. https://doi.org/10.1016/j.artint.2022.103708

14. J. Lin, Y. Wan, J. Xu, X. Qi, Long-tailed graph neural networks via graph structure learning for node classification, *Appl. Intell.*, **53** (2023), 20206–20222. https://doi.org/10.1007/s10489-023-04534-3

15. Z. Liu, T. Nguyen, Y. Fang, Tail-GNN: Tail-node graph neural networks, in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, (2021), 1109–1119. https://doi.org/10.1145/3447548.3467276

16. S. Zhu, C. Zhou, S. Pan, X. Zhu, B. Wang, Relation structure-aware heterogeneous graph neural network, in *2019 IEEE International Conference on Data Mining (ICDM)*, (2019), 1534–1539. https://doi.org/10.1109/ICDM.2019.00203
17. L. Liang, Z. Xu, Z. Song, I. King, Y. Qi, J. Ye, Tackling long-tailed distribution issue in graph neural networks via normalization, *IEEE Trans. Knowl. Data Eng.*, **36** (2024), 2213–2223. https://doi.org/10.1109/TKDE.2023.3315284
18. H. Deng, N. Zou, M. Du, W. Chen, G. Feng, Z. Yang, et al., Unifying fourteen post-hoc attribution methods with taylor interactions, *IEEE Trans. Pattern Anal. Mach. Intell.*, **46** (2024), 4625–4640. https://doi.org/10.1109/TPAMI.2024.3358410
19. F. Hu, Y. Zhu, S. Wu, W. Huang, L. Wang, T. Tan, GraphAIR: Graph representation learning with neighborhood aggregation and interaction, *Pattern Recognit.*, **112** (2021), 107745. https://doi.org/10.1016/j.patcog.2020.107745
20. J. Gao, J. Gao, X. Ying, M. Lu, J. Wang, Higher-order interaction goes neural: A substructure assembling graph attention network for graph classification, *IEEE Trans. Knowl. Data Eng.*, **35** (2023), 1594–1608. https://doi.org/10.1109/TKDE.2021.3105544
21. A. Kazi, L. Cosmo, S. Ahmadi, N. Navab, M. M. Bronstein, Differentiable graph module (DGM) for graph convolutional networks, *IEEE Trans. Pattern Anal. Mach. Intell.*, **45** (2023), 1606–1617. https://doi.org/10.1109/TPAMI.2022.3170249
22. J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, G. Sun, xDeepFM: Combining explicit and implicit feature interactions for recommender systems, in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, (2018), 1754–1763. https://doi.org/10.1145/3219819.3220023
23. Z. Shen, Z. Kang, When heterophily meets heterogeneous graphs: Latent graphs guided unsupervised representation learning, *IEEE Trans. Neural Networks Learn. Syst.*, **36** (2025), 10283–10296. https://doi.org/10.1109/TNNLS.2025.3540063
24. S. Yun, K. Kim, K. Yoon, C. Park, LTE4G: Long-tail experts for graph neural networks, in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, (2022), 2434–2443. https://doi.org/10.1145/3511808.3557381
25. J. Wu, J. He, J. Xu, DEMO-Net: Degree-specific graph neural networks for node and graph classification, in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, (2019), 406–415. https://doi.org/10.1145/3292500.3330950
26. X. Jiang, R. Zhu, P. Ji, S. Li, Co-Embedding of nodes and edges with graph neural networks, *IEEE Trans. Pattern Anal. Mach. Intell.*, **45** (2023), 7075–7086. https://doi.org/10.1109/TPAMI.2020.3029762
27. X. Tang, H. Yao, Y. Sun, Y. Wang, J. Tang, C. Aggarwal, et al., Investigating and mitigating degree-related biases in graph convolutional networks, in *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, (2020), 1435–1444. https://doi.org/10.1145/3340531.3411872
28. Z. Zhao, Z. Yang, C. Li, Q. Zeng, W. Guan, M. Zhou, Dual feature interaction-based graph convolutional network, *IEEE Trans. Knowl. Data Eng.*, **35** (2023), 9019–9030. https://doi.org/10.1109/TKDE.2022.3220789

29. W. Cheng, Y. Shen, L. Huang, Adaptive factorization network: learning adaptive-order feature interactions, in *Proceedings of the AAAI Conference on Artificial Intelligence*, **34** (2020), 3609–3616. https://doi.org/10.1609/aaai.v34i04.5768

30. M. Guang, C. Yan, Y. Xu, J. Wang, C. Jiang, Graph convolutional networks with adaptive neighborhood awareness, *IEEE Trans. Pattern Anal. Mach. Intell.*, **46** (2024), 7392–7404. https://doi.org/10.1109/TPAMI.2024.3391356

31. W. Guan, X. Yang, M. Li, Q. Guo, K. Liu, Q. Sun, VQIT-GNN: A collaborative knowledge transfer for node-level structure imbalance, *Pattern Recognit.*, **172** (2026), 112632. https://doi.org/10.1016/j.patcog.2025.112632

32. P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, preprint, arXiv:1710.10903.

33. G. Zhu, K. Liu, X. Yang, Q. Guo, UMGCN: Updating multi-graph for graph convolutional networks, *Comput. Electr. Eng.*, **123** (2025), 109957. https://doi.org/10.1016/j.compeleceng.2024.109957

34. Q. Guo, X. Yang, M. Li, Y. Qian, Collaborative graph neural networks for augmented graphs: a local-to-global perspective, *Pattern Recognit.*, **158** (2025), 111020. https://doi.org/10.1016/j.patcog.2024.111020

35. H. Cong, X. Yang, K. Liu, Q. Guo, Feature-topology cascade perturbation for graph neural network, *Eng. Appl. Artif. Intell.*, **152** (2025), 110657. https://doi.org/10.1016/j.engappai.2025.110657

AIMS Press