Contemporary Mathematics
https://ojs.wiserpub.com/index.php/CM/

Research Article

A Novel Graph Classification Approach Based on HITS-Inspired Graph Attention Network

Jie Yang[1], Qingyang Li[2], Min Yao[1], Yu Zhou[3*]

[1] Digitalization Department, Shanxi Branch, China Unicom, Taiyuan, Shanxi, China
[2] College of Artificial Intelligence, Taiyuan University of Technology, Taiyuan, Shanxi, China
[3] College of Computer Science and Technology (College of Data Science), Taiyuan University of Technology, Taiyuan, Shanxi, China
E-mail: zhouyu@tyut.edu.cn

Received: 10 June 2025; **Revised:** 9 July 2025; **Accepted:** 16 July 2025

Abstract: Graph Neural Networks (GNNs) have achieved notable success in graph classification tasks, such as chemical molecular graph classification in cheminformatics and drug design, owing to their ability to effectively capture structural features of molecular graphs. However, conventional GNNs still face challenges in comprehensively characterizing complex node interactions, particularly in discerning distinct node roles and their intricate interplay. To address this limitation, we propose a novel Hyperlink-Induced Topic Search (HITS)-inspired Interactive Graph ATtention network (HITS-InterGAT) for graph classification. This model employs an interactive attention mechanism to capture nuanced node interactions within molecular graphs. Specifically, we integrate the HITS algorithm to compute node authority and hub features, followed by an interaction graph attention mechanism to model feature-level relationships. Extensive experiments on multiple chemical and molecular datasets demonstrate that HITS-InterGAT significantly outperforms existing baselines in both classification accuracy and robustness. Our findings underscore the substantial potential of combining interactive attention mechanisms with the HITS algorithm for advancing chemical molecular graph classification.

***Keywords*:** graph classification, graph attention mechanism, HITS algorithm, Graph Neural Networks (GNNs)

MSC: 68T07

Abbreviation

GNN	Graph Neural Network
HITS	Hyperlink Induced Topic Search
GAT	Graph Attention Network
HITS-InterGAT	HITS-inspired Interactive Graph Attention Network

1. Introduction

Graph classification plays a significant role in various real-world applications. In cheminformatics and drug design, molecular structures are naturally represented as graphs. Extracting meaningful features from these molecular graphs facilitates the prediction of physical, chemical, and biological properties. This capability is crucial for screening compounds with specific characteristics, significantly accelerating drug discovery efficiency [1–4]. In biology, graph classification is vital for analyzing protein interaction networks, gene regulatory networks, and cell communication networks. It enables the identification of disease-related genes and prediction of protein functions, proving indispensable for research on major diseases [5, 6]. Similarly, in social network analysis, graph classification studies user behaviors and relationship patterns, uncovering insights that enable more accurate personalized recommendations and enhance user experience [7, 8]. Across these domains, entities represent atoms, proteins, genes, users, or items, while edges denote interactions (e.g., chemical bonds, protein interactions, gene expression regulations, purchase relationships). The key challenge for graph classification tasks lies in effectively capturing both local and global structural information and embedding nodes along with the entire graph into Euclidean space.

Graph Neural Networks (GNNs) have emerged as a fundamental methodology for graph classification. GNNs uncover deep structural relationships by learning node- and graph-level representations through message passing between nodes. Their applications span cheminformatics, drug design, bioinformatics, social science, and other domains, driving innovation and technological advancement. Specifically, GNNs aggregate information from each node's neighbors and update the node's features. Global graph representations are then obtained through pooling functions [9]. Liu et al. [10] introduce a GNN architecture that adaptively learns receptive paths for nodes. Qian et al. [11] learn hierarchical kernelized representations by mapping substructure invariants into composite invariants. Notably, different nodes often play distinct roles (e.g., authority and hub roles in the HITS algorithm), which should interact to achieve consistent node- and graph-level representations. Traditional GNNs frequently fail to account for these distinct roles, resulting in representations that capture only partial structural information.

To address this limitation, we leverage authority and hub roles from the HITS algorithm and their interactions to capture node-level features. Integrating these distinct roles into graph-level representations enhances classification accuracy for differently labeled graphs. Building on this foundation, we propose the HITS-inspired Interactive Graph Attention network (HITS-InterGAT), a novel framework for learning graph-level representations. Specifically, each node is characterized by two feature types: authority features and hub features. The framework updates authority features through weighted aggregation of neighboring authority features, with weights dynamically computed from hub features. Conversely, hub features are updated via weighted aggregation of neighboring hub features, with weights derived from authority features. This bidirectional interaction enables HITS-InterGAT to capture intricate relational patterns and dependencies within the graph structure. The mechanism adaptively modulates neighboring nodes' influence, enhancing the model's ability to capture nuanced structural dependencies. Consequently, integrating authority/hub features and their dynamic interactions significantly boosts graph classification performance. Experimental evaluations substantiate our hypothesis, demonstrating that HITS-InterGAT effectively captures complex interaction patterns. The model achieves outstanding performance across multiple chemical molecule datasets, delivering notable improvements in classification accuracy and robustness for intricate graph-structured data.

The contributions of this paper are summarized as follows:

1. We propose an interactive graph attention network incorporating two distinct node roles and their interactions. By independently computing authority and hub features, the model adeptly captures intricate graph interaction patterns, significantly enhancing the accuracy and robustness of graph classification tasks.

2. We introduce a novel graph pooling operator using Top-k pooling to selectively retain the highest-scoring nodes. This operator enhances global feature learning by preserving the most informative nodes, effectively balancing dimensionality reduction with critical structural information retention.

3. We conduct comprehensive experiments on multiple chemical molecule datasets, comparing our approach with baseline models. Results demonstrate that HITS-InterGAT consistently and significantly outperforms traditional GNNs in classification accuracy, highlighting its broad applicability in molecular graph classification tasks.

The remainder of this paper is structured as follows. Section 2 overviews related work on graph neural networks and classification. Section 3 presents notations, fundamental concepts, and background. Section 4 details the HITS-InterGAT architecture. Section 5 discusses experimental setup, evaluations, and results analysis. Finally, Section 6 summarizes findings and contributions.

2. Related work

GNNs have made remarkable breakthroughs in graph classification tasks, establishing themselves as a fundamental framework for this domain. To my knowledge, the origins of GNN research trace back to the seminal work in [12]. This pioneering study marked a milestone by successfully adapting Convolutional Neural Networks (CNNs) to graph-structured data, laying the foundation for graph representation learning and profoundly influencing subsequent GNN development. Building on this foundation, Kipf and Welling [13] introduce a groundbreaking graph convolutional network model for semi-supervised learning on graph-structured data. Veličković et al. [9] present the graph attention network, which employs attention mechanisms to operate on graph-structured data and has become a foundational work in the GNN community. Hamilton et al. [14] introduce a framework for inductive representation learning on large graphs that scalably generates embeddings for unseen nodes. Zhou et al. [15] provide a comprehensive survey of GNNs, presenting a systematic taxonomy instrumental in understanding and categorizing GNN approaches.

Graph Classification. Applying GNNs to graph classification tasks requires graph pooling operators to aggregate node embeddings into a unified global graph representation. Differentiable Pooling (DiffPool), proposed by Ying et al. [16], learns hierarchical graph structures by employing a soft assignment approach to aggregate nodes into super-nodes, making it highly effective for graph classification. Graph U-Net, proposed by Gao and Ji [17], introduces a novel GNN architecture inspired by U-Net [17]. They propose a framework that incorporates pooling (down-sampling) and unpooling (up-sampling) operations to enable hierarchical feature learning on graphs. The work in [18] proposes Graph Neural Networks with Learnable Structural and Positional Representations (LSPR), integrating structural and positional encodings into the GNN architecture. These encodings, learned during training, provide complementary information to node features and graph topology. Liu et al. [10] introduce a GNN architecture that adaptively learns receptive paths for nodes, effectively capturing local and global structural information for tasks like graph and node classification.

Graph Representation Learning. Graph representation learning [19] is a form of unsupervised learning that leverages reconstruction or contrastive losses to learn meaningful node embeddings. These embeddings can then be utilized for downstream tasks such as graph classification, node classification, community detection, and subgraph counting [20], among others. The Variational Graph Autoencoder (VGAE) model, proposed by Kipf and Welling [21], combines ideas from variational autoencoders [22] and graph convolutional networks [11] to learn meaningful graph latent representations. GraphGAN, proposed by Wang et al. [23], leverages Generative Adversarial Networks (GANs) [24] for graph generation and representation learning. It generates realistic graph structures by learning node embeddings through adversarial training. The work in [25] improves unsupervised graph representation learning by introducing a framework that uses personalized data augmentation for graph contrastive learning. Hou et al. [26] employ masked autoencoders to learn node representations on graph-structured data in an unsupervised manner. Specifically, GraphMAE [26] uses masked autoencoding to reconstruct graph structures and node features, enabling learning without labeled data. The authors subsequently extend GraphMAE to devise a decoding-enhanced masked self-supervised graph learner.

3. Preliminaries

3.1 *Symbol definitions*

A graph is denoted as $G = (V, E)$, where V represents the node set and E the edge set. For any node $v \in V$, N_v indicates the set of its one-hop neighbors in G. Given a graph dataset $\mathscr{G} = \{G^1, \ldots, G^N\}$, the graph classification task aims to predict the label $y_G \in \mathscr{Y} = \{1, \ldots, C\}$ for each graph $G \in \mathscr{G}$. The i-th graph $G^i = (V^i, E^i)$ consists of a node set $V^i = \{v_1^i, v_2^i, \ldots, v_{n_i}^i\}$ with n_i nodes, an edge set E^i, an adjacency matrix $A^i \in \mathbb{R}^{n_i \times n_i}$ capturing node connectivity, and a node feature matrix $H^i \in \mathbb{R}^{n_i \times d}$, where the k-th row $h_k^i \in \mathbb{R}^d$ corresponds to the feature vector of node v_k^i (d denotes the feature dimension). Additionally, each graph G^i has a graph-level representation $h_{G^i} \in \mathbb{R}^d$, where the embedding dimension d typically matches the node feature dimension.

3.2 *Hyperlink-induced topic search*

The Hyperlink-Induced Topic Search (HITS) algorithm, introduced by Kleinberg [27], is a link analysis method for webpage ranking. Its core concept involves identifying two distinct page roles through hyperlink structure analysis:

1. Authority pages: Topic-specific resources extensively cited by other pages;
2. Hub pages: Connectors that reference multiple authority pages.

The algorithm leverages the mutual reinforcement relationship between these roles: authorities derive weight from hubs pointing to them, while hubs gain significance by linking to authorities. This interdependence is quantified through iterative updates of authority and hub scores.

Let $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ represent a directed graph, where $\widetilde{V}$ is the set of webpages, and $\widetilde{E}$ is the set of hyperlinks (i.e., directed edges) between the webpages. Let $\widetilde{a}_i$ be the authority value of webpage i, $\widetilde{h}_i$ be the hub value of webpage i, N_i be the set of webpages with incoming links to webpage i (i.e., the set of all webpages that point to i), and M_i be the set of webpages with outgoing links from webpage i (i.e., the set of all webpages that i points to).

First, initial authority and hub values are assigned to all webpages, typically initialized according to equation (1):

$$\widetilde{a}_i^{(0)} = 1, \widetilde{h}_i^{(0)} = 1. \tag{1}$$

The HITS algorithm iteratively calculates authority and hub values. In each iteration, the update process consists of two steps: updating authority values and updating hub values. As shown in equation (2), the authority value of a webpage is updated to the sum of the hub values of the pages that point to it:

$$\widetilde{a}_i^{(k+1)} = \sum_{j \in N_i} \widetilde{h}_j^{(k)}. \tag{2}$$

As shown in equation (3), the hub value of a webpage is updated to the sum of the authority values of the pages it points to:

$$\widetilde{h}_i^{(k+1)} = \sum_{j \in M_i} \widetilde{a}_j^{(k)}. \tag{3}$$

After each iteration, the authority and hub values are typically normalized to mitigate overflow caused by excessively large values. The normalization step is shown in equation (4):

$$\text{normalize}(\widetilde{\mathbf{a}}) = \frac{\widetilde{a}_i}{\|\widetilde{\mathbf{a}}\|_2}, \quad \text{normalize}(\widetilde{\mathbf{h}}) = \frac{\widetilde{h}_i}{\left\|\widetilde{\mathbf{h}}\right\|_2}, \tag{4}$$

where $\|\widetilde{\mathbf{a}}\|_2$ and $\left\|\widetilde{\mathbf{h}}\right\|_2$ represent the $L2$-norms of the authority vector $\widetilde{\mathbf{a}}$ and the hub vector $\widetilde{\mathbf{h}}$, respectively. The HITS algorithm iteratively updates the authority and hub values until the changes are less than a predefined threshold or the maximum number of iterations is reached, as shown in equation (5):

$$\left\| \widetilde{\mathbf{a}}^{(k+1)} - \widetilde{\mathbf{a}}^{(k)} \right\|_2 < \varepsilon, \quad \left\| \widetilde{\mathbf{h}}^{(k+1)} - \widetilde{\mathbf{h}}^{(k)} \right\|_2 < \varepsilon, \tag{5}$$

where ε is a small positive number serving as the tolerance for convergence.

A defining characteristic of the HITS algorithm is its division of pages into two distinct roles: authorities and hubs. This role-based approach fundamentally differentiates it from global ranking algorithms like PageRank, which measure overall webpage influence. While HITS demonstrates high effectiveness for topic-specific page ranking, it exhibits notable limitations:

1. Sensitivity to link spamming: Artificial creation of hub pages can inflate authority rankings;

2. Query-specific computation: Requires separate processing per query, resulting in higher computational costs. Despite these limitations, HITS remains a foundational web ranking algorithm that has profoundly influenced search engine development.

3.3 *GNN*

Graph Neural Networks (GNNs) are deep learning methods designed for graph-structured data. They learn node and graph representations by leveraging structural information (nodes, edges, and features) through a multi-layer message-passing mechanism. This iterative process operates as follows:

1. Nodes exchange information with neighbors;
2. Each node updates its representation using aggregated neighbor information;
3. Steps 1-2 repeat for predefined iterations or until convergence.

The resulting representations enable effective performance on downstream tasks including, e.g. node classification, graph classification, link prediction.

The Laplacian matrix L represents graph structure and is fundamental for defining graph convolution operations in Graph Neural Networks. Given a graph's adjacency matrix A, the degree matrix D is diagonal with entries $D_{ii} = \sum_j A_{ij}$ representing node degrees. The standard graph Laplacian $L = D - A$ enables convolution operations as in equation (6):

$$h_v^{(k)} = \sigma\left(\sum_{u \in \mathcal{N}(v)} L_{vu} \cdot h_u^{(k-1)}\right) \tag{6}$$

where L_{vu} denotes the Laplacian element representing connection weight from u to v, and σ is an activation function.

To address node degree variations, we normalize the Laplacian according to the following equation:

$$\tilde{L} = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}} \tag{7}$$

yielding the normalized convolution according to the following equation:

$$h_v^{(k)} = \sigma\left(\sum_{u \in \mathcal{N}(v)} \tilde{L}_{vu} \cdot h_u^{(k-1)}\right) \tag{8}$$

General GNNs (e.g., GCN) follow the message passing framework:

$$m_{vu}^{(k)} = h_u^{(k-1)} \cdot w_{vu} \tag{9}$$

where $m_{vu}^{(k)}$ is the message from v to u, and w_{vu} represents edge weight. Node features update by aggregating neighborhood messages (equation (10)):

$$h_u^{(k)} = \sigma\left(\sum_{v \in \mathcal{N}(u)} m_{vu}^{(k)} \cdot b_u\right) \tag{10}$$

4. Graph classification model

To extract structural information of molecular graphs and improve the quality of graph embedding learning, thus enhancing the accuracy of graph classification, this paper proposes a graph neural network model based on an interactive attention mechanism (HITS-InterGAT), as shown in Figure 1.

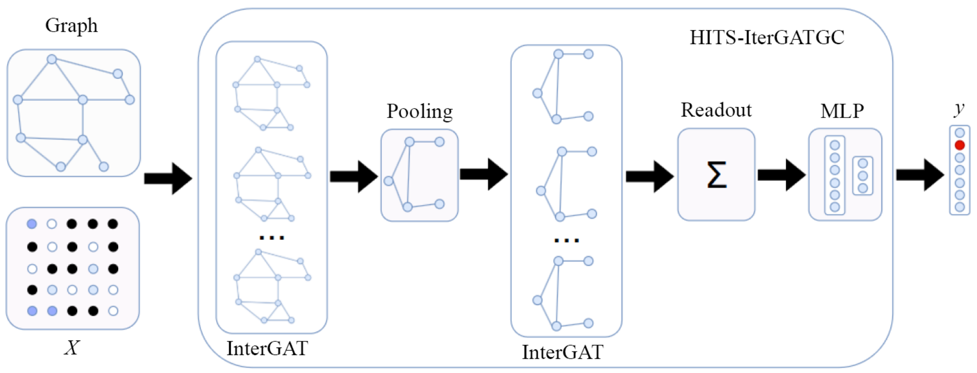

Figure 1. Model framework. Here X is the node representation for the graph

The proposed architecture comprises three core components:

1. Interactive Attention Module: Extracts neighbor information using graph topology and HITS algorithm principles. Each module contains dual graph convolution layers that output updated node embeddings, with the primary objective of capturing structural information;

2. Graph Pooling Layer: Performs down-sampling by aggregating local nodes to generate compact graph representations. This is particularly critical for large-scale graph processing and global feature extraction;

3. Classification Network: Receives graph representations processed through a final interactive attention layer to perform graph classification.

4.1 *Interactive attention mechanism module*

Specifically, the core module in HITS-InterGAT, named HITS-InterGATConv, takes responsibility for updating Hub and Authority scores. The HITS-InterGAT module sequentially connects multiple dual HITS-InterGATConv layers and applies batch normalization layers after each layer to stabilize training. The structure of the HITS-InterGAT module is shown in Figure 2.

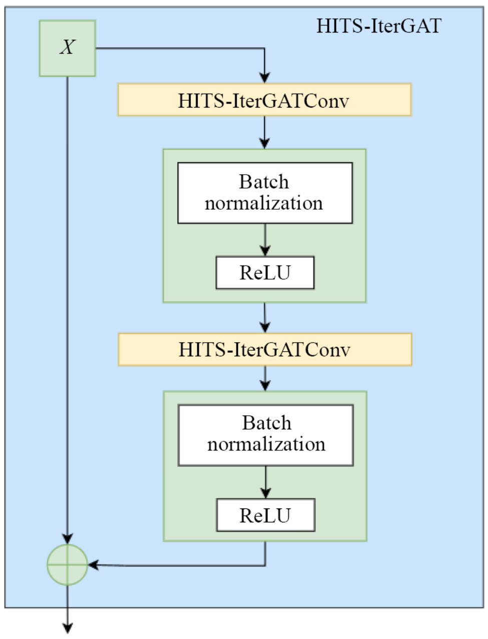

Figure 2. HITS-InterGAT module

In the HITS-InterGATConv module, we employ a dual attention mechanism to compute the Authority and Hub representations of nodes. This approach is inspired by the HITS algorithm, where Authority measures a node's importance as an information receiver, while Hub measures its importance as an information sender. Through these dual perspectives, our model better captures node relationships and achieves more accurate feature aggregation during message passing.

We compute edge Hub weights using node Authority features. For nodes u and v with Authority features $\widetilde{a}_u$ and $\widetilde{a}_v$, the Hub weight is given by equation (11):

$$w_{uv}^{\text{hub}} = \frac{\widetilde{a}_u \cdot \widetilde{a}_v}{\sqrt{d_{\text{out}}(u)}} \tag{11}$$

where $d_{\text{out}}(u)$ denotes the out-degree of node u (edges originating from u). These weights are normalized via Softmax (equation (12)):

$$\hat{w}_{uv}^{\text{hub}} = \frac{\exp(w_{uv}^{\text{hub}})}{\sum_{v' \in \mathcal{N}(u)} \exp(w_{uv'}^{\text{hub}})} \tag{12}$$

Authority weights are computed from Hub features. For node v, the Authority weight is (equation (13)):

$$w_{vu}^{\text{auth}} = \frac{\widetilde{h}_u \cdot \widetilde{h}_v}{\sqrt{d_{\text{in}}(v)}} \tag{13}$$

where $\widetilde{h}_u$ and $\widetilde{h}_v$ are Hub features, and $d_{\text{in}}(v)$ is v's in-degree (edges terminating at v). Normalization yields (equation (14)):

$$\hat{w}_{vu}^{\text{auth}} = \frac{\exp(w_{vu}^{\text{auth}})}{\sum_{u' \in \mathcal{N}(v)} \exp(w_{vu'}^{\text{auth}})} \tag{14}$$

During message passing, Hub and Authority features update through neighborhood aggregation. Hub feature update (equation (15)):

$$\widetilde{h}_v^{\text{new}} = \text{LeakyReLU}\left(\sum_{u \in \mathcal{N}(v)} \hat{w}_{uv}^{\text{hub}} \cdot \widetilde{h}_u\right) \tag{15}$$

Authority feature update (equation (16)):

$$\widetilde{a}_v^{\text{new}} = \text{LeakyReLU}\left(\sum_{u \in \mathcal{N}(v)} \hat{w}_{vu}^{\text{auth}} \cdot \widetilde{a}_u\right) \tag{16}$$

Finally, updated Authority and Hub features are stacked into new node representations (equation (17)):

$$X_i = \text{stack}(\widetilde{a}_v^{\text{new}}, \widetilde{h}_v^{\text{new}}) \tag{17}$$

where stack denotes feature concatenation, producing a tensor of shape $(N, \text{outdim}, 2)$ with N nodes, outdim feature dimension, and 2 channels (Authority/Hub).

We demonstrate the computation process of each HITS-InterGAT module using a 4-node molecular graph. The adjacency matrix A represents graph connectivity:

$$A = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix}.$$

Initial Authority and Hub features are:

$$\widetilde{a} = \begin{bmatrix} 0.1 \\ 0.2 \\ 0.3 \\ 0.4 \end{bmatrix}, \quad \widetilde{h} = \begin{bmatrix} 0.5 \\ 0.6 \\ 0.7 \\ 0.8 \end{bmatrix}.$$

As shown in equation (11), Hub weights w_{uv}^{hub} are calculated from Authority features. First, node out-degrees are computed:

• $d_{\text{out}}(v_1) = 2$ (edges to v_2 and v_4),
• $d_{\text{out}}(v_2) = 2$ (edges to v_1 and v_3),
• $d_{\text{out}}(v_3) = 2$ (edges to v_2 and v_4),

• $d_{\text{out}}(v_4) = 2$ (edges to v_1 and v_3).
Hub weights are then computed for each edge. For v_1-v_2 ($\widetilde{a}_{v_1} = 0.1$, $\widetilde{a}_{v_2} = 0.2$):

$$w_{12}^{\text{hub}} = \frac{\widetilde{a}_{v_1} \cdot \widetilde{a}_{v_2}}{\sqrt{d_{\text{out}}(v_1)}} = \frac{0.1 \times 0.2}{\sqrt{2}} \approx 0.013.$$

Other edge weights: $w_{13}^{\text{hub}} \approx 0.028$, $w_{14}^{\text{hub}} \approx 0.014$, $w_{23}^{\text{hub}} \approx 0.017$, $w_{24}^{\text{hub}} \approx 0.024$, $w_{34}^{\text{hub}} \approx 0.024$.
Softmax normalization is applied to Hub weights. For v_1's neighbors (v_2 and v_4):

$$\hat{w}_{12}^{\text{hub}} = \frac{\exp(w_{12}^{\text{hub}})}{\sum_{j \in \mathcal{N}(1)} \exp(w_{1j}^{\text{hub}})} = \frac{\exp(0.014)}{\exp(0.014) + \exp(0.028)} \approx 0.497.$$

We similarly compute normalized weights for all edges.
As shown in equation (15), Hub features are updated using normalized weights. For v_1:

$$h_1^{\text{new}} = \text{LeakyReLU}\left(\hat{w}_{12}^{\text{hub}} \cdot \widetilde{h}_2 + \hat{w}_{14}^{\text{hub}} \cdot \widetilde{h}_4\right),$$

where $\widetilde{h}_2$ and $\widetilde{h}_4$ denote Hub features of v_2 and v_4. Other nodes' Hub features update analogously. Authority features update similarly. Final node representations concatenate updated Authority and Hub features.

4.2 *Graph pooling*

General graph pooling operations include summing, averaging, taking the maximum, or performing weighted summation of node features, all satisfying the two requirements of graph pooling functions. In GIN networks, the graph pooling function first sums these features and then concatenates the outputs from all layers. This pooling operation reduces graph size while preserving structural information. Top-k pooling selects nodes based on feature importance, serving as a method that retains the most informative nodes in graph convolutional networks. This enables more efficient graph representation by reducing graph size. The approach assigns each node a score through a scoring mechanism and retains the top k of nodes with the highest scores.

First, we compute a score for each node in the graph. The input node feature matrix is $X \in \mathbb{R}^{N \times d}$, where N denotes the number of nodes and d represents the feature dimension of each node. Typically, these node scores are obtained through a learnable linear transformation according to the following equation

$$s = Xw. \tag{18}$$

Here, $w \in \mathbb{R}^d$ is a learnable weight vector, and $s \in \mathbb{R}^N$ represents the score vector for all nodes.

For the score vector s, we select the top k% of nodes with the highest scores (commonly referred to as top-k nodes). The number of selected nodes is $k = p \times N$, where p denotes the pooling ratio (e.g., 50% corresponds to $p = 0.5$). These top-scoring nodes are selected using an index mask according to the following equation:

$$\text{idx}_{\text{top}k} = \text{argsort}(s)[-k:] \tag{19}$$

The features of retained nodes are normalized through scaling using equation (20):

$$X_{\text{pool}} = X_{\text{top}k} \odot \sigma(s_{\text{top}k}) \tag{20}$$

Here, $X_{\text{top}k}$ denotes the feature matrix of the retained top-k nodes, $\sigma(\cdot)$ represents a nonlinear activation function, $s_{\text{top}k}$ corresponds to the scores of these retained nodes, and $\odot$ indicates element-wise multiplication.

Finally, we update the graph structure and regenerate a subgraph using the retained nodes. This requires updating the adjacency matrix to retain only edges incident to the top-k nodes. Given the original adjacency matrix $A \in \mathbb{R}^{N \times N}$, the new adjacency matrix is computed as follows according to the following equation:

$$A_{\text{pool}} = A[X_{\text{top}k},\, \text{idx}_{\text{top}k}]. \tag{21}$$

Using the adjacency matrix A from Section 4.1 as an example, we first apply a linear transformation to the node features. Given weight vector $w = [0.5,\, 0.5]$, the score vector s is computed as:

$$s = Xw = \begin{bmatrix} 0.1 & 0.5 \\ 0.2 & 0.6 \\ 0.3 & 0.7 \\ 0.4 & 0.8 \end{bmatrix} \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.15 \\ 0.3 \\ 0.6 \\ 0.7 \end{bmatrix}$$

Assuming we retain the top 50% of nodes (i.e., the top 2 nodes) by highest scores, we first sort the score vector and select nodes 3 and 4, yielding $\text{idx}_{\text{top}k} = [2,\, 3]$. The feature matrix of selected nodes is:

$$X_{\text{top}k} = \begin{bmatrix} 0.5 & 0.7 \\ 0.6 & 0.8 \end{bmatrix}$$

Applying element-wise multiplication:

$$X_{\text{pool}} = \begin{bmatrix} 0.5 \times 0.6 & 0.7 \times 0.6 \\ 0.6 \times 0.7 & 0.8 \times 0.7 \end{bmatrix} = \begin{bmatrix} 0.3 & 0.42 \\ 0.42 & 0.56 \end{bmatrix}$$

Based on the retained node indices $\text{idx}_{\text{top}k} = [2,\, 3]$, we update the adjacency matrix by retaining only edges incident to these nodes. The resulting adjacency matrix A_{pool} contains only connections to nodes 2 and 3:

$$A_{\text{pool}} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

This indicates an edge exists solely between nodes 2 and 3, with all other connections removed. The process yields a more compact graph while preserving the most informative content for downstream tasks.

4.3 *Loss function*

Cross-Entropy Loss quantifies the discrepancy between a model's output probability distribution and the true distribution, commonly employed in multi-class classification tasks. Given a sample x with true label y, the model produces a class probability distribution $p(y|x)$. The cross-entropy loss is defined as equation (22):

$$CELoss = -\sum_{c=1}^{C} y_c \log(p_c) \tag{22}$$

where C denotes the number of classes, y_c represents the binary indicator (0/1) for class c in the true distribution, and p_c is the predicted probability for class c.

5. Experimental evaluations

5.1 *Dataset*

We evaluate the proposed HITE-InterGAT model on five popular graph classification datasets from TUDataset [28], with detailed statistics shown in Table 1. Here, G, C, V_{avg}, E_{avg}, N_{labels}, and E_{labels} denote the number of graphs, number of classes, average nodes per graph, average edges per graph, presence of node labels, and presence of edge labels, respectively. The symbols '+' and '-' indicate feature presence and absence.

1. MUTAG [29] contains 188 mutagenic aromatic and nitroaromatic compounds, labeled by their mutagenicity against *Salmonella Typhimurium*.

2. NCI1 [30] contains 4,110 compound graphs screened for activity against non-small cell ovarian cancer, with binary labels indicating presence or absence of anticancer activity.

3. PTC [31] comprises 344 compounds labeled by their carcinogenicity in rats, with binary classifications (carcinogenic/non-carcinogenic).

4. PROTEINS [32] contains 1,113 protein graphs labeled as enzymes/non-enzymes. Nodes represent amino acids with edges connecting those < 6? apart. Three discrete node labels represent helix, sheet, and turn structures.

5. Mutagenicity [33] contains 4,337 drug compound graphs with binary labels (mutagen/non-mutagen).

Table 1. Dataset

Dataset	MUTAG	NCI1	PTC	PROTEINS	Mutagenicity	ENZYMES	COLORS-3
G	188	4,110	344	1,113	4,337	600	10,500
C	2	2	2	2	2	6	11
V_{avg}	17.93	29.87	14.29	39.6	30.32	32.63	61.31
E_{avg}	19.79	32.30	14.69	72.82	30.77	62.14	91.03
N_{labels}	+	+	+	+	+	+	□C
E_{labels}	+	□C	+	□C	+	□C	□C

5.2 *Experimental setup and evaluation metrics*

The experiment performs 10-fold cross-validation for each dataset, with the dataset divided into 80% training set and 20% testing set. We use the Adam optimizer to minimize the cross-entropy loss function. The batch size and learning rate are respectively set to 64 and 0.01, and the learning rate is annealed by the cosine scheduling strategy.

The performance of the model is evaluated using accuracy, which measures the proportion of correctly classified samples, see (23).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{23}$$

5.3 *Baseline model*

To validate the effectiveness of the model, it will be compared with the following baseline models:

1. GCN [13]: Kipf et al. proposed a hierarchical propagation rule based on a first-order approximation of spectral convolution on graphs, which can encode both graph structure and node features, making it useful for semi-supervised classification.

2. GraphSAGE [14]: Hamilton et al. proposed a method that can directly generate features for unseen nodes without the need to repeat the entire feature generation process when a new node is added.

3. GIN [34]: Xu et al. proposed the theoretical foundation for reasoning about the expressive power of graph neural networks, proving the expressive power limits of popular GNN variants, and designing a graph isomorphism network with the maximum expressive power.

4. PPGN [35]: Maron et al. proposed a multilayer perceptron with matrix multiplication, demonstrating its ability to achieve the expressive power of the 3-WL test.

5. CapsGNN [36]: Zhang et al. inspired by capsule networks, extract node features in the form of capsules and capture important information at the graph level through a routing mechanism.

6. GraphNorm [37]: Cai et al. proposed the GraphNorm normalization method, which achieves faster convergence compared to BatchNorm and LayerNorm, while also improving the generalization ability of GNNs, resulting in better performance in graph classification.

7. HM-GNN [20]: Yu et al. constructed a heterogeneous motif graph that includes motif nodes and molecular nodes by extracting motif structures [38] from graph datasets. They used HM-GNN to learn graph embeddings, and then concatenated these embeddings with atomic-level graph embeddings learned by another GNN, which were used for graph classification tasks.

5.4 *Results and discussions*

As shown in Table 2, the proposed HITS-InterGAT model outperforms baseline models in classification accuracy on the MUTAG, NCI1, PTC, and Mutagenicity datasets, while achieving comparable accuracy to GCN on PROTEINS. This demonstrates significant performance improvements by HITS-InterGAT. The first three baselines—GCN, GraphSAGE, and GIN—employ standard graph neural networks for classification. The last four models (DGCNN, PPGN, CapsGNN, and GraphNorm) introduce architectural enhancements: selecting top-k nodes for pooling-based embedding generation, incorporating multi-layer perceptrons with matrix multiplication, or applying normalization techniques like GraphNorm. While significantly improving upon basic GNN performance, each method has limitations:

• DGCNN's sorting-based pooling captures top-k important nodes but may overlook local structures, yielding incomplete graph representations.

• PPGN enhances expressive power via multi-layer perceptrons but increases computational and storage overhead.

• CapsGNN improves robustness through capsule networks, but their complexity heightens training difficulty and instability.

• GraphNorm advances structural normalization but may underperform on graphs with highly imbalanced node counts.

Table 2. Baseline experiment classification accuracy results

Model	MUTAG	NCI1	PTC	PROTEINS	Mutagenicity
GCN	0.7437	0.7747	0.7000	**0.8335**	0.7767
GraphSAGE	0.7276	0.7169	0.6800	0.8250	0.6922
GIN	0.6969	0.6992	0.6571	0.7775	0.6969
PPGN	0.6932	0.7593	0.6857	0.7290	0.5902
CapsGNN	0.6922	0.6992	0.7714	0.8050	0.5286
GraphNorm	0.7494	0.6922	0.7714	0.7486	0.6615
HM-GNN	0.7544	0.7897	0.7854	0.7846	0.7789
HITS-IterGAT	**0.7553**	**0.7932**	**0.7857**	0.8317	**0.8112**

5.5 *The impact of the number of graph neural network layers on model performance*

During training, DualGCN employs only a single graph neural network layer. To analyze the impact of layer depth, we investigate performance using 1 to 5 layers. As shown in Figure 3, classification accuracy marginally decreases or stabilizes across all five datasets as layer count increases. Optimal performance occurs with a single layer, achieving peak classification accuracy.

Compared to the 4-layer GIN architecture in baseline model HM-GNN, this demonstrates that our interactive attention mechanism significantly reduces model complexity. Simultaneously, these results suggest that capturing higher-order neighbor features through additional layers may negatively impact graph embedding representations, ultimately failing to improve classification performance.

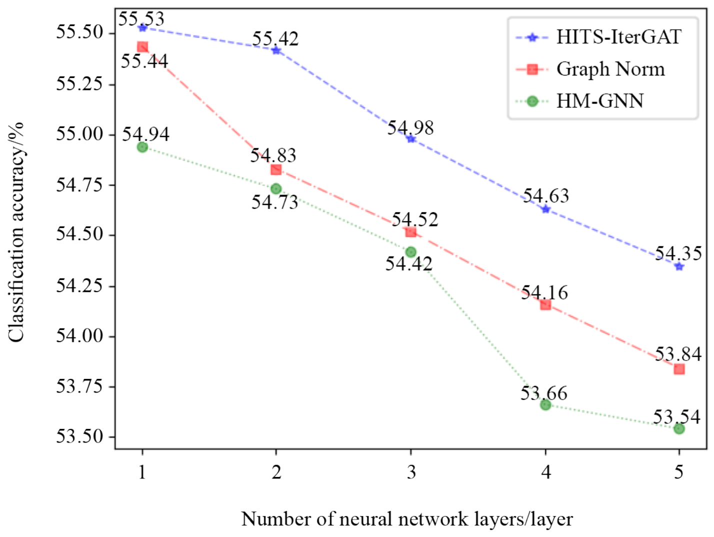

Figure 3. The impact of different network layer depths on data classification accuracy

5.6 *The impact of graph pooling layers on model performance*

To better understand the role of graph pooling layers, we conduct an ablation study by removing these layers and comparing results with the full model. Graph pooling layers are crucial in GNNs for dimensionality reduction and global information aggregation, enabling capture of long-range node dependencies. Their removal restricts the model to local graph information, impairing global structural representation.

Experimental results in Table 3 show HITS-InterGAT-pool (without pooling layers) exhibits performance degradation, particularly on complex graph structures. The pooling-free model converges slower during training and becomes prone to gradient vanishing in deeper architectures. Although computational load decreases and inference speed slightly improves without pooling layers, these benefits are outweighed by the overall performance decline.

Table 3. The impact of graph pooling layers on model performance

Model	MUTAG	NCI1	PTC	PROTEINS	Mutagenicity
InterGAT	**0.5553**	**0.5932**	**0.5857**	**0.5275**	**0.5450**
InterGAT-pool	0.5473	0.5789	0.5678	0.5089	0.5378

5.7 *The impact of Top-k values on model performance*

To investigate the impact of *k* values in the Top-*k* strategy, we conduct an ablation study by varying *k* and comparing performance outcomes. The Top-*k* operation selects the *k* most important nodes/features to reduce information processing load while enhancing computational efficiency. Different *k* values significantly influence graph convolutional networks' feature extraction capabilities and model performance.

Table 4. The impact of Top-*k* values on model performance

	20%	35%	50%	60%
MUTAG	0.5234	**0.5553**	0.5401	0.5112
NCI1	**0.5932**	0.5801	0.5634	0.5123
PTC	**0.5857**	0.5738	0.5521	0.5234
PROTEINS	0.5023	0.5157	**0.5275**	0.5241
Mutagenicity	0.5408	**0.5450**	0.5293	0.5126

Experimental results in Table 4 demonstrate that Top-*k* selection significantly influences model performance. Specifically, accuracy initially increases then stabilizes with increasing *k* values. At small *k* values, lower accuracy indicates insufficient capture of key features. As *k* increases, accuracy improves progressively until reaching an optimum before stabilizing. This reveals that moderate *k* values enhance performance, while excessively large values risk information redundancy.

6. Conclusion

This paper presented HITS-InterGAT, a novel method for chemical molecular graph classification based on an interactive attention mechanism. By integrating a dual graph convolutional network architecture with the HITS algorithm, HITS-InterGAT uniquely computes and leverages distinct node authority and centrality features. Experimental evaluations demonstrate that the method effectively captures intricate interactive information within molecular graphs, achieving excellent performance across multiple benchmark chemical datasets. Significant improvements in classification accuracy and model robustness were consistently observed. Through its core techniques of edge weight normalization and interactive node feature updating, HITS-InterGAT achieves a deep fusion of molecular structural information, substantially enhancing its capability to represent complex molecular relationships. These findings underscore the broad application potential and promising prospects of the HITS-InterGAT framework, powered by its interactive attention mechanism, for chemical molecular graph classification tasks.

Future work will explore several promising directions. Firstly, to enrich the graph-level representation and capture higher-order structural patterns, we plan to incorporate graph statistics such as network motifs and graphlets. Motifs are small, recurring, and statistically significant subgraph patterns that serve as fundamental building blocks of complex networks. Graphlets are small, connected, non-isomorphic induced subgraphs providing a finer-grained topological characterization. Formalizing a feature vector based on the frequency distribution of specific motifs and graphlets within a molecule could provide complementary structural signatures. This motif/graphlet signature could be integrated with the node-level features learned by HITS-InterGAT, potentially revealing functional or property-related patterns linked to specific local topologies. Secondly, evaluating the generalizability and effectiveness of HITS-InterGAT on other graph-structured data domains, such as social networks or citation networks, is crucial to validate its broader applicability beyond cheminformatics.

Acknowledgement

This research work was supported by the following projects, and we would like to express our sincere gratitude:

1. Shanxi Province Basic Research Project (202203021222120);
2. Shanxi Province Higher Education Science and Technology Innovation Project (2023L242).

The support from these projects provided crucial assurance for the smooth progress of this research and laid a solid foundation for the achievement of related academic results.

Conflict of interest

The authors declare no competing financial interest.

References

[1] Gilmer J, Schoenholz SS, Riley PF, Vinyals O, Dahl GE. Neural message passing for quantum chemistry. In: *ICML '17: Proceedings of the 34th International Conference on Machine Learning*. Sydney, Australia; 2017. p.1263-1272.

[2] Ozturk H, Ozgur A, Ozkirimli E. DeepDTA: Deep drug-target binding affinity prediction. *Bioinformatics*. 2018; 34(17): 821-829. Available from: https://doi.org/10.1093/bioinformatics/bty593.

[3] Jin W, Barzilay R, Jaakkola T. Junction tree variational autoencoder for molecular graph generation. In: *Proceedings of the International Conference on Machine Learning*. Stockholm, Sweden; 2018. p.2323-2332.

[4] Askr H, Elgeldawi E, Ella HA, Elshaier YAMM, Gomaa MM, Hassanien AE. Deep learning in drug discovery: An integrative review and future challenges. *Artificial Intelligence Review*. 2023; 56(7): 5975-6037. Available from: https://doi.org/10.1007/s10462-022-10306-1.

[5] Zhang M, Cui Z, Neumann M, Chen Y. An end-to-end deep learning architecture for graph classification. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. USA: AAAI Press; 2018. p.4438-4445.

[6] Ying R, You J, Morris C, Ren X, Hamilton WL, Leskovec J. Hierarchical graph representation learning with differentiable pooling. In: *NIPS '18: Proceedings of the 32nd International Conference on Neural Information Processing Systems*. Canada: Curran Associates Inc.; 2018. p.4805-4815.

[7] Ying R, Wang R, You J, Morris C, Leskovec J. Graph convolutional neural networks for web-scale recommender systems. In: *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. London United Kingdom: Association for Computing Machinery; 2018. p.974-983.

[8] He X, Deng K, Wang X, Li Y, Zhang Y, Wang M. LightGCN: Simplifying and powering graph convolution network for recommendation. In: *SIGIR '20: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery; 2020. p.639-648.

[9] Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y. Graph attention networks. In: *Proceedings of the International Conference on Learning Representations*. ICLR; 2018. p.1-12.

[10] Liu Z, Chen C, Li L, Zhou J, Li X, Song L, et al. GeniePath: Graph neural networks with adaptive receptive paths. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2019; 33(1): 4424-4431. Available from: https: //doi.org/10.1609/aaai.v33i01.33014424.

[11] Qian F, Bai L, Cui L, Li M, Lyu Z, Du H, et al. DHAKR: Learning deep hierarchical attention-based kernelized representations for graph classification. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2025; 39(19): 19995-20003. Available from: https://doi.org/10.1609/aaai.v39i19.34202.

[12] Niepert M, Ahmed M, Kutzkov K. Learning convolutional neural networks for graphs. In: *ICML '16: Proceedings of the 33rd International Conference on International Conference on Machine Learning*. USA; 2016. p.2014-2023.

[13] Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. In: *Proceedings of the International Conference on Learning Representations*. ICLR; 2017. p.1-14.

[14] Hamilton WL, Ying R, Leskovec J. Inductive representation learning on large graphs. In: *NIPS '17: Proceedings of the 31st International Conference on Neural Information Processing Systems*. USA: Curran Associates Inc.; 2017. p.1025-1035.

[15] Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, et al. Graph neural networks: A review of methods and applications. *AI Open*. 2020; 1: 57-81. Available from: https://doi.org/10.1016/j.aiopen.2021.01.001.

[16] Ying R, You J, Morris C, Ren X, Hamilton W, Leskovec J. Hierarchical graph representation learning with differentiable pooling. In: *NIPS '18: Proceedings of the 32nd International Conference on Neural Information Processing Systems*. Canada: Curran Associates Inc.; 2018. p.4805-4815.

[17] Gao H, Ji S. Graph u-nets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2022; 44(9): 4948-4960. Available from: https://doi.org/10.1109/TPAMI.2021.3081010.

[18] Ronneberger O, Fischer P, Brox T. U-Net: Convolutional networks for biomedical segmentation. In: *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer; 2015. p.234-241.

[19] Dwivedi VP, Luu AT, Laurent T, Bengio Y, Bresson X. Graph neural networks with learnable structural and positional representations. In: *Proceedings of the International Conference on Learning Representations*. ICLR; 2022. p.1-14.

[20] Yu Z, Gao H. Molecular graph representation learning via heterogeneous motif graph neural networks. In: *Proceedings of the International Conference on Machine Learning*. ICLR; 2022. p.25581-25594.

[21] Kipf TN, Welling M. Variational graph auto-encoders. In: *NeurIPS Workshop on Bayesian Deep Learning*. Canada; 2016. p.1-3.

[22] Kingma DP, Welling M. Auto-encoding variational bayes. In: *Proceedings of the International Conference on Learning Representations*. ICLR; 2014. p.1-14.

[23] Wang H, Wang J, Wang J, Zhao M, Zhang W, Zhang F, et al. GraphGAN: Graph representation learning with generative adversarial nets. In: *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*. AAAI; 2018. p.2508-2515.

[24] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative adversarial networks. *Communications of the ACM*. 2014; 63(11): 139-144. Available from: https://doi.org/10.1145/3422622.

[25] Zhang X, Tan Q, Huang X, Li B. Graph contrastive learning with personalized augmentation. *IEEE Transactions on Knowledge and Data Engineering*. 2024; 36(11): 6305-6316. Available from: https://doi.org/10.1109/TKDE.2024.3388728.

[26] Hou Z, Liu X, Cen Y, Dong Y, Yang H, Wang C, et al. GraphMAE: Self-supervised masked graph autoencoders. In: *KDD '22: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. USA: Association for Computing Machinery; 2022. p.594-604.

[27] Kleinberg JM. Authoritative sources in a hyperlinked environment. In: *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*. USA; 1999. p.668-677.

[28] Morris C, Kriege NM, Bause F, Kersting K, Mutzel P, Neumann M. TUDataset: A collection of benchmark datasets for learning with graphs. In: *ICML Workshop on Graph Representation Learning and Beyond*. ICML; 2020. p.1-11.

[29] Debnath AK, Lopez de Compadre RL, Debnath G, Shusterman AJ, Hansch C. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*. 1991; 34(2): 786-797.

[30] Nisius B, Weskamp N, Goller AH. Comparison of descriptor spaces for chemical classification. *Knowledge and Information Systems*. 2008; 14(14): 347-375. Available from: https://doi.org/10.1007/s10115-007-0103-5.

[31] Toivonen H, Srinivasan A, King RD, Kramer S, Helma C. Statistical evaluation of the predictive toxicology challenge 2000-2001. *Bioinformatics*. 2003; 19(10): 1183-1193. Available from: https://doi.org/10.1093/bioinformatics/btg130.

[32] Borgwardt KM, Ong CS, Schönauer S, Vishwanathan SVN, Smola AJ, Kriegel HP. Protein function prediction via graph kernels. *Bioinformatics*. 2005; 21(10): 47-56. Available from: https://doi.org/10.1093/bioinformatics/bti1007.

[33] Kazius J, McGuire R, Bursi R. Derivation and validation of toxicophores for mutagenicity prediction. *Journal of Medicinal Chemistry*. 2005; 48(1): 312-320. Available from: https://doi.org/10.1021/jm040835a.

[34] Xu K, Hu W, Leskovec J, Jegelka S. How powerful are graph neural networks? In: *Proceedings of the International Conference on Learning Representations*. ICLR; 2019. p.1-17.

[35] Maron H, Ben-Hamu H, Serviansky H, Lipman Y. Provably powerful graph networks. In: *Proceedings of the 33rd Conference on Neural Information Processing Systems*. Canada: Curran Associates Inc.; 2019. p.1-12.

[36] Zhang X, Chen L. Capsule graph neural network. In: *Proceedings of the International Conference on Learning Representations*. ICLR; 2018. p.1-16.

[37] Cai T, Luo S, Xu K, He D, Liu TY, Wang L. GraphNorm: A principled approach to accelerating graph neural network training. In: *Proceedings of the International Conference on Machine Learning*. ICLR; 2021. p.1204-1215.

[38] Ribeiro P, Paredes P, Silva MEP, Aparicio D, Silva FMA. A survey on subgraph counting: Concepts, algorithms, and applications to network motifs and graphlets. *ACM Computing Surveys*. 2022; 54(2): 1-36. Available from: https://doi.org/10.1145/343365.