Electronic
Research Archive

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

ERA, 34(2): 866–889.
DOI: 10.3934/era.2026040
Received: 17 November 2025
Revised: 15 January 2026
Accepted: 20 January 2026
Published: 28 January 2026

Research article

Extending graph neural networks to edge-level neighborhood awareness via line graph

Yutong Guo[1], **Wenrui Guan**[1,2], **Qihang Guo**[1,2], **Yuge Wang**[1], **Keyu Liu**[1] **and Xibei Yang**[1,*]

[1] School of Computer, Jiangsu University of Science and Technology, Zhenjiang 212100, China

[2] School of Economics and Management, Jiangsu University of Science and Technology, Zhenjiang 212100, China

* **Correspondence:** Email: jsjxy_yxb@just.edu.cn.

Abstract: Graph neural networks (GNNs) have been widely studied for handling graph-structured data. Neighborhood awareness, a mechanism for integrating context into graphs, plays a crucial role in learning node embeddings in GNNs. Existing methods typically perform neighborhood-aware steps only at the node or hop level, which limits their ability to learn edge-level semantic information. There are two main challenges in extending neighborhood awareness to the edge level: (1) designing a learning framework with message propagation of edge embeddings, which supports the construction of edge embeddings to capture pairwise node relationships and propagate messages to perceive local context, and (2) developing a fusion approach that bridges node and edge embedding spaces, thereby compressing edge-level structural information into node space to enhance the original neighborhood awareness. In this study, we propose an edge-level neighborhood awareness network (ELNA-Net) that integrates both node- and edge-level context for more comprehensive neighborhood awareness. Specifically, we use the Taylor interaction effect to construct explicit interactions between pairwise nodes, which approximates edge embeddings and captures intricate non-linear correlations. Furthermore, the adaptive edge-aware propagation module adopts a line graph to switch the roles of nodes and edges, revealing potential dependencies among neighboring edges and promoting neighborhood awareness. Finally, we propose a cross-mapping fusion approach to integrate node- and edge-level contexts within the graph convolution operator. Experimental results demonstrate that ELNA-Net achieves superior performance in semi-supervised node classification and link prediction, outperforming state-of-the-art models.

Keywords: semi-supervised learning; graph neural network; 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]. Neighborhood awareness is one of the critical mechanisms affecting the performance of GNN [2, 3]. In terms of aggregating node information, the existing neighborhood awareness can be classified into two main categories: greedy neighborhood awareness (GNA) and node-level neighborhood awareness (NLNA) [4].

GNA receives node information from neighbors without distinguishing the importance of nodes [5]. In contrast, NLNA selectively aggregates node information from relevant neighbors via attention [6] or sampling [7] mechanisms.

Despite the remarkable performance of emerging neighborhood-aware methods [8], these approaches primarily focus on direct or multi-hop neighbors while largely overlooking the edges that encode interaction patterns [9, 10]. As a fundamental component of graphs, edges not only define non-Euclidean associations between nodes but also carry rich semantic information about pairwise relationships [11]. Existing methods, however, often treat edges merely as binary indicators of connectivity or assign them scalar weights (e.g., attention scores), without explicitly modeling their intrinsic semantic content [12]. In practice, edges are critical for enhancing graph-based reasoning across diverse tasks. For instance, in molecular graph learning, edges encode chemical properties, such as bond type and spatial distance, facilitating accurate property prediction [13]; in knowledge graph reasoning, edges capture relational semantics, including hierarchies and temporal dependencies, supporting effective link prediction [14]; and in social recommendation networks, edges reflect user–item interactions, such as frequency and ratings, improving recommendation quality [15]. Neglecting such edge-level information limits the ability of neighborhood-aware methods to capture interaction patterns, often resulting in indistinguishable node embeddings, particularly in graphs with heterogeneous relationships [16]. Here we provide a simple example that will help to better understand this issue.

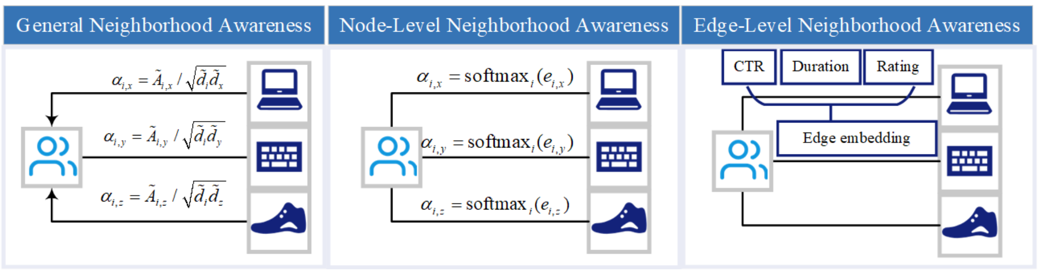

General Neighborhood Awareness
$\alpha_{i,x} = \tilde{A}_{i,x} / \sqrt{\tilde{d}_i \tilde{d}_x}$
$\alpha_{i,y} = \tilde{A}_{i,y} / \sqrt{\tilde{d}_i \tilde{d}_y}$
$\alpha_{i,z} = \tilde{A}_{i,z} / \sqrt{\tilde{d}_i \tilde{d}_z}$
Node-Level Neighborhood Awareness
$\alpha_{i,x} = \mathrm{softmax}_i(e_{i,x})$
$\alpha_{i,y} = \mathrm{softmax}_i(e_{i,y})$
$\alpha_{i,z} = \mathrm{softmax}_i(e_{i,z})$
Edge-Level Neighborhood Awareness
CTR
Duration
Rating
Edge embedding

Figure 1. Illustration of edge-level neighborhood awareness network.

As illustrated in Figure 1, social networks involve diverse types of relationships between users and items, while aggregation weights derived from the normalized Laplacian are fixed and thus limited in capturing fine-grained user preferences across heterogeneous interactions. Although attention mechanisms assign different importance to neighboring nodes or items, most existing methods rely on a single scalar coefficient to model each interaction, which is insufficient to represent rich and heterogeneous edge-level information. To overcome this limitation, the edge-level neighborhood awareness constructs explicit edge embeddings to encode diverse interaction features, such as click-through rates, user ratings, and dwell time. By modeling these signals directly at the edge level, the proposed approach provides more informative representations and improves prediction accuracy, while also enabling more effective identification of heterogeneous abnormal behaviors in applications such as fraud detection [17], which are difficult to capture using a single attention coefficient.

Building upon our conclusion that incorporating edge-level semantic information is critical to enhance model performance, this naturally gives rise to two key challenges. The first challenge is constructing a learning framework to generate edge embeddings and implement edge-level message propagation. Existing graph data for node classification typically lacks explicit edge features, creating a limitation in generating edge embeddings to capturing pairwise node relationships. In addition, the node-node adjacency matrix, as a carrier for message propagation, is not suitable for recursively aggregating edge embeddings. The second challenge is developing a cross-space fusion approach. Nodes and edges inherently reside in distinct semantic spaces: nodes represent entities while edges indicates relationships. This necessitates a fusion approach to bridge their heterogeneous feature spaces, enabling the adaptive compression of edge-level knowledge into the node embedding space.

To this end, we propose an edge-level neighborhood awareness network (ELNA-Net), a novel GNN framework that integrates two modules: adaptive edge-aware propagation and cross-mapping fusion. Specifically, in the adaptive edge-aware propagation module, we consider the high-level interaction of pairwise nodes from Taylor Interaction effect to adaptively construct edge embeddings. For achieving edge-level message propagation, we leverage line graph space transformation to update the edge embeddings. Through switching the role of edge and node, the edge embeddings are acuquired via convolutional operation to explore the relationship information of edges. Immediately, interactions are introduced into the edge level instead of only a node level. Finally, we design a cross-mapping fusion approach to compress edge-level semantic information into node space, achieving edge-level neighborhood awareness. Compared with other neighborhood-aware methods, the node embeddings obtained by ELNA-Net additionally capture the adaptive edge embedding of pairwise nodes and edge-level structure information through line graph. It is worth noting that we theoretically analyzed that constructing edge embeddings can improve the non-linear fitting ability of the model and provide classification-related semantic information. The main contributions can be summarized as follows.

- To promote the level of neighborhood awareness, an adaptive edge embeddings are constructed that adopts a node-edge switching strategy in line graph space to reveal the intricate relational dependencies.
- Considering that nodes and edges exist in different semantic spaces, we design a cross-mapping fusion approach to compress the semantic information of edges into the node space, achieving edge-level neighborhood awareness.
- We provide theoretical analysis to demonstrate the effectiveness of edge embedding, as well as abundant experiment results, and achieved excellent performance in node classification and link prediction tasks.

2. Related work

Graph neural networks (GNNs) have achieved tremendous success in graph-structured data learning. The pivotal mechanism determining the effectiveness of a GNN is neighborhood awareness [18]. However, the majority of neighborhood-aware methods can only perform a node aggregation that simply merges neighbors' features, without fully considering the interaction involving more refined relationship information. Therefore, we discuss related works from this perspective.

2.1. Node aggregation

Essentially, node aggregation operation is key of GNN, which converges the neighbors' feature for each node and then the node embedding representataion is obtained [19]. Typically, GCN combines a symmetric Laplacian smoothing into the operation of node aggregation [20]. Recently, with respect to various requirements, enumours advanced aggregation operations have been developed. For example, GraphSAGE [21] introduces the node-wise sampling method, which randomly selects k-hop neighbors to reduce the dependency of graph topology and then decrease the computational complexity of graph convolution. Furthermore, it is worth noting that following the popular information fusion thought, node aggregation operation can also be introduced into some combinational GNN frameworks. For instance, PAGCN fuses the node aggregations from the perspective of both augmentation graph and raw graph [13]; MAGCN attentively fuses the multiple node aggregations which are derived by multiple trustable topologies [22]. In addition, node-level domain awareness has gradually gained widespread attention from researchers. Its aggregation method mainly focuses on the importance of nodes within local neighborhoods. For example, GAT utilizes an attention-based aggregator that generates different coefficients to represent the correlation of neighbors; SGAT learns sparse attention coefficients under an L_0-norm regularization, which can identify task-irrelevant edges and perform aggregation [12]. Ginternet [23] extends the node-level scope of GNNs by incorporating auxiliary graphs into the aggregation process. It uses an attention mechanism to aggregate node relationships from other graphs into the current one. LOGO-GNN [24] introduces complementary learning from multiple perspectives and employs an attention mechanism to integrate both local and global perspectives. While there have been some works on modeling edge features, most of them focus on domains with explicit edge features [25, 26], such as knowledge graphs and molecular graphs. Methods for undirected graphs without edge features are relatively sparse. For instance, CensNet [27] uses an approach from recommender systems, where edge features are initialized as trainable parameters with a uniform distribution, relying solely on GCN aggregation to learn structural information. EGNN [28] also pays attention to edge feature learning. It combines the features of both ends of the edge with node features, concatenates them, and feeds the result into an encoder to obtain edge features.

Overall, greedy neighborhood awarenesss rely on fixed weights for aggregation, while attention-based aggregation allows for dynamic weighting based on node relationships. Considering edge features in aggregation models enables a richer understanding of the relationships between nodes, as they can provide additional context that is particularly valuable in tasks such as knowledge graph construction and molecular modeling. Edge-aware aggregation enhances the flexibility and expressiveness of the model because it allows for the adaptation of the model to more complex, real-world data structures.

2.2. Feature interaction

In order to capture more refined relationship information, a few researcheres [29, 30] have attempted to exploit the interactive neighborhood awareness [31]. The interactive neighborhood awareness explicitly constructs the interactions based on the paradigm of factorization machine [32], and then combines interactions with the raw node embedding [33]. For instance, AFN [34] transforms feature embeddings into a logarithmic space and constructs arbitrary-order feature interactions through feedforward architecture; DFI-GCN [35] extracts the arbitrary-order interactions between different features via Newton's identities, and integrates interactions into node embeddings using attention mechanism; and

GraphAIR [19] takes into account the interactions between nodes in different channels, and thus it represents the interactions as the inner product of node embeddings between two channels. However, the above interactions are designed manually that lacks adaptive capabilities, which cannot be applied to unfamiliar interaction patterns. Furthermore, these methods are limited to representing interactions at the node level and ignore the connectivity among edges, resulting in the loss of high-level relationship information.

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 $A_v \in \{0, 1\}^{N\times N}$, where each element $A_{v(i,j)} = 1$ 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.

Let $L(G) = (V_e, E_e)$ be a line graph with node set V_e and edge set $E_e \subseteq V_e \times V_e$. $A_e \in \{0, 1\}^{M\times M}$ is the node adjacency matrix of $L(G)$, or edge adjacency matrix of G. $T \in \{0, 1\}^{N\times M}$ is a binary transformation matrix and $T_{(i,j)} = 1$ represents that node v_i holds edge e_j, where $i \in \{1, 2, ..., N\}$ and $j \in \{1, 2, ..., M\}$. In Table 1, notations commonly used in the article are defined.

Table 1. Notation description.

Notation	Description
G	Original graph
A_v	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
$L(G)$	Line graph related to G
A_e	Node adjacency matrix of $L(G)$
V_e	Node set of $L(G)$
T	Transformation matrix
H^v	Node embedding
H^a	Adaptive edge embedding
H^e	High-level edge embedding
$\mathcal{N}$	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: GNA and NLNA. 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:

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

where $h_i^{(l)}$ is the embedding of the node v_i from the l-th convolutional layer, σ is a non-linear activation function which we used in this paper is *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:

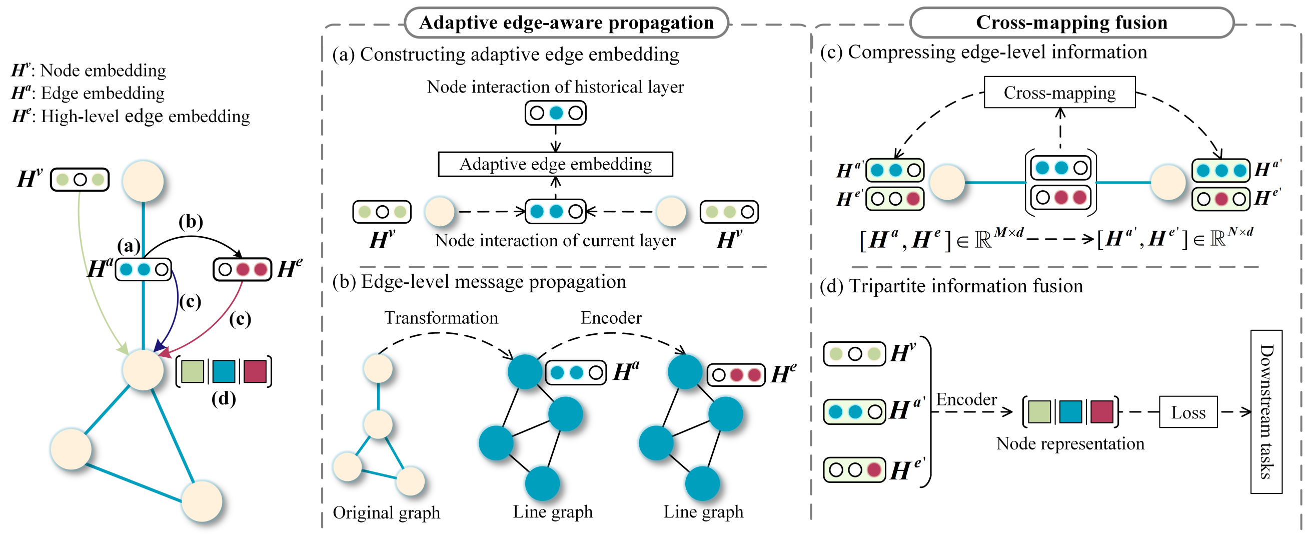
$$h_i^{(l+1)} = \sigma\Big(\sum_{v_j \in \mathcal{N}(i)} a_{ij}^{(l)} h_j^{(l)} W^{(l)}\Big), \tag{3.3}$$

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

4. Method

ELNA-Net aims to introduce edge-level semantic information to enrich the existing neighborhood awareness, which attempt to generate more discriminative node representations. Therefore, the model mainly consists of two main components: adaptive edge-aware propagation module (for constructing and learning edge embeddings) and cross-mapping approach (for bridging the edge-node feature space to integrate information). The pseudocode is provided in Algorithm 1, and the framework is displayed in Figure 2.

The following sections are structured to present the details and implementation of ELNA-Net. In Section 4.1, we present the process of adaptive edge-aware propagation module: Constructing adaptive edge embeddings and edge-level message propagation. In Section 4.2, we illustrate the details of cross-mapping fusion approach which integrates node- and edge-level semantic information into node embeddings. In Section 4.3, we introduce the loss functions of ELNA-Net to different downstream tasks: semi-supervised node classification and link prediction. In Section 4.4, we provide a detailed theoretical analysis to demonstrate the effectiveness of edge embedding.



H^v: Node embedding
H^a: Edge embedding
H^e: High-level edge embedding
Adaptive edge-aware propagation
(a) Constructing adaptive edge embedding
Node interaction of historical layer
Adaptive edge embedding
Node interaction of current layer
(b) Edge-level message propagation
Transformation
Encoder
Original graph
Line graph
Line graph
Cross-mapping fusion
(c) Compressing edge-level information
Cross-mapping
$[H^a, H^e] \in \mathbb{R}^{M\times d} \dashrightarrow [H^{a'}, H^{e'}] \in \mathbb{R}^{N\times d}$
(d) Tripartite information fusion
Encoder
Node representation
Loss
Downstream tasks

Figure 2. Overview framework of ELNA-Net. Module (a) constructs the adaptive edge embeddings by fusing the node interaction of current and historical layer. Module (b) explores the high-level edge embeddings through converting the original graph to the line graph. Module (c) compresses dual edge embeddings into the node space via the cross-mapping. Module (d) integrates tripartite information to jointly encode the node representation.

4.1. Adaptive edge-aware propagation

To provide comprehensive structure information at both node and edge level, we first focus on how to build a informative edge embedding. As the main component in graph structure, edges not only provide binary relationships (i.e., existence or non existence) or weight coefficients, but also represent the interaction patterns of reaching target nodes' neighbors.

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

$$f(\mathbf{x}) = f(\mathbf{b}) + \underbrace{\sum_{i=1}^{n} \frac{1}{1!} \cdot \frac{\partial f(\mathbf{b})}{\partial x_i} \cdot (x_i - b_i)}_{Independent \quad effects} + \underbrace{\sum_{i=1}^{n} \sum_{j=1}^{n} \frac{1}{2!} \cdot \frac{\partial^2 f(\mathbf{b})}{\partial x_i \partial x_j} \cdot (x_i - b_i)(x_j - b_j) + \cdots}_{Interaction \quad effects} \tag{4.1}$$

By aligning the results with neighborhood perception, we can observe that independent effect aims to merge the feature of neighboring nodes, while interactive effect aims to capture non-linear information of paired nodes. It can be concluded that the interaction term is the key to improve the non-linear fitting ability of the model. Unfortunately, the interaction term coefficient decreases rapidly with the increase of the order, and the existing literature [19, 36] has proved that the third-order interaction term coefficient is less than 1/48, which limits the model learning ability.

Therefore, we attempt to approximate edge embeddings from the perspective of interaction effects by explicitly constructing nonlinear feature interactions. To bridge the theoretical Taylor expansion with our specific implementation, we explicitly define the baseline point configuration. In the subsequent construction of adaptive edge embeddings, we set the baseline point to the zero vector,(i.e., $\mathbf{b} = \mathbf{0}$).

Under this zero-baseline assumption, the interaction term $(x_i - b_i)(x_j - b_j)$ in Eq (4.1) simplifies to $x_i x_j$. Consequently, the element-wise product introduced in our method (Eqs (4.3) and (4.4)) serves as a direct realization of these second-order interaction terms. While real-world node embeddings may not be strictly zero-mean, this choice is justified by the bias absorption property of neural networks: the mathematical deviation (linear and constant terms) caused by the non-zero mean is naturally absorbed by the learnable linear transformations in subsequent GNN layers. Furthermore, the zero-baseline assumption avoids the high computational cost of dynamic mean estimation, allowing our method to efficiently capture fine-grained pairwise interactions via simple element-wise products.

4.1.1. Constructing adaptive edge embedding

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

$$h_i^{(v,l+1)} = \sigma(\sum_{v_j \in \mathcal{N}(i)} \hat{A}_{v(i,j)} h_j^{(v,l)} W_v^{(l)}), \tag{4.2}$$

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

On this basis, a natural idea to model feature 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 interaction between v_i and v_j, and $\odot$ is the element-wise multiplication operator. The element-wise product captures fine-grained feature interactions: each dimension reflects the direct interaction of corresponding node features, highlighting strong relations while suppressing weak ones. It should be pointed out that this approximation method lacks consideration of the adaptive coefficient, and it is easy to cause performance tempering if it is blindly added to the forward propagation.

In order to comprehensively perceive node relationships, ELNA-Net not only captures node embeddings of the current layer, but also maintains node interactions of historical layer through residual connection. Then, through a learnable feature matrix S, adaptive node interaction can be automatically represented. The learning of adaptive edge embedding H^a is defined as follows:

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

where $h_{(i,j)}^{(a,l)}$ is the adaptive edge embedding between nodes v_i and v_j from the l-th layer, and $h_{(i,j)}^{(a,0)} = \sigma(h_i^{(v,0)} \odot h_j^{(v,0)})$. We denote $\odot$ as the element-wise product and $\oplus$ as the concatenation operation, respectively. $S \in \mathbb{R}^{d \times d'}$ is a learnable feature matrix, where d is the dimension of current layer and d' is the sum of interaction dimensions. The residual connection allows the model to retain historical edge information, complementing the current node interactions and preventing information loss.

4.1.2. Edge-level message propagation

As analyzed earlier, it is important to consider both node- and edge-level semantic information. Fortunately, the line graph transformation, which switches the role of nodes and edges, provides a natural approach for learning the edges as the primary focus. As shown in Figure 3, we illustrate the process of line graph transformation. Significantly, in line graph space, the pairwise nodes (v_i, v_j) are mapped to an edge index k, where $k = \{1, ..., M\}$. Thus, each adaptive node interaction $h_{(i,j)}^{(a,l)}$ becomes $h_k^{(a,l)}$.

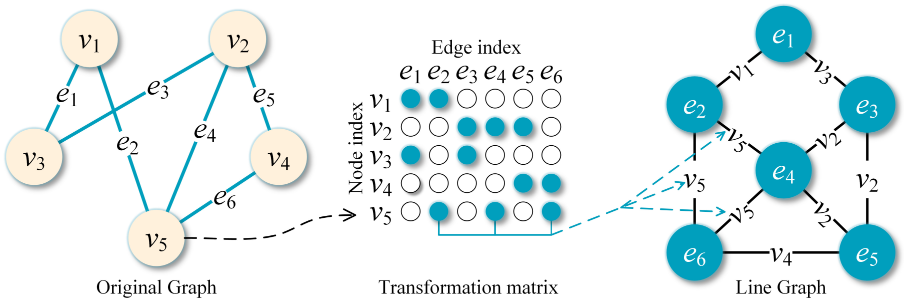


Figure 3. Illustration of the line graph transformation procedure. Each node in the line graph corresponds to a unique edge in the original graph.

As shown in Figure 2(b), by converting original graph G to the line graph $L(G)$, the edge embeddings are updated via convolutional operation to represent high-level edge embeddings. Given a node adjacency matrix A_e of $L(G)$ and adaptive edge embedding H^a, we generate high-level edge embedding H^e, and its learning process can be denoted as:

$$h_k^{(e,l+1)} = \sigma\Big(\sum_{v_i \in \mathcal{N}(k)} \hat{A}_{e(k,i)}(h_i^{(e,l)} + h_i^{(a,l)})W_e^{(l)}\Big), \tag{4.5}$$

where $h_k^{(e,l)}$ is the high-level edge embedding among the neighbor edges $\mathcal{N}(k)$ at the l-th layer with $h_k^{(e,0)} = h_k^{(a,0)}$, $W_e^{(l)}$ is the weight matrix of high-level edge embedding, $\hat{A}_e \in \mathbb{R}^{M\times M}$ is the symmetrically normalized adjacency matrix of the line graph, processed similarly to $\hat{A}_v$. By flexibly using multi-layer conventional operation, high-level edge embedding can perceive a broader range of relationship information. Furthermore, it is crucial to consider the complexity and computational overhead introduced by the line graph transformation and the subsequent edge embedding propagation.

In terms of computational complexity, converting the original graph G to the line graph $L(G)$ involves a transformation that switches the node and edge roles, effectively increasing the number of edges. The line graph adjacency matrix A_e and the subsequent edge-level propagation introduce additional layers of computation. Specifically, the propagation requires computing interactions between neighboring edges, which can increase the computational load, especially for large graphs. The time complexity of the transformation can be considered $O(M^2)$, where M is the number of edges in the graph.

The propagation process in line graph space, as shown in Eq (4.5), utilizes a convolutional operation to update edge embeddings iteratively. While this allows for high-level edge embeddings that can capture complex relationships, it also introduces additional computational steps due to the multiple layers of propagation and the need to compute interactions over all neighboring edges. Thus, while the line graph transformation provides a natural way to model edge relationships, the computational overhead increases as the number of edges grows, and the complexity of propagation operations grows with the depth of the layers.

4.2. *Cross-mapping fusion*

Due to the mismatching between edge embedding $[H^a, H^e] \in \mathbb{R}^{M\times d}$ and node embedding $H^v \in \mathbb{R}^{N\times d}$, it is essential to aggregate the edge embedding into the node space. Therefore, we propose a cross-mapping approach to bridge two semantic spaces, which is shown in Figure 2(c). Cross-mapping allows each node to acquire dual edge embedding $[H^a, H^e]$ from its neighbors, which is expressed as follows:

$$\begin{aligned} h_i^{(a)'} &= \sigma\Big(\sum_{v_j\in\mathcal{N}(i)} \frac{1}{\sqrt{|\mathcal{N}_{(i)}||\mathcal{N}_{(j)}|}} h_{(i,j)}^{(a)}\Big); \\ h_i^{(e)'} &= \sigma\Big(\sum_{v_j\in\mathcal{N}(i)} \frac{1}{\sqrt{|\mathcal{N}_{(i)}||\mathcal{N}_{(j)}|}} h_{(i,j)}^{(e)}\Big); \end{aligned} \tag{4.6}$$

where $[h_i^{(a)'}, h_i^{(e)'}]$ is the dual edge embedding of node v_i. Through the transformation matrix $T \in \mathbb{R}^{N\times M}$, the above aggregation process can be simplified to a convolutional form, denoted as:

$$[H^{a'}, H^{e'}] = \sigma(\widetilde{T}[H^a, H^e]W_t), \tag{4.7}$$

where $[H^{a'}, H^{e'}] \in \mathbb{R}^{N\times d}$ is the dual edge embedding in node space, $\widetilde{T}$ is the standardized transformation matrix T, and W_t is the weight matrix of cross-mapping.

Through the above process, we can obtain tripartite information: node embedding H^v, adaptive edge embedding $H^{a'}$ and high-level edge embedding $H^{e'}$. The final node embedding Z^{agg} is obtained from the cooperation of tripartite information, which is represented as follows:

$$Z^{agg} = g_{task}(H^v \oplus H^{a'} \oplus H^{e'}), \tag{4.8}$$

where $g_{task}(\cdot)$ is an encoder designed for downstream tasks and $\oplus$ is a concatenation operation to prevent confusion of different semantic information.

4.3. *Task-dependent loss functions*

We depict that ELNA-Net is flexible and powerful to learn the embeddings for two graph learning tasks, including semi-supervised node classification and link prediction.

4.3.1. Semi-supervised node classification

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

$$\mathscr{L}(\Theta) = -\sum_{v_i\in V}\sum_{j}^{C} Y_{ij}\log(\widetilde{Z}_{ij}^{agg}), \tag{4.9}$$

where $\widetilde{Z}^{agg}$ is the softmax result of Z^{agg}, 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 $[H^{a'}, H^{e'}]$ to construct two auxiliary classifiers. Specifically, we employ an additional graph convolutional layer $\mathbf{P}$ that encodes $[H^{a'}, H^{e'}]$ into embeddings (Z^a, Z^e). Eventually, the overall objective function is the weighted sum of the three losses,

$$\mathscr{L}_{node} = \mathscr{L}(Z^{agg}, Y) + \beta_1\mathscr{L}(Z^a, Y) + \beta_2\mathscr{L}(Z^e, Y), \tag{4.10}$$

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

4.3.2. Link prediction

We combine ELNA-Net with variational autoencoder (VAE) for link prediction. We introduce stochastic latent variables $R = \{r_1, r_2, ..., r_N\}$, and take a simple inference model parameterized by a 2-layer network:

$$\mathrm{Q}(R \,|\, X, \widetilde{A}_v) = \prod_{i=1}^{N} Q(r_i \,|\, X, \widetilde{A}_v),\ \textit{with}\ Q(r_i \,|\, X, \widetilde{A}_v) = \mathscr{N}(r_i \,|\, \mu_i, diag(\sigma_i^2)), \tag{4.11}$$

where $\mu = Encoder_\mu(X, \widetilde{A}_v)$ is the matrix of mean vectors μ_i and $log_\sigma = Encoder_\sigma(X, \widetilde{A}_v)$. The 2-layer network encoder is defined as $Encoder(X, \widetilde{A}_v) = \widetilde{A}_v RELU(ELNA - Net(X, \widetilde{A}_v, \beta_1, \beta_2)) W_i$, where W_i is the learnable weight matrices for μ and σ. The decoder model follows the design of the original literature [37].

By combing the encoder and decoder models, the loss function of ELNA-VAE is represented as:

$$\mathscr{L}_{link}(\Theta) = \mathbb{E}_{Q(R|X,\widetilde{A}_v)}[\log P(\widetilde{A}_v|R) - KL[Q(R|X, \widetilde{A}_v)||P(R)]], \tag{4.12}$$

where R is the embedding matrix, and $KL[Q(\cdot)||P(\cdot))]$ is the Kullback-Leibler divergence, and we use a Gaussian prior $P(R) = \prod_i P(r_i) = \prod_i N(r_i \,|\, 0, \mathbf{I})$.

Finally, we optimize the model parameters with respect to the objective using stochastic gradient descent. Without loss of generality, the algorithm of ELNA-Net is elaborated as follows.

Algorithm 1: ELNA-Net

```
Input: Original graph $G = (V, E)$; Feature matrix $X$; iterations $epochs$; Hyperparameters $\beta_1$,$\beta_2$;
Output: The node embedding $Z^{agg}$;
Convert the original graph $G$ to the line graph $L(G)$;
Obtain the transformation matrix $T$ and the adjacency matrix of edges $A_e$;
for $epochs$ do
    Encode feature matrix $X$ to obtain node embedding $H^v$ via Eq (4.2);
    Construct adaptive edge embedding of pairwise nodes $H^a$ via Eq (4.3);
    Obtain high-level edge embedding $H^e$ by conventional operation via Eq (4.4);
    Aggregate $[H^a, H^e]$ into node space $[H^{a'}, H^{e'}]$ via Eq (4.6);
    Calculate the node embedding $Z^{agg}$ via Eq (4.8);
    Calculate the corresponding loss and update the parameter set $\Theta$ via Eq (4.10) or Eq (4.12);
end
Return The node embedding $Z^{agg}$;
```

4.4. Theoretical analysis

This theoretical analysis aims to justify why the proposed edge-enhanced representations are intrinsically more discriminative for node classification. Instead of introducing an additional learning objective, we adopt an auxiliary linear classifier as a linear probe to analyze the quality of the learned representations. Specifically, we fix the node representations produced by our edge modeling module and investigate whether correct predictions can be achieved using a linear classifier alone. This setting allows us to study whether the information encoded by edge embeddings is sufficient to induce linear separability among different classes.

Let $h_v \in \mathbb{R}^d$ denote the edge-enhanced embedding of node v generated by our model, and let $\mathbf{P} = [p_1, \ldots, p_C] \in \mathbb{R}^{d\times C}$ denote the parameters of an auxiliary classifier with C classes. The classifier is trained by minimizing the average cross-entropy loss:

$$\min_{\mathbf{P}} \frac{1}{|V|} \sum_{v\in V} L(h_v, y_v; \mathbf{P}), \tag{4.13}$$

where y_v is the ground-truth label of node v.

The predicted probability that node v belongs to class i is given by the softmax function:

$$c_{v,i} = \frac{\exp(p_i^\top h_v)}{\sum_{j=1}^{C} \exp(p_j^\top h_v)}. \tag{4.14}$$

Gradient dynamics of the linear probe. We analyze the gradient of the loss with respect to the classifier parameters. For a node v with label y_v, the gradient $\partial L/\partial p_i$ can be decomposed into two cases: when i corresponds to the correct label and when it does not. By aggregating gradients over all nodes, the update of p_i after one gradient descent step with learning rate η can be written as:

$$p_i^{(T)} = p_i^{(0)} + \frac{\eta}{|V|} \sum_{v\in V, y_v = i} (1 - c_{v,i}) h_v - \frac{\eta}{|V|} \sum_{v \in V, y_v \neq i} c_{v,i} h_v, \tag{4.15}$$

where $p_i^{(0)}$ is initialized from a zero-mean Gaussian distribution. At initialization, the predicted probabilities $\{c_{v,i}\}$ are close to uniform.

The first term in Eq (4.15) encourages p_i to align with the embeddings of nodes belonging to class i, while the second term suppresses the influence of embeddings from other classes. Assuming a sufficiently small learning rate, these updates accumulate over iterations.

Class separation induced by edge embeddings. After $T = \Omega(\eta^{-1} C)$ iterations, the projection of $p_i^{(T)}$ onto the subspace spanned by embeddings of class i reaches a constant order,

$$p_i^{(T)\top} h_v = \Omega(1), \quad \text{for } y_v = i. \tag{4.16}$$

In contrast, the projection onto embeddings of incorrect classes is negative and bounded by a much smaller magnitude,

$$p_i^{(T)\top} h_v \leq -\Omega\left(\frac{1}{C}\right), \quad \text{for } y_v \neq i. \tag{4.17}$$

Therefore, for any node v with ground-truth label j, we have

$$h_v^\top p_j^{(T)} - h_v^\top p_{j'}^{(T)} \geq \Omega(1), \quad \forall j' \neq j. \tag{4.18}$$

This implies that the correct class achieves a strictly larger logit than all incorrect classes, leading to

$$\hat{y}_v = \arg\max_{i\in[C]} c_{v,i} = y_v. \tag{4.19}$$

Implications for edge modeling. The above analysis shows that the edge-enhanced embeddings learned by our model are linearly separable with a constant margin. Compared with node-only representations, incorporating edge-level interactions introduces interaction-aware variations that reduce inter-class overlap and increase intra-class compactness. As a result, even a simple linear classifier can effectively distinguish different categories. This theoretical result provides a principled explanation for the empirical performance gains observed in our edge-level modeling strategy.

5. Experiments

In this section, we comprehensively evaluate ELNA-Net with state-of-the-art models in semi-supervised node classification and link prediction tasks.

Datasets. The experiments are conducted over six real-world datasets which are summarized in Table 2. Cora, Citeseer, and Pubmed are the research paper citation networks [38]. ACM is extracted from ACM dataset [39]. Chameleon and Texas are extracted from Wikipedia network [40].

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
Texas	183	1703	5	309	9/12/18	50
Chameleon	2277	36,101	4	2325	16/28/120	1000

Baselines. In semi-supervised node classification, we compare ELNA-Net with different types of methods: 1) GNA methods: GCN [9], PAGCN [13], Logo-GNN [24], and MOGCN [18]; 2) NLNA methods: GAT [10] and SGAT [12]; 3) Interaction-related (IR) methods: CensNet [27] and GraphAIR [19]. Furthermore, in the link prediction, we compare ELNA-Net with DW [41], GAE [42], VGAE [37], CensNet-VAE [27], GNAE [43], and AIR-GAE [19].

Parameter setting. We train ELNA-Net using the full batch in each training epoch and optimize it by the Adam algorithm with the learning rate of 0.001 to 0.005, the training and test sets are split as shown in Table 2. The hyperparameters β_1, β_2 are selected in the search range $\{0, 0.1, ..., 1\}$. The maximum number of training epochs is 500, and the weight decay is set to 5e-4. The experiment results are obtained by the optimal hyperparameters and the number of iterations.

Evaluation metrics. Following existing metrics [9, 43] in evaluating GNN, we adopt classification accuracy (ACC) to evaluate the performance of baselines and ELNA-Net for node classification. Additionally, we employ area under the ROC curve (AUC) and average precision (AP) for link prediction.

5.1. Semi-supervised node classification

We compare different neighborhood-aware methods in semi-supervised node classification, as shown in Table 3. In each setting, the best-performing results are clearly marked in bold, and we have the following observations.

1. Compared to GNA and NLNA, ELNA-Net has strong competitiveness over most datasets. It can be attributed to the fact that ELNA-Net additionally constructs the interactions between pairwise

nodes to capture the relationship information in the graph.

2. Compared to other Interaction-related methods, ELNA-Net still has superior performance on most datasets. This superiority can be attributed to the fact that ELNA-Net provides comprehensive interactions at both node and edge level. At node level, we maintain the interactions of historical layer to provide the richer context during the learning of interactions. At edge level, we use line graph transformation to explore complex dependencies among edges.
3. Following the failure cases shown in Table 3, taking the Citeseer, and Texas datasets as examples, the performance of ELNA-Net is weaker than that of GraphAIR. It can be attributed to the fact that GraphAIR adds a auxiliary channel to learn interactions. Thus, complementary information between different channels can be introduced into the interactions.

Table 3. ACC (%) of node classification.

	Training	GNA				NLNA		IR		
		GCN	MOGCN	Logo-GNN	PAGCN	GAT	SGAT	CensNet	GraphAIR	ELNA-Net
Cora	14	43.8	**62.8**	59.2	47.6	44.5	45.7	57.7	54.1	62.3
	28	62.3	71.5	73.7	72.1	66.8	67.1	67.1	73.3	**74.5**
	140	81.7	82.4	84.6	83.6	83.1	81.9	79.1	84.2	**84.8**
ACM	15	51.8	68.1	70.3	69.7	56.3	56.2	64.2	68.1	**82.4**
	30	65.6	87.5	86.4	87.2	64.6	71.2	75.9	87.5	**88.5**
	60	87.8	90.1	91.4	90.9	87.4	90.2	89.7	90.7	**91.8**
Citeseer	15	24.7	58.9	55.8	60.4	33.1	28.1	57.6	55.3	**62.5**
	30	43.6	62.8	62.3	64.7	58.4	49.3	62.5	63.5	**65.2**
	160	70.4	72.4	73.4	70.4	72.6	70.6	68.1	**73.5**	72.5
Pubmed	15	43.9	63.2	66.8	63.5	42.6	40.2	61.4	59.6	**71.9**
	30	60.5	68.5	74.1	73.5	56.6	62.4	65.7	68.3	**77.9**
	60	79.0	79.2	80.9	79.3	79.1	80.2	69.9	80.1	**81.4**
Texas	9	42.2	46.2	52.6	53.9	52.3	54.1	55.9	53.2	**57.2**
	12	52.7	51.5	57.7	55.6	54.8	58.3	60.9	60.3	**61.6**
	18	54.6	57.6	48.6	65.3	57.6	62.6	64.3	**67.7**	67.5
Chameleon	16	23.6	25.5	26.5	27.7	27.1	26.3	27.4	24.6	**28.2**
	28	31.4	30.2	31.4	31.9	28.2	29.2	**32.5**	31.8	31.5
	120	47.6	46.9	49.7	49.0	27.9	32.2	47.6	48.5	**50.2**

5.2. *Link prediction*

We benchmark the three citation graphs for link prediction: Cora, Citeseer, and Pubmed. The link prediction results are shown in Table 4, and the following conclusions can be summarized.

1. Compared to the baselines, we can observe that the ELNA-VAE outperforms other methods on most datasets, which once again verifies the necessity of incorporating the edge-level information to neighborhood awareness.
2. It is clear from the Table 4 that the performance of ELNA-VAE obtains obvious improvements, compared with AIR-GAE based on the node-level interactions. It can be attributed to the fact that ELNA-VAE leverages the line graph space transformation to perceive additional edge-level relationship information, enhancing the performance in edge-relevant tasks.
3. It is worth noting that ELNA-VAE performs better than CensNet that also uses the line graph

transformation. This can be attributed to the fact that the proposed cross-mapping approach can effectively bridge different semantic spaces to integrate interactions into node embeddings.

Table 4. AUC (%) and AP (%) of link prediction.

		DW	GAE	VGAE	CensNet-VAE	GNAE	AIR-GAE	ELNA-VAE
Cora	AUC	83.1	91.0	91.4	91.7	92.6	93.3	**94.1**
	AP	85.0	89.5	92.6	92.6	93.6	92.9	**95.4**
Citeseer	AUC	80.5	89.4	90.8	90.6	**94.6**	93.5	93.6
	AP	83.6	90.3	92.0	91.6	**94.8**	94.3	94.7
Pubmed	AUC	84.2	96.4	94.4	95.5	96.4	95.6	**97.4**
	AP	84.1	96.1	94.7	95.9	96.3	96.7	**98.3**

5.3. Ablation study

We also perform the following ablation studies to verify the contribution of each component and the effectiveness of interaction construction methods within ELNA-Net in this subsection.

The contribution of components. Essentially, the final node representation of ELNA-Net is obtained by integrating three components: node information (NI), adaptive edge embedding (AEE) and high-level edge embedding (HEE). To explore the contributions of components, ELNA-Net selects different components to construct the variants, such as ELNA-w/o-A represents that AEE is removed from ELNA-Net and ELNA-w/o-NA represents that NI, and AEE are removed from ELNA-Net. The results of node classification are shown in Table 5, and we have the following observations.

1. Considering only a single component of ELNA-Net, ELNA-w/o-NH has a better performance than ELNA-w/o-AH. This superior performance indicates that the AEE can effectively assist learners in capturing the interaction pattern between pairwise nodes. Furthermore, the performance of ELNA-w/o-NA has significantly decreased compared to those of ELNA-w/o-NH and ELNA-w/o-AH. It can be attributed to the fact that the HEE is obtained via convolution operation in the line graph, which mainly focuses on the relationship information among edges. Therefore, ELNA-w/o-NA is not suitable for independently performing node classification.
2. Considering pairwise components of ELNA-Net, we can observe that the performance of ELNA-w/o-N and ELNA-w/o-A are superior to that of ELNA-w/o-H. This superiority highlights the auxiliary role of HEE, which explores the relationship information among edges to learn the comprehensive node representation.

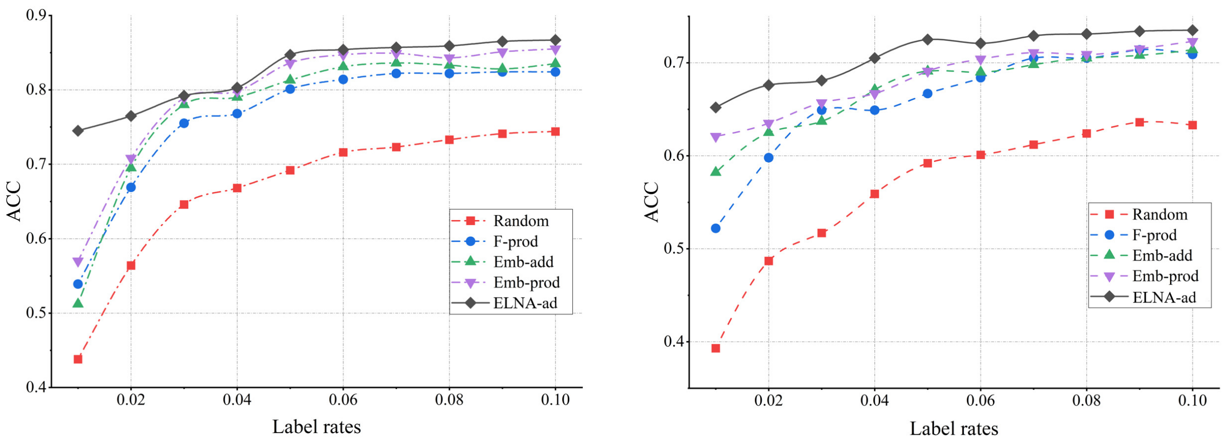
Table 5. ACC (%) of node classification with different combinations.

	NI	AEE	HEE	Cora	ACM	Citeseer	Pubmed
ELNA-w/o-AH	✓			81.7	87.8	70.4	79.0
ELNA-w/o-NH		✓		82.4	88.3	70.8	80.8
ELNA-w/o-NA			✓	68.5	74.1	50.1	64.8
ELNA-w/o-N		✓	✓	83.1	91.1	71.7	81.1
ELNA-w/o-A	✓		✓	83.6	90.4	72.4	80.6
ELNA-w/o-H	✓	✓		83.2	90.8	71.7	80.9
ELNA-Net	✓	✓	✓	**84.3**	**91.4**	**72.5**	**81.4**

The effectiveness of interaction construction methods. We select various construction methods of edge embedding [34, 35] to evaluate the effectiveness of our approach. Since existing methods mainly consider the interaction at node level, we only use adaptive edge embedding for comparison to ensure experimental fairness:

1. **Random** generates random vector to represent interactions without relying on features;
2. **F-prod** is the original construction method, which represents inner product between pairwise node features as the interactions;
3. **Emb-add** leverages the addition of neighbors' embeddings to construct interactions;
4. **Emb-prod** is the popular construction method, which expresses interactions via inner product between pairwise node embeddings;
5. **ELNA-ad** is our construction method of adaptive edge embedding that additionally considers the interactions of historical layer.

The comparison results are shown in Figure 4, and we have the observations that ELNA-ad has a better performance than other methods across various label rates. It is worth noting that while the baseline performance rapidly deteriorates with a decrease in label rate, ELNA-ad performs exceptionally well even at extremely low label rates. This is because ELNA-ad considers the information of both current layer and historical layer, which is more effective in capturing the context during the learning.



(a) Cora (b) Citeseer

Figure 4. The performance comparisons of interaction construction methods in node classification on Cora and Citeseer.

5.3.1. Comparison of different fusion mechanisms

Based on the aggregation process (e.g., Eq [4.7]) , we compared the impact of different fusion mechanisms on experimental results. In the Table 6, the specific meanings of different fusion mechanisms are as follows:

1. **Contact** fuses the embeddings by directly summing them, with no weighting or interpolation;
2. **Attn-Fuse** employs an attention mechanism to assign dynamic weights to each embedding, fusing them through a weighted sum, giving more importance to relevant embeddings;

3. **Gate-Fuse** uses a gating mechanism to selectively adjust the contribution of each embedding, applying interpolation to flexibly combine the embeddings;
4. **CM-Fuse** fuses the embeddings by using cross-mapping, establishing interactive relationships between different embeddings to extract more complex features, thus improving the fusion quality.

Table 6. ACC (%) of node classification with different fusion mechanisms.

	Concat	Attn-fuse	Gate-fuse	CM-fuse
Cora	80.4	81.2	82.5	**84.8**
Citeseer	69.2	70.5	71.4	**72.5**
Pubmed	78.3	78.6	79.2	**81.4**
ACM	86.2	88.4	89.5	**91.8**

Based on the results in Table 6, the cross-mapping fusion mechanism shows the best performance. This can be attributed to its ability to capture complex relationships between different embedded vectors, improving feature complementarity and the model's expressive power. By integrating information from various data sources more precisely, CM-Fuse enhances overall performance. Additionally, it minimizes information loss, contributing to the model's robustness and stability.

5.4. Parameter sensitivity

In this subsection, we attempt to investigate how the hyperparameters of loss function (i.e., Eq [4.10]) impact the model performance. To this end, we conduct experiments on Cora with β_1 and β_2, where $\beta_1, \beta_2 \in \{0, 0.1, ..., 1\}$. We report the node classification performance in terms of accuracy, and the corresponding results are shown in Figure 5.

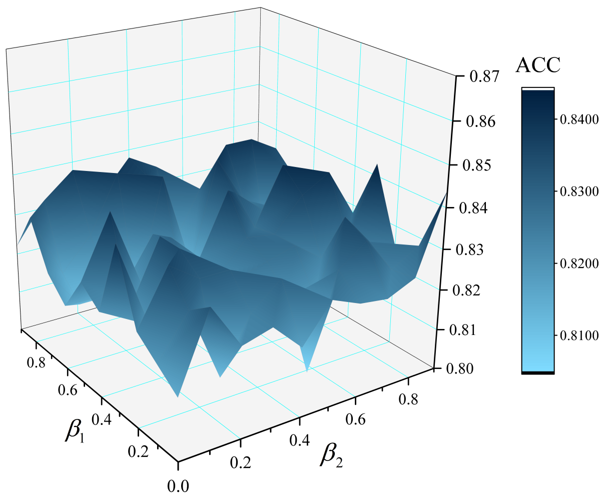


Figure 5. ACC of node classification on Cora with varying hyperparameters.

The results show that the variation of accuracy is irregular with two hyperparameters, and the range of variation is between 0.81 and 0.85. The areas with higher accuracy (darker color) are mainly concentrated in the fringe areas, that is, the combination of two parameters with larger differences can suggest better performance. In addition, our experimental results reveal a notable performance drop in the middle region. This phenomenon arises because the two parameters function as weight coefficients for two edge-embedding-based auxiliary classifiers. The observed performance degradation suggests a mutual exclusivity between these two edge embeddings, which stems from their architectural relationship: one embedding aggregates the outputs of the other. This hierarchical dependency creates an imbalance in learning dynamics, making the system prone to either overfitting or underfitting if the optimal weight range is not properly constrained. Consequently, the asynchronous learning behavior between the classifiers highlights the necessity of prioritizing a single auxiliary classifier to maintain model stability.

Furthermore, we use the random forest regression model to evaluate the importance of two hyper-parameters, as shown in Figure 6. Our results reveal that the importance of adaptive node interaction parameter β_1 is slightly greater than high-level edge interaction parameter β_2.

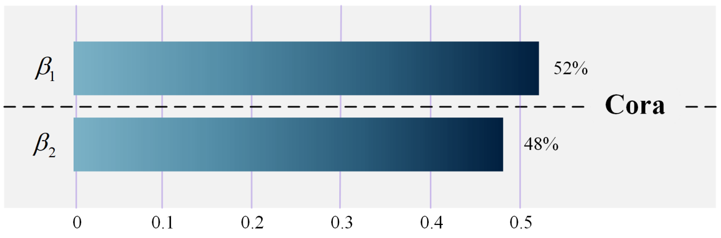


Figure 6. Parameter importance assessment.

5.5. *Visualization*

As shown in Figures 7 and 8, we visualize the graph embeddings learned on Cora. GraphAIR and CensNet are selected for comparison because GraphAIR introduces node interactions to enrich neighborhood awareness, while CensNet exploits line graph space transformation to obtain edge embeddings. We employ two types of visual analysis to demonstrate the distribution and quality of node representations. Firstly, we utilize t-SNE to provide the distribution of node representations $\widetilde{Z}^{agg}$, as shown in Figure 7. Secondly, we show the visualization of similarity matrix $\widetilde{Z}^{agg}\widetilde{Z}^{agg^T}$ to analyze the quality of node representations, as shown in Figure 8.

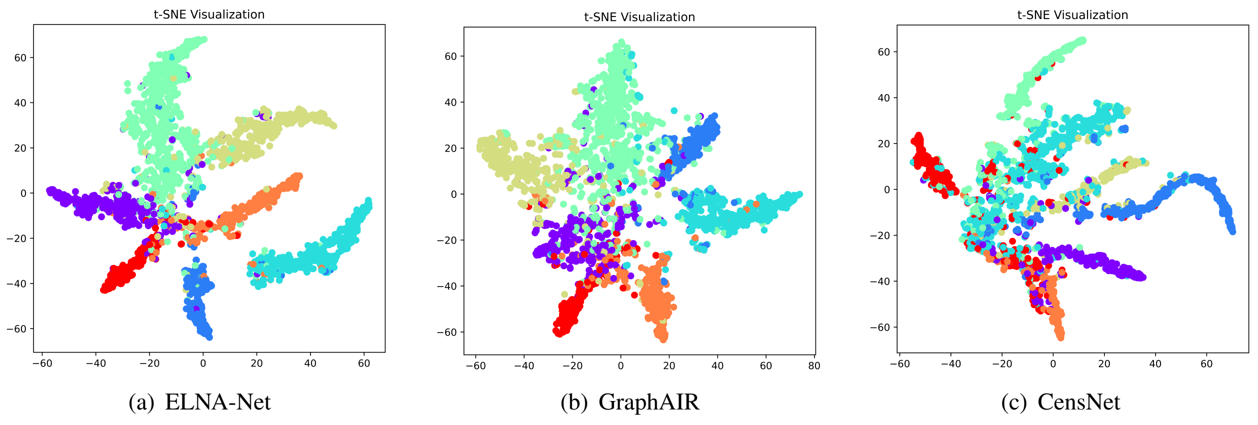
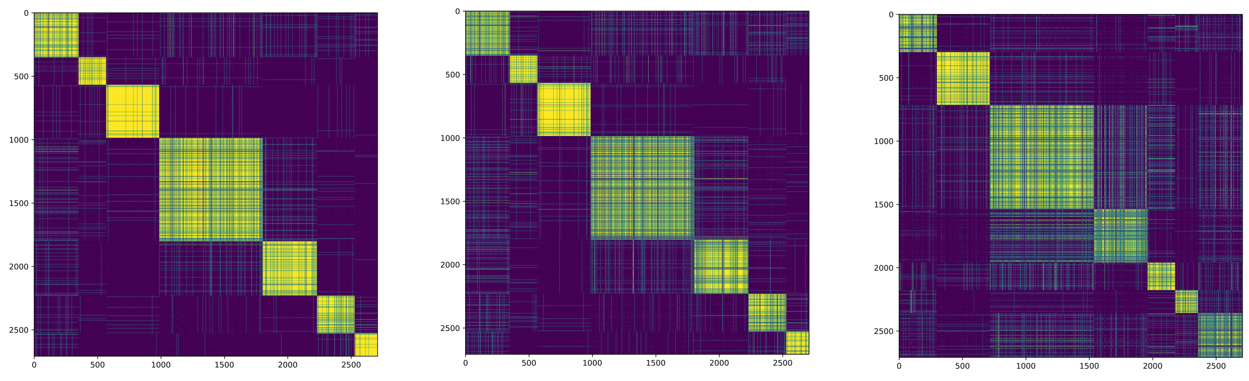


Figure 7. Visualization comparison of node representation distribution on Cora.

It is clear from Figure 7(a) that integrating dual interaction makes samples in the same class more clustered and samples in different classes more distant from each other. In addition, ELNA-Net exhibits the most optimal embedding, the highest intra-class similarity, and the clearest distinct boundaries among different classes.

As shown in Figure 8, we further analyze the quality of node representation. We can observe that the similarity matrix of ELNA-Net has more complete diagonal matrix blocks, indicating higher similarity intra-class and more accurate node representations.



(a) ELNA-Net (b) GraphAIR (c) CensNet

Figure 8. Visualization comparison of node representation quality on Cora.

6. Application of patient classification for Alzheimer's disease

6.1. Experimental configuration

Dataset. The Alzheimer's disease dataset comes from the ADNI public dataset [44], and its details are shown in Table 7. The dataset consists of 220 Alzheimer's disease (AD) patients, 981 patients with late stage mild cognitive impairment (LMCI) and 642 normal controls (NC). The features of each sample include multi-model measurement indicators, such as cognitive tests, MRI, PET, and CSF. The performance of patient classification is evaluated using a 5-fold cross-validation strategy.

Table 7. Alzheimer's disease dataset.

	Samples	Years in education	Average age	Female/Male
NC	642	15.97	74.73	325/317
LMCI	981	15.52	73.36	387/594
AD	220	14.75	73.78	106/114

Baselines. Some advanced methods are utilized as the baselines for patient classification, which include Ada-boost [45], GCN [9], DGM [46], and DDGFCN [47]. Among them, DDGFCN and DGM are the representative models that use the strategy of dynamically adjusting graph structure. In addition, we closely follow the experimental setting of previous works, and the performance of the baselines is reported from their original papers.

6.2. The binary classification of Alzheimer's disease

We perform individualized diagnoses using different models and summarize the results in Table 8.

Table 8. ACC (%) and AUC (%) of patient classification.

		Ada-boost	GCN	DGM	DDGFCN	**ELNA-Net**
AD vs NC	ACC	90.0	92.1	93.7	95.3	**96.5**
	AUC	86.3	91.8	92.1	94.8	**95.9**
AD vs LMCI	ACC	90.0	88.5	92.5	94.6	**96.6**
	AUC	84.8	89.4	93.3	96.0	**97.2**
LMCI vs NC	ACC	86.7	90.5	92.3	93.0	**94.7**
	AUC	82.8	85.2	91.0	92.1	**93.5**

As shown in Table 8, the results indicate that ELNA-Net achieves the superior performance on binary classification tasks. It is worth noting that the classification between AD and LMCI is a challenging task due to their similar early-stage symptoms. In the AD vs LMCI classification task, the ACC of ELNA-Net is 2% higher than the second-ranked method, which further confirms the effectiveness of ELNA-Net. Therefore, our method can be applied to the patient classification of Alzheimer's disease, demonstrating its flexible capability in real-life application.

7. Conclusions and future work

In this study, we introduce ELNA-Net, a novel GNN framework that addresses the limitations of existing neighborhood-aware methods in capturing relationship information of different levels. ELNA-Net stands out as an effective solution, offering a comprehensive approach from both nodes and edges perspectives. Our experimental evaluations, focusing on node classification and link prediction tasks, showcase the superior performance of ELNA-Net compared to state-of-the-art methods. The application of Alzheimer's disease confirms its effectiveness in real-life scenario. In conclusion, ELNA-Net emerges as a promising solution to enhancing the capability of neighborhood awareness in GNN.

Although this study has made significant progress in different downstream tasks, there are still many issues worthy of further investigation. For example, there remains a lack of a unified theoretical understanding of why these interaction-related methods are effective and how they are related. In future work, we will resort to Taylor interaction effects for explaining the existing IR methods, which attempts to unify methods into a weighted allocation of two typical effects: independent effects and interaction effects.

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 the National Natural Science Foundation of China (62506145) and the Natural Science Foundation of Jiangsu Higher Education Institutions (25KJB520008).

Conflict of interest

The authors declare there is 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/10.1145/3653986

2. G. Xue, M. Zhong, T. Qian, J. Li, PSA-GNN: An augmented GNN framework with priori subgraph knowledge, *Neural Networks*, **173** (2024), 106155. https://doi.org/10.1016/j.neunet.2024.106155

3. W. L. Hamilton, R. Ying, J. Leskovec, Representation learning on graphs: Methods and applications, preprint, arXiv:1709.05584.

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

5. F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, K. Weinberger, Simplifying graph convolutional networks, preprint, arXiv:1902.07153.

6. L. Peng, R. Hu, F. Kong, J. Gan, Y. Mo, X. Shi, et al., Reverse graph learning for graph neural network, *IEEE Trans. Neural Networks Learn. Syst.*, **35** (2024), 4530–4541. https://doi.org/10.1109/TNNLS.2022.3161030

7. 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

8. H. Liu, B. Yang, D. Li, Graph collaborative filtering based on dual-message propagation mechanism, *IEEE Trans. Cybern.*, **53** (2021), 352–364. https://doi.org/10.1109/TCYB.2021.3100521

9. T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, preprint, arXiv:1609.02907.

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

11. H. Zeng, H. Zhou, A. Srivastava, R. Kannan, V. Prasanna, GraphSAINT: Graph sampling based inductive learning method, preprint, arXiv:1907.04931.

12. Y. Ye, S. Ji, Sparse graph attention networks, *IEEE Trans. Knowl. Data Eng.*, **35** (2021), 905–916. https://doi.org/10.1109/TKDE.2021.3072345

13. 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

14. C. Huang, M. Li, F. Cao, H. Fujita, Z. Li, X. Wu, Are graph convolutional networks with random weights feasible?, *IEEE Trans. Pattern Anal. Mach. Intell.*, **45** (2022), 2751–2768. https://doi.org/10.1109/TPAMI.2022.3183143

15. 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

16. Y. Yang, Y. Sun, F. Ju, S. Wang, J. Gao, B. Yin, Multi-graph fusion graph convolutional networks with pseudo-label supervision, *Neural Networks*, **159** (2023), 305–317. https://doi.org/10.1016/j.neunet.2022.11.027

17. Z. Chen, J. Bruna, L. Li, Supervised community detection with line graph neural networks, preprint, arXiv:1705.08415.

18. J. Wang, J. Liang, J. Cui, J. Liang, Semi-supervised learning with mixed-order graph convolutional networks, *Inf. Sci.*, **573** (2021), 171–181. https://doi.org/10.1016/j.ins.2021.05.057

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. B. Jiang, Y. Chen, B. Wang, H. Xu, B. Luo, DropAGG: Robust graph neural networks via drop aggregation, *Neural Networks*, **163** (2023), 65–74. https://doi.org/10.1016/j.neunet.2023.03.022

21. L. Cai, J. Li, J. Wang, S. Ji, Line graph neural networks for link prediction, *IEEE Trans. Pattern Anal. Mach. Intell.*, **44** (2021), 5103–5113. https://doi.org/10.1109/TPAMI.2021.3080635

22. 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

23. Q. Guo, X. Yang, W. Ding, Y. Qian, Cross-graph interaction networks, *IEEE Trans. Knowl. Data Eng.*, **173** (2025), 11059. https://doi.org/10.1109/TKDE.2025.3543377

24. 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

25. X. Wang, S. Trajanovski, R. E. Kooij, P. Van Mieghem, Degree distribution and assortativity in line graphs of complex networks, *Physica A*, **445** (2016), 343–356. https://doi.org/10.1016/j.physa.2015.10.109

26. Z. Zhang, S. Sun, G. Ma, C. Zhong, Line graph contrastive learning for link prediction, *Pattern Recognit.*, **140** (2023), 109537. https://doi.org/10.1016/j.patcog.2023.109537

27. X. Jiang, R. Zhu, S. Li, P. Ji, Co-embedding of nodes and edges with graph neural networks, *IEEE Trans. Pattern Anal. Mach. Intell.*, **45** (2020), 7075–7086. https://doi.org/10.1109/TPAMI.2020.3029762

28. L. Gong, Q. Cheng, Exploiting edge features for graph neural networks, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, (2019), 9211–9219. https://doi.org/10.1109/CVPR.2019.00943

29. W. Song, C. Shi, Z. Xiao, Z. Duan, Y. Xu, M. Zhang, et al., AutoInt: Automatic feature interaction learning via self-attentive neural networks, in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM)*, (2019), 1161–1170. https://doi.org/10.1145/3357384.3357925

30. R. Wang, B. Fu, G. Fu, M. Wang, Deep & cross network for ad click predictions, in *Proceedings of the ADKDD'17*, (2017), 1–7. https://doi.org/10.1145/3124749.3124754

31. S. Zhu, C. Zhou, S. Pan, X. Zhu, B. Wang, Relation structure-aware heterogeneous graph neural network, in *Proceedings of the 23rd International Conference on Data Mining (ICDM)*, (2019), 1534–1539. https://doi.org/10.1109/ICDM.2019.00203

32. Z. Xie, W. Zhang, B. Sheng, P. Li, C. L. Philip Chen, BaGFN: Broad attentive graph fusion network for high-order feature interactions, *IEEE Trans. Neural Networks Learn. Syst.*, **34** (2021), 4499–4513. https://doi.org/10.1109/TNNLS.2021.3116209

33. S. Rendle, Factorization machines, in *Proceedings of the 10th International Conference on Data Mining (ICDM)*, (2010), 995–1000. https://doi.org/10.1109/ICDM.2010.127

34. 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

35. 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** (2022), 9019–9030. https://doi.org/10.1109/TKDE.2022.3220789

36. 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

37. T. N. Kipf, M. Welling, Variational graph auto-encoders, preprint, arXiv:1611.07308.

38. A. Bojchevski, S. Günnemann, Deep Gaussian embedding of graphs: Unsupervised inductive learning via ranking, preprint, arXiv:1707.03815.

39. X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, et al., Heterogeneous graph attention network, in *Proceedings of The World Wide Web Conference (WWW)*, 2019. https://doi.org/10.1145/3308558.3313562

40. H. Pei, B. Wei, K. Chang, Y. Lei, B. Yang, Geom-GCN: Geometric graph convolutional networks, preprint, arXiv:2002.05287.

41. B. Perozzi, R. Al-Rfou, S. Skiena, DeepWalk: Online learning of social representations, in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (2014), 701–710. https://doi.org/10.1145/2623330.2623732

42. J. Schulman, P. Moritz, S. Levine, M. I. Jordan, P. Abbeel, High-dimensional continuous control using generalized advantage estimation, preprint, arXiv:1506.02438.

43. S. J. Ahn, M. Kim, Variational graph normalized autoencoders, in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management (CIKM)*, (2021), 2827–2831. https://doi.org/10.1145/3459637.3482215

44. C. R. Jack Jr, M. A. Bernstein, N. C. Fox, P. Thompson, G. Alexander, D. Harvey, et al., The Alzheimer's disease neuroimaging initiative (ADNI): MRI methods, *J. Magn. Reson. Imaging*, **27** (2008), 685–691. https://doi.org/10.1002/jmri.21049

45. V. Sanjay, P. Swarnalatha, An enhanced approach for detecting Alzheimer's disease, in *Proceedings of the 2022 Smart Technologies, Communication and Robotics (STCR)*, (2022), 1–5. https://doi.org/10.1109/STCR55312.2022.10009274

46. A. Kazi, L. Cosmo, S. A. Ahmadi, N. Navab, M. M. Bronstein, Differentiable graph module (DGM) for graph convolutional networks, *IEEE Trans. Pattern Anal. Mach. Intell.* , **45** (2022), 1606–1617. https://doi.org/10.1109/TPAMI.2022.3170249

47. F. Li, Z. Wang, Y. Guo, C. Liu, Y. Zhu, Y. Zhou, et al., Dynamic dual-graph fusion convolutional network for Alzheimer's disease diagnosis, in *Proceedings of the 2023 International Conference on Image Processing (ICIP)*, (2023), 675–679. https://doi.org/10.1109/ICIP49359.2023.10222732