Dual-interaction joint awareness for graph neural network

Wenrui Guan[a,b], Keyu Liu[a], Qihang Guo[a], Xibei Yang[a,*], Hengrong Ju[c], Yuhua Qian[d]

[a]*School of Computer, Jiangsu University of Science and Technology, Zhenjiang, Jiangsu 212100, China*
[b]*School of Automation, Jiangsu University of Science and Technology, Zhenjiang, Jiangsu 212100, China*
[c]*School of Artificial Intelligence and Computer Science, Nantong University, Nantong, Jiangsu 226019, China*
[d]*Institute of Big Data Science and Industry, Shanxi University, Taiyuan, Shanxi 030006, China*

Abstract

Graph Neural Network (GNN) has been widely studied to handle graph-structured data. Neighborhood awareness, as a mechanism for integrating context in graph, plays an important role in learning node embeddings of GNN. Currently, a promising direction is to explicitly model the interaction between pairwise nodes for enriching neighborhood awareness. However, such a strategy faces two challenges: 1) interactions are mostly designed manually which lacks adaptive capabilities, making them unsuitable for complex learning scenarios; 2) interactions are primarily acquired from the node relationships, failing to perceive high-level structural information beyond the node level. In this study, we propose a Dual-Interaction Neighborhood Awareness Network (DinaNet) that integrates two modules of adaptive node interaction and high-level edge interaction for more comprehensive neighborhood awareness. Specifically, adaptive node interaction module leverages node embeddings of current layer and interactions of historical layer to learn node relationships. Furthermore, high-level edge interaction module switches the role of nodes and edges, thus it can reveal the potential dependencies among neighbors' edges to promote the level of neighborhood awareness. Finally, we propose a cross-mapping approach to fuse node embeddings and dual interaction in the operator of graph convolution. Experimental results demonstrate that DinaNet achieves excellent performance in semi-supervised node classification and link prediction compared with the state-of-the-art models. The source code of DinaNet is available at: https://github.com/chen0xixi/DinaNet.

Keywords: Graph neural networks, Neighborhood awareness, Interaction, Line graph

1. Introduction

In recent years, there has been a surge in approaches that handle graph-structured data via Graph Neural Network (GNN) Khoshraftar (2024); Xue (2024); Jiang (2023). Neighborhood awareness is one of the critical mechanisms affecting the performance of GNN Hamilton (2017); Phan (2023); Zhu (2019). In terms of aggregating node information, the existing neighborhood awareness can be classified into two main categories Guang (2024): Greedy Neighborhood

*Corresponding author
Email addresses: wrguan@stu.just.edu.cn (Wenrui Guan), kyliu@just.edu.cn (Keyu Liu), 211110701107@stu.just.edu.cn (Qihang Guo), jsjxy_yxb@just.edu.cn (Xibei Yang), juhengrong@ntu.edu.cn (Hengrong Ju), jinchengqyh@126.com (Yuhua Qian)

Awareness (GNA) and Node-Level Neighborhood Awareness (NLNA). GNA receives node information from neighbors without distinguishing the importance of nodes Wang (2020); Huang (2023). As a typical representative GNA, Graph Convolutional Network (GCN) integrates graph structure with node features, and extends the range of neighborhood awareness to multi-hop neighbors by layer-wise propagation Kipf (2017). In contrast, NLNA selectively aggregates node information from relevant neighbors via attention Peng (2024) or sampling Wang (2024) mechanisms. For example, Graph Attention Network (GAT) calculates different attention coefficients to represent the correlation of nodes, thus flexibly capturing local structural information Veličković (2018); GraphSAGE employs node-wise sampling to control the range of neighborhood awareness for the sake of computational complexity Zeng (2019).

Despite the remarkable performance achieved by the emerging neighborhood-aware methods, they do not comprehensively utilize the relationship information Phan (2023), since they just follow the general aggregation strategy Wu (2019). Such simple combination of neighbors without considering the interactions between nodes limits the capability of capturing intricate relational dependencies in graphs Yan (2024). Therefore, Interactive Neighborhood Awareness (INA), referring to harnessing the interactions for neighborhood awareness, has been developed as a promising solution Gao (2021). Essentially, INA figures out how to effectively represent invisible interactions Hu (2021). The construction methods of interactions mainly follow the paradigm of factorization machine Rendle (2010) where the interaction between pairwise nodes is constructed as the inner product of latent vectors Lian (2018); Guo (2017). For instance, AFN Cheng (2020) leverages logarithmic space transformation to denote vector addition as interaction; DeepFM Guo (2017) uses FM to extract feature interactions and feed them into deep neural networks to improve performance; AutoInt Song (2019) adopts a multi-head self-attentive neural network with residual connections to expresses interactions as feature combinations of different orders.

However, the aforementioned interactions are designed manually, which lack adaptive capabilities in complex learning scenarios. On the one hand, manual interaction relies heavily on the prior knowledge, so it is difficult to be generalized to unfamiliar interaction patterns. On the other hand, manual interaction only considers the node embeddings of the current layer and ignores interactions of historical layer, which cannot adapt to variable node relationships during the process of learning. In this case, blindly introducing interactions with limited adaptive capabilities can backfire on the performance of learners. Furthermore, the majority of INA depends only on the connectivities at node level to extract interactions, while ignoring the relationship information among the edges. This omission poses a significant deviation in edge-relevant tasks, particularly link prediction. Therefore, it is desired to develop a method capturing interactions adaptaively and comprehensively to enhance the neighborhood awareness.

To this end, we propose a Dual-Interaction Neighborhood Awareness Network (DinaNet) — a novel GNN framework that integrates two modules: adaptive node interaction and high-level edge interaction. Specifically, in the adaptive node interaction module, we fuse node embeddings of current layer and node interactions of historical layer which are originally derived by factorization machine to extract the adaptive interactions. In the high-level edge interaction module, we leverage line graph space transformation Cai (2021) to express the edge interactions. 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 module to aggregate dual interaction into node embeddings, achieving joint neighborhood awareness of edge-level and node-level interactions. The main contributions can be summarized as follows.

(1) Considering that the manual interaction lacks flexible capabilities, an adaptive node interaction is designed by fusing node embeddings of current layer and node interactions of historical layer, which can be automatically learned for various real-life scenarios.
(2) To promote the level of neighborhood awareness, a high-level edge interaction is devised that adopts a node-edge switching strategy to reveal the intricate relational dependencies in graphs.
(3) To provide comprehensive interactions on both node and edge level, a cross-mapping module is proposed to integrate dual interactions into node embeddings.
(4) We conduct extensive experiments on six datasets in both semi-supervised node classification and link prediction tasks, and the experimental results demonstrate that DinaNet achieves state-of-the-art performance.

2. Related work

Graph neural network (GNN) has achieved tremendous success in graph-structured data learning. The pivotal mechanism determining the effectiveness of a GNN is neighborhood awareness. 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 Dwivedi (2023); Kazi (2022). Typically, GCN combines a symmetric Laplacian smoothing into the operation of node aggregation Huang (2023). Recently, with respect to various requirements, enumours advanced aggregation operations have been developed. For example, GraphSAGE firstly 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 Zeng (2019); GAT utilizes an attention-based aggregator that generates different coefficients to represent the correlation of neighbors Veličković (2018); SGAT learns sparse attention coefficients under an L_0-norm regularization, which can identify task-irrelevant edges and perform aggregation Ye (2021). Furthermore, it is worth noting that following the popular information fusion thought, node aggregation operation can also be introduced into some combinational GNN frameworks Wang (2021); Yang (2023); Teng (2024). For instance, PAGCN fuses the node aggregations from the perspective of both augmentation graph and raw graph Guo (2024); MAGCN attentively fuses the multiple node aggregations which are derived by multiple trustable topologies Yao (2022). Although node aggregation has achieved outstanding successes in various tasks, the interaction between pairwise nodes is rarely considered. Such an oversight highly limits the capabilities of capturing intricate relational dependencies, thereby greatly compromising the performance of GNN.

2.2. Interaction

In order to capture more refined relationship information, a few researcheres Jiang (2020) have attempted to exploit the interactive neighborhood awareness Zhu (2019). The interactive neighborhood awareness explicitly constructs the interactions based on the paradigm of factorization machine Xie (2018), and then combines interactions with the raw node embedding Rendle

(2010); Wang (2017). For instance, AFN Cheng (2020) transforms feature embeddings into a logarithmic space and constructs arbitrary-order feature interactions through feedforward architecture; DFI-GCN Zhao (2022) extracts the arbitrary-order interactions between different features via Newton's identities, and integrates interactions into node embeddings using attention mechanism; GraphAIR Hu (2021) takes into account the interactions between nodes in different channels, thus it represents the interactions as the inner product of node embeddings between two channels. However, the above interactions are 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, $V = \{v_1, v_2, ..., v_N\}$ 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 node interaction
H^e	High-level edge interaction
$\mathcal{N}$	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:

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

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:

$$h_i^{\{l+1\}} = \sigma(W^{\{l\}} \cdot AGGREGATE(\{h_i^{\{l\}}\} \cup \{h_j^{\{l\}}, v_j \in \mathcal{N}_s(i)\})), \tag{2}$$

where $AGGREGATE$ is a aggregation function, which commonly includes that 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:

$$h_i^{\{l+1\}} = \sigma\Big(\sum_{v_j \in \mathcal{N}(i)} a_{ij}^{\{l\}} h_j^{\{l\}} W^{\{l\}}\Big), \tag{3}$$

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

4. Method

DinaNet differs from traditional INA in three main aspects. 1) Adaptive node interactions are designed by fully considerng node embeddings of current layer and node interactions of historical layer, which can effectively capture relationship information of node level. 2) High-level edge interactions are explored via line graph space transformation, which can reveal the potential dependencies at edge level. 3) A cross-mapping approach is proposed to integrate dual interaction into node embedding in the operator of graph convolution. The pseudocode is provided in Algorithm 1, and the framework is displayed in Fig. 1.

The following sections are structured to present the details and implementation of DinaNet. In Section 4.1, we present the process of constructing dual interaction: adaptive node interaction between pairwise nodes and high-level edge interaction among neighbors' edges. In Section 4.2, we illustrate the details of cross-mapping which integrates dual interaction into node embedding. In Section 4.3, we introduce the loss functions of DinaNet to different downstream tasks: semi-supervised node classification and link prediction.

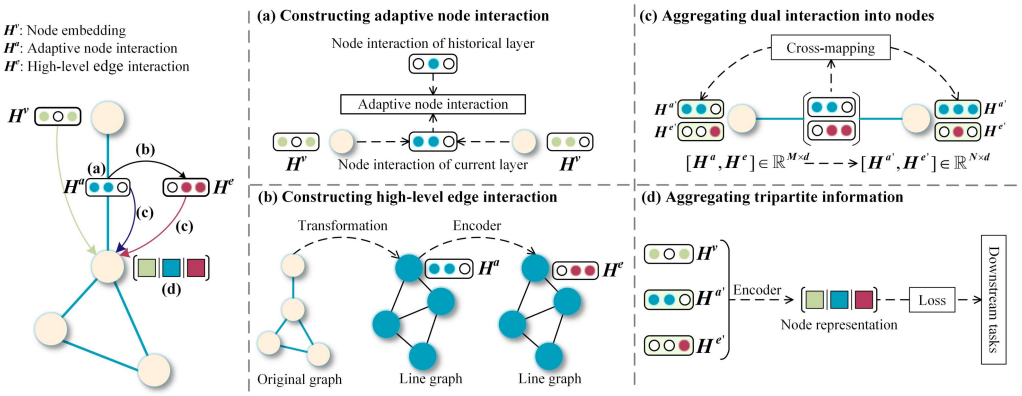


Figure 1: Overview framework of DinaNet. Module (a) constructs the adaptive node interaction by fusing the node interaction of current layer and historical layer. Module (b) explores the high-level edge interaction through converting the original graph to the line graph. Module (c) aggregates dual interaction into the node embedding via the cross-mapping. Module (d) integrates tripartite information to jointly encode the node representation.

4.1. The construction of dual interaction

INA aims to effectively represent the interactions to enhance the expressive power of GNN. As graph data becomes increasingly complex, manual interactions primarily rely on node embeddings of current layer, resulting in the lack of adaptive capabilities. Furthermore, the level of INA is undoubtedly limited because existing methods fail to capture the interactions among edges. Therefore, to provide comprehensive interactions at both node and edge level, we design two types of interaction: adaptive node interaction and high-level edge interaction.

4.1.1. Adaptive node interaction

As mentioned above, we design a novel interaction construction method at node level, as shown in Fig. 1(a). By extracting node interactions of historical layer and node embeddings of current layer, adaptive node interaction can be learned. In the following, we provide a detailed procedure of the adaptive node interaction.

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

$$h_i^{\{v,l+1\}} = \sigma\Big(\sum_{v_j \in \mathcal{N}(i)} \hat{A}_{v(i,j)} h_j^{\{v,l\}} W_v^{\{l\}}\Big), \tag{4}$$

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.

In order to comprehensively perceive node relationships, DinaNet 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 node interaction H^a is defined as follows:

$$h_{(i,j)}^{\{a,l+1\}} = \sigma\Big(\Big[h_{(i,j)}^{\{a,l\}} \oplus (h_i^{\{v,l+1\}} \odot h_j^{\{v,l+1\}})\Big] S^T\Big), \tag{5}$$

where $h_{(i,j)}^{\{a,l\}}$ is the adaptive node interaction 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.

4.1.2. High-level edge interaction

As analyzed earlier, it is important to consider both node-level and edge-level interactions. Fortunately, the line graph transformation, which switches the role of nodes and edges, provides a natural approach for learning the interactions with edges as the primary focus Wang (2016); Cai (2021). As shown in Fig. 2, we illustrate the process of the 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 $h^{\{a,l\}}_{(i,j)}$ becomes $h^{\{a,l\}}_{k}$.

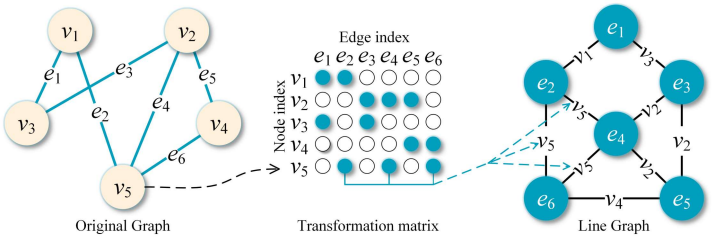


Figure 2: Illustration of the line graph transformation procedure. Each node in the line graph corresponds to a unique edge in the original graph.

As shown in Fig. 1(b), by converting original graph G to the line graph $L(G)$, the edge embeddings are acuquired via convolutional operation to represent high-level edge interactions. Given a node adjacency matrix A_e of $L(G)$ and adaptive node interaction H^a, we generate high-level edge interaction H^e, and its learning process can be denoted as:

$$h^{\{e,l+1\}}_{k} = \sigma\Big(\sum_{v_i \in \mathcal{N}(k)} \hat{A}_{e(k,i)}(h^{\{e,l\}}_{i} + h^{\{a,l\}}_{i})W^{\{l\}}_{e}\Big), \tag{6}$$

where $h^{\{e,l\}}_{k}$ is the high-level edge interaction among the neighbor edges $\mathcal{N}(k)$ at the l-th layer with $h^{\{e,0\}}_{k} = h^{\{a,0\}}_{k}$, $W^{\{l\}}_{e}$ is the weight matrix of high-level edge interaction. $\hat{A}_e \in \mathbb{R}^{M \times M}$ 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 interaction can perceive a broader range of relationship information.

4.2. The aggregation from dual interaction to node embedding

Due to the mismatching between the dual interaction $[H^a, H^e] \in \mathbb{R}^{M \times d}$ and node embedding $H^v \in \mathbb{R}^{N \times d}$, it is essential to aggregate the dual interaction into the node space. Therefore, we propose a cross-mapping approach to bridge two semantic spaces, which is shown in Fig. 1(c). Cross-mapping allows each node to acquire dual interaction $[H^a, H^e]$ from its neighbors, which is expressed as follows:

$$[h^{\{a\}'}_{i}, h^{\{e\}'}_{i}] = \sigma\Big(\sum_{v_j \in \mathcal{N}(i)} \alpha_{i,j}[h^{\{a\}}_{(i,j)}, h^{\{e\}}_{(i,j)}]\Big), \tag{7}$$

where $[h^{\{a\}'}_{i}, h^{\{e\}'}_{i}]$ is the dual interaction of node v_i, and $\alpha_{i,j}$ is the aggregation weight between node v_i and v_j. Through the transformation matrix $T \in \mathbb{R}^{N \times M}$, the above aggregation process can be simplified to a convolutional form, denoted as:

$$[H^{a'}, H^{e'}] = \sigma(\widetilde{T}[H^a, H^e]W_t), \tag{8}$$

where $[H^{a'}, H^{e'}] \in \mathbb{R}^{N\times d}$ is the dual interaction 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 node interaction $H^{a'}$ and high-level edge interaction $H^{e'}$. The final node embedding Z^{agg} is obtained from the cooperation of tripartite information, which is represented as follows:

$$Z^{agg} = g_{task}(H^v \oplus H^{a'} \oplus H^{e'}), \tag{9}$$

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 DinaNet 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 fuction $\mathscr{L}$ can be represented as follows:

$$\mathscr{L}(\Theta) = -\sum_{v_i\in V}\sum_{j}^{C} Y_{ij}\log(\widetilde{Z}^{agg}_{ij}), \tag{10}$$

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

To obtain more accurate node embeddings, we leverage dual interaction $[H^{a'}, H^{e'}]$ to construct two auxiliary classifiers. Specifically, we employ an additional graph convolutional layer that encodes dual interaction $[H^{a'}, H^{e'}]$ into embeddings (Z^a, Z^e). Eventually, the overall objective function is the weighted sum of the three losses:

$$\mathscr{L}_{node}(\Theta) = -\sum_{v_i\in V}\sum_{j}^{C}[Y_{ij}\log(\widetilde{Z}^{agg}_{ij}) + \beta_1 Y_{ij}\log(Z^a_{ij}) + \beta_2 Y_{ij}\log(Z^e_{ij})], \tag{11}$$

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 DinaNet with Variational Autoencoder (VAE) to construct the variant DinaNet-VAE for link prediction Ahn (2021). We introduce stochastic latent variables $R = \{r_1, r_2, ..., r_N\}$, and take a simple inference model parameterized by a 2-layer network:

$$Q(R\,|\,X, \widetilde{A}_v) = \prod_{i=1}^{N} Q(r_i\,|\,X, \widetilde{A}_v),\ with\ Q(r_i\,|\,X, \widetilde{A}_v) = \mathscr{N}(r_i\,|\,\mu_i, diag(\sigma_i^2)), \tag{12}$$

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 $Encoder(X, \widetilde{A}_v) = \widetilde{A}_v RELU(DinaNet(X, \widetilde{A}_v, \beta_1, \beta_2))W_i$,

where W_i is the learnable weight matrices for μ and σ. The decoder model follows the design of the original literature Kipf (2016).

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

$$\mathscr{L}_{link}(\Theta) = \mathbb{E}_{Q(R|X,\widetilde{A}_v)}[\log P(\widetilde{A}_v|R) - KL[Q(R|X,\widetilde{A}_v)\|P(R)]], \tag{13}$$

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 DinaNet is elaborated as follows.

Algorithm 1: DinaNet

Input: Original graph $G = (V, E)$; Feature matrix X; iterations $epochs$;
Hyperparameters β_1, β_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);
  Construct adaptive node interaction of pairwise nodes H^a via Eq. (5);
  Obtain high-level edge interaction H^e by conventional operation via Eq. (6);
  Aggregate dual interaction $[H^a, H^e]$ into node space $[H^{a'}, H^{e'}]$ via Eq. (8);
  Calculate the node embedding Z^{agg} via Eq. (9);
  Calculate the corresponding loss and update the parameter set Θ via Eq. (11) or Eq. (13);
end
Return The node embedding Z^{agg};

5. Experiments

In this section, we comprehensively evaluate DinaNet 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 Bojchevski (2017). ACM is extracted from ACM dataset Wang (2019). Chameleon and Texas are extracted from Wikipedia network Pei (2020).

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	19717	44338	3	500	15/30/60	1000
ACM	3025	26256	3	1870	15/30/60	1000
Texas	183	1703	5	309	9/12/18	50
Chameleon	2277	36101	4	2325	16/28/120	1000

Baselines. In semi-supervised node classification, we compare DinaNet with different types of methods: 1) GNA methods: GCN Kipf (2017), PAGCN Guo (2024), and MOGCN Wang

(2021); 2) NLNA methods: GAT Veličković (2018) and SGAT Ye (2021); 3) INA methods: CensNet Jiang (2020) and GraphAIR Hu (2021). Furthermore, in the link prediction, we compare DinaNet with DW Perozzi (2014), GAE Kipf (2016), VGAE Kipf (2016), CensNet-VAE Jiang (2020), GNAE Ahn (2021) and AIR-GAE Hu (2021).

Parameter setting. We train DinaNet 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 Kipf (2017); Ahn (2021) in evaluating GNN, we adopt classification accuracy (ACC) to evaluate the performance of baselines and DinaNet 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, INA methods have strong competitiveness over most datasets. It can be attributed to the fact that INA methods additionally construct the interactions between pairwise nodes to capture the relationship information in the graph.
2. Compared to other INA methods, DinaNet still has superior performance on most datasets. This superiority can be attributed to the fact that DinaNet 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, Texas datasets as examples, the performance of DinaNet 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		INA		
		GCN	MOGCN	PAGCN	GAT	SGAT	CensNet	GraphAIR	DinaNet
Cora	14	43.8	**62.8**	47.6	44.5	45.7	57.7	54.1	62.3
	28	62.3	71.5	72.1	66.8	67.1	67.1	73.3	**74.5**
	140	81.7	82.4	83.6	83.1	81.9	79.1	84.2	**84.8**
ACM	15	51.8	68.1	69.7	56.3	56.2	64.2	68.1	**82.4**
	30	65.6	87.5	87.2	64.6	71.2	75.9	87.5	**88.5**
	60	87.8	90.1	90.9	87.4	90.2	89.7	90.7	**91.8**
Citeseer	15	24.7	58.9	60.4	33.1	28.1	57.6	55.3	**62.5**
	30	43.6	62.8	64.7	58.4	49.3	62.5	63.5	**65.2**
	160	70.4	72.4	70.4	72.6	70.6	68.1	**73.5**	72.5
Pubmed	15	43.9	63.2	63.5	42.6	40.2	61.4	59.6	**71.9**
	30	60.5	68.5	73.5	56.6	62.4	65.7	68.3	**77.9**
	60	79.0	79.2	79.3	79.1	80.2	69.9	80.1	**81.4**
Texas	9	42.2	46.2	53.9	52.3	54.1	55.9	53.2	**57.2**
	12	52.7	51.5	55.6	54.8	58.3	60.9	60.3	**61.6**
	18	54.6	57.6	65.3	57.6	62.6	64.3	**67.7**	67.5
Chameleon	16	23.6	25.5	27.7	27.1	26.3	27.4	24.6	**28.2**
	28	31.4	30.2	31.9	28.2	29.2	**32.5**	31.8	31.5
	120	47.6	46.9	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, i.e., 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 DinaNet-VAE outperforms other methods on most datasets, which once again verifies the necessity of incorporating the interactions to neighborhood awareness.
2. It is clear from the Table 4 that the performance of DinaNet-VAE obtains obvious improvements, compared with AIR-GAE based on the node-level interactions. It can be attributed to the fact that DinaNet-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 DinaNet-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	DinaNet-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 DinaNet in this subsection.

The contribution of components. Essentially, the final node representation of DinaNet is obtained by integrating three components: Node Information (NI), Adaptive Node Interaction (ANI) and High-level Edge Interaction (HEI). To explore the contributions of components, DinaNet selects different components to construct the variants, such as Dina-A represents that ANI is removed from DinaNet and Dina-NA represents that NI, ANI are removed from DinaNet. The results of node classification are shown in Table 5, and we have the following observations.

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

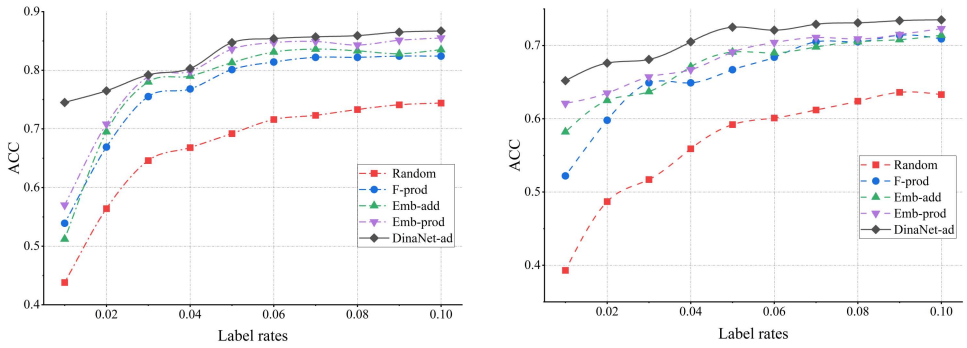
Table 5: ACC (%) of node classification with different combinations.

	NI	ANI	HEI	Cora	ACM	Citeseer	Pubmed
Dina-AH	✓			81.7	87.8	70.4	79.0
Dina-NH		✓		82.4	88.3	70.8	80.8
Dina-NA			✓	68.5	74.1	50.1	64.8
Dina-N		✓	✓	83.1	91.1	71.7	81.1
Dina-A	✓		✓	83.6	90.4	72.4	80.6
Dina-H	✓	✓		83.2	90.8	71.7	80.9
DinaNet	✓	✓	✓	**84.3**	**91.4**	**72.5**	**81.4**

The effectiveness of interaction construction methods. We select various construction methods of node interaction Cheng (2020); Zhao (2022) to evaluate the effectiveness of our approach. Since existing methods mainly consider the interaction at node level, we use adaptive node interaction 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 inner product between pairwise node embeddings;
5. **DinaNet-ad** is our construction method of adaptive node interaction that additionally considers the interactions of historical layer.

The comparison results are shown in Fig. 3, and we have the observations that DinaNet-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, DinaNet-ad performs exceptionally well even at extremely low label rates. This is because DinaNet-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 3: The performance comparisons of interaction construction methods in node classification on Cora, Citeseer.

5.4. Parameter sensitivity

In this subsection, we attempt to investigate how the hyperparameters of loss function (i.e., Eq. 11) 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 Fig. 4.

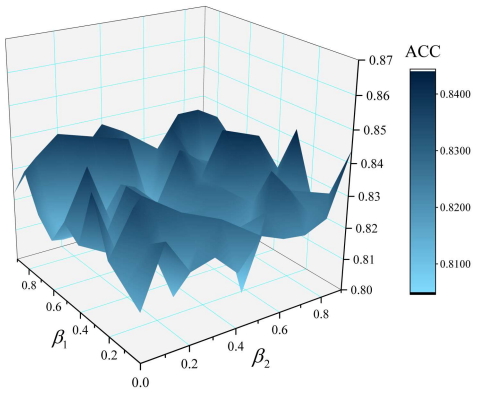


Figure 4: ACC of node classification on Cora with varying hyperparameters.

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. In addition, 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.

Furthermore, we use the random forest regression model to evaluate the importance of two hyperparameters, as shown in Fig. 5. Our results reveal that the importance of adaptive node interaction parameter β_1 is slightly greater than high-level edge interaction parameter β_2.

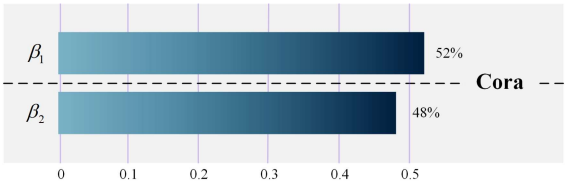
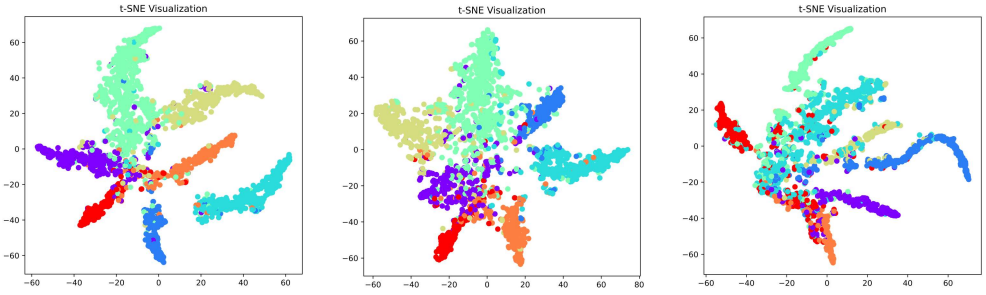


Figure 5: Parameter importance assessment.

5.5. *Visualization*

As shown in Figs. 6-7, 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 Fig. 6. 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 Fig. 7.

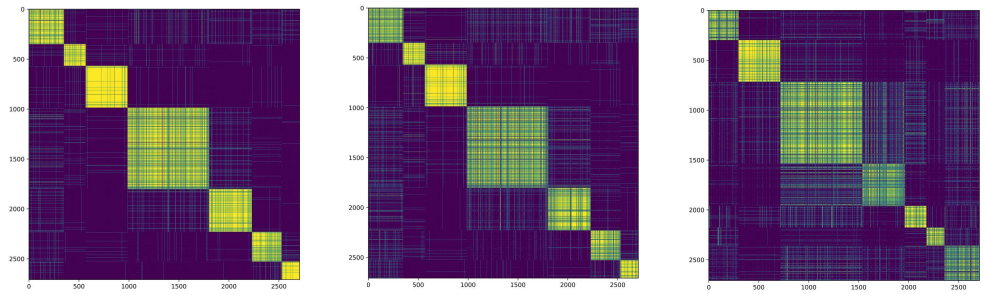


(a) DinaNet (b) GraphAIR (c) CensNet

Figure 6: Visualization comparison of node representation distribution on Cora.

It is clear from Fig. 6(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, DinaNet exhibits the most optimal embedding, the highest intra-class similarity, and the clearest distinct boundaries among different classes.

As shown in Fig. 7, we further analyze the quality of node representation. We can observe that the similarity matrix of DinaNet has more complete diagonal matrix blocks, indicating higher similarity intra-class and more accurate node representations.



(a) DinaNet (b) GraphAIR (c) CensNet

Figure 7: 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 Jack Jr (2008), and its details are shown in Table 6. 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 6: 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 Sanjay (2016), GCN Kipf (2017), DGM Kazi (2022) and DDGFCN Li (2023). Among them, DDGFCN, 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 7.

Table 7: ACC (%) and AUC (%) of patient classification.

		Ada-boost	GCN	DGM	DDGFCN	**DinaNet**
AD vs NC	ACC	90.0	92.1	98.7	99.3	**99.5**
	AUC	86.3	95.8	99.1	99.8	**99.9**
AD vs LMCI	ACC	90.0	88.5	92.5	94.6	**96.6**
	AUC	84.8	89.4	95.3	98.0	**99.2**
LMCI vs NC	ACC	86.7	90.5	95.3	98.0	**98.7**
	AUC	82.8	91.2	98.0	99.1	**99.5**

As shown in Table 7, the results indicate that DinaNet 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 DinaNet is 2% higher than the second-ranked method, which further confirms the effectiveness of DinaNet. Therefore, our method can be applied to the patient classification of Alzheimer's disease, demonstrating its flexible capability in real-life application.

7. Conclusion and future work

In this study, we introduce Dual-Interaction Neighborhood Awareness Network (DinaNet) — a novel GNN framework that addresses the limitations of existing INA methods in capturing relationship information of different levels. DinaNet 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 DinaNet compared to state-of-the-art methods. The application of Alzheimer's disease confirms its effectiveness in real-life scenario. In conclusion, DinaNet 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 INA methods are effective and how they are related. In

future work, we will resort to Taylor interaction effects for explaining the existing INA methods, which attempts to unify INA methods into a weighted allocation of two typical effects: independent effects and interaction effects.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

The research is supported by National Natural Science Foundation of China (62076111), Jiangsu Province Graduate Research and Practice Innovation Program Project (KYCX_233883).

References

Khoshraftar, S., & An, A. (2024). A survey on graph representation learning methods. ACM Transactions on Intelligent Systems and Technology, 15(1), 1-55. https://doi.org/10.1145/3633518.

Kipf, T.N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In Proc. of the 5th International Conference on Learning Representations. 24-26. https://doi.org/10.48550/arXiv.1609.02907.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2018). Graph attention networks. In Proc. of the 6th International Conference on Learning Representations. https://doi.org/10.48550/arXiv.1710.10903.

Zeng, H., Zhou, H., Srivastava, A., Kannan, R., & Prasanna, V. (2019). GraphSAINT: Graph sampling based inductive learning method. In Proc. of the 7th International Conference on Learning Representations. https://doi.org/10.48550/arXiv.1907.04931.

Ye, Y., & Ji, S. (2021). Sparse graph attention networks. IEEE Transactions on Knowledge and Data Engineering, 35(1), 905-916. https://doi.org/10.1109/TKDE.2021.3072345.

Guo, Q., Yang, X., Zhang, F., & Xu, T. (2024). Perturbation-augmented graph convolutional networks: A graph contrastive learning architecture for effective node classification tasks. Engineering Applications of Artificial Intelligence, 129, 107616. https://doi.org/10.1016/j.engappai.2023.107616.

Wang, J., Liang, J., Cui, J., & Liang, J. (2021). Semi-supervised learning with mixed-order graph convolutional networks. Information Sciences, 573, 171-181. https://doi.org/10.1016/j.ins.2021.05.057.

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

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., & Weinberger, K. (2019). Simplifying graph convolutional networks. In Proc. of the 36th International Conference on Machine Learning. 6861-6871. https://doi.org/10.1145/3292500.3330851.

Wang, X., Yang, X., Wang, P., Yu, H., & Xu, T. (2024). SSGCN: A sampling sequential guided graph convolutional network. International Journal of Machine Learning and Cybernetics, 15(5), 2023-2038. https://doi.org/10.1007/s13042-023-02013-2.

Hu, F., Zhu, Y., Wu, S., Huang, W., Wang, L., & Tan, T. (2021). Graphair: Graph representation learning with neighborhood aggregation and interaction. Pattern Recognition, 112, 107745. https://doi.org/10.1016/j.patcog.2020.107745.

Jiang, X., Zhu, R., Li, S., & Ji, P. (2020). Co-embedding of nodes and edges with graph neural networks. IEEE Transactions on Pattern Analysis and Machine Intelligence. 45(6), 7075-7086. https://doi.org/10.1109/TPAMI.2020.3029762.

Guang, M., Yan, C., Xu, Y., Wang, J., & Jiang, C. (2024). Graph convolutional networks with adaptive neighborhood awareness. IEEE Transactions on Pattern Analysis and Machine Intelligence. https://doi.org/10.1109/TPAMI.2024.3391356.

Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Representation learning on graphs: Methods and applications. IEEE Data Engineering Bulletin. 40(3), 52-74. https://doi.org/10.48550/arXiv.1709.05584.

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

Cai, L., Li, J., Wang, J., & Ji, S. (2021). Line graph neural networks for link prediction. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(9), 5103-5113. https://doi.org/10.1109/TPAMI.2021.3080635.

Wang, X., Zhu, M., Bo, D., Cui, P., Shi, C., & Pei, J. (2020). AM-GCN: Adaptive multi-channel graph convolutional networks. In Proc. of the 26th ACM International Conference on Knowledge Discovery and Data Mining. 1243-1253. https://doi.org/10.1145/3394486.3403177.

Jack Jr, C. R., Bernstein, M. A., Fox, N. C., Thompson, P., Alexander, G., Harvey, D., ... & Weiner, M. W. (2008). The Alzheimer's disease neuroimaging initiative (ADNI): MRI methods. Journal of Magnetic Resonance Imaging: An Official Journal of the International Society for Magnetic Resonance in Medicine, 27(4), 685-691. https://doi.org/10.1002/jmri.21049.

Dwivedi, V. P., Joshi, C. K., Luu, A. T., Laurent, T., Bengio, Y., & Bresson, X. (2023). Benchmarking graph neural networks. Journal of Machine Learning Research, 24(43), 1-48. http://jmlr.org/papers/v24/22-0567.html.

Peng, L., Hu, R., Kong, F., Gan, J., Mo, Y., Shi, X., & Zhu, X. (2024). Reverse Graph Learning for Graph Neural Network. IEEE Transactions on Neural Networks and Learning Systems, 35(4), 4530-4541. https://doi.org/10.1109/TNNLS.2022.3161030.

Phan, H. T., Nguyen, N. T., & Hwang, D. (2023). Fake news detection: A survey of graph neural network methods. Applied Soft Computing, 139, 110235. https://doi.org/10.1016/j.asoc.2023.110235.

Huang, C., Li, M., Cao, F., Fujita, H., Li, Z., & Wu, X. (2022). Are graph convolutional networks with random weights feasible?. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(3), 2751-2768. https://doi.org/10.1109/TPAMI.2022.3183143.

Yang, Y., Sun, Y., Ju, F., Wang, S., Gao, J., & Yin, B. (2023). Multi-graph fusion graph convolutional networks with pseudo-label supervision. Neural Networks, 158, 305-317. https://doi.org/10.1016/j.neunet.2022.11.027.

Kazi, A., Cosmo, L., Ahmadi, S. A., Navab, N., & Bronstein, M. M. (2022). Differentiable graph module (dgm) for graph convolutional networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(2), 1606-1617. https://doi.org/10.1109/TPAMI.2022.3170249.

Li, F., Wang, Z., Guo, Y., Liu, C., Zhu, Y., Zhou, Y., ... & Wang, H. (2023). Dynamic dual-graph fusion convolutional network for alzheimer's disease diagnosis. In Proc. of the 30th International Conference on Learning Representations. 675-679. https://doi.org/10.1109/ICIP49359.2023.10222732.

Teng, Q., Yang, X., Sun, Q., Wang, P., Wang, X., & Xu, T. (2024). Sequential attention layer-wise fusion network for multi-view classification. International Journal of Machine Learning and Cybernetics, 1-13. https://doi.org/10.1007/s13042-024-02260-x.

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

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

Lian, J., Zhou, X., Zhang, F., Chen, Z., & Sun, G. (2018). Xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In Proc. of the 24th ACM International Conference on Knowledge Discovery and Data Mining. 1754-1763. https://doi.org/10.1145/3219819.3220023.

Gao, J., Gao, J., Ying, X., Lu, M., & Wang, J. (2021). Higher-order interaction goes neural: A substructure assembling graph attention network for graph classification. IEEE Transactions on Knowledge and Data Engineering, 35(2), 1594-1608. https://doi.org/10.1109/TKDE.2021.3105544.

Zhao, Z., Yang, Z., Li, C., Zeng, Q., Guan, W., & Zhou, M. (2022). Dual feature interaction-based graph convolutional network. IEEE Transactions on Knowledge and Data Engineering, 35(9), 9019-9030. https://doi.org/10.1109/TKDE.2022.3220789.

Yan, S., Li, C., Wang, H., Lin, B., & Yuan, Y. (2024). Feature interactive graph neural network for KG-based recommendation. Expert Systems with Applications, 237, 121411. https://doi.org/10.1016/j.eswa.2023.121411.

Zhu, S., Zhou, C., Pan, S., Zhu, X., & Wang, B. (2019). Relation structure-aware heterogeneous graph neural network. In Proc. of the 23th ACM International Conference on Data Mining. 1534-1539. https://doi.org/10.1109/ICDM.2019.00203.

Rendle, S. (2010). Factorization machines. In Proc. of the 10th ACM International Conference on Data Mining, 995-1000. https://doi.org/10.1109/ICDM.2010.127.

Guo, H., Tang, R., Ye, Y., Li, Z., & He, X. (2017). DeepFM: a factorization-machine based neural network for CTR prediction. In Proc. of the 26th International Joint Conference on Artificial Intelligence, 1725-1731. https://doi.org/10.24963/IJCAI.2017/239

Wang, R., Fu, B., Fu, G., & Wang, M. (2017). Deep & cross network for ad click predictions. In Proc. of the 11th ACM International Conference on Knowledge Discovery and Data Mining, 1-7. https://doi.org/10.1145/3124749.3124754.

Song, W., Shi, C., Chence Shi, Zhijian Duan, Yewen Xu, Ming Zhang, & Jian Tang. (2019). Autoint: Automatic fea-

ture interaction learning via self-attentive neural networks. In Proc. of the 28th ACM International Conference on Information and Knowledge Management, 1161-1170. https://doi.org/10.1145/3357384.335792.

Xie, Z., Zhang, W., Sheng, B., Li, P., & Chen, C. P. (2021). BaGFN: Broad attentive graph fusion network for high-order feature interactions. IEEE Transactions on Neural Networks and Learning Systems, 34(8), 4499-4513. https://doi.org/10.1109/TNNLS.2021.3116209.

Ahn, S. J., & Kim, M. (2021). Variational graph normalized autoencoders. In Proc. of the 30th ACM International Conference on Information and Knowledge Management, 2827-2831. https://doi.org/10.1145/3459637.3482215.

Kipf, T. N., & Welling, M. (2016). Variational graph auto-encoders. NeurIPS Workshop Bayesian Deep Learn. https://doi.org/10.48550/arXiv.1611.07308.

Cheng, W., Shen, Y., & Huang, L. (2020). Adaptive factorization network: Learning adaptive-order feature interactions. In Proc. of the AAAI Conference on Artificial Intelligence, 34(4), 3609-3616. https://doi.org/10.1609/aaai.v34i04.5768.

Bojchevski, A., & Günnemann, S. (2017). Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In Proc. of 6th International Conference on Learning Representations. https://doi.org/10.48550/arXiv.1707.03815.

Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., & Yu, P. S. (2019). Heterogeneous graph attention network. In Proc. of the 28th World Wide Web Conference, 13-17. https://doi.org/10.1145/3308558.3313562.

Pei, H., Wei, B., Chang, K. C. C., Lei, Y., & Yang, B. (2020). Geom-GCN: Geometric graph convolutional networks. In Proc. of 8th International Conference on Learning Representations, 26-30. https://doi.org/10.48550/arXiv.2002.05287.

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

Sanjay, V., & Swarnalatha, P. (2022). An enhanced approach for detecting Alzheimer's disease. In Proc. of the 2022 Smart Technologies, Communication and Robotics, 1-5. https://doi.org/10.1109/STCR55312.2022.10009274.