OXFORD

The Computer Journal, 2025, 1–8
https://doi.org/10.1093/comjnl/bxaf086
Original Article

A dual graph embedding based node classification method

Yi Liu‡, Yefang Chen‡, Yu Xin*

Faculty of Electrical Engineering and Computer Science, Ningbo University, 818 Fenghua Road, Jiangbei District, 315211 Ningbo, China

*Corresponding author: Faculty of Electrical Engineering and Computer Science, Ningbo University, 818 Fenghua Road, Jiangbei District, 315211 Ningbo, China. E-mail: xinyu@nbu.edu.cn

‡Authors Yi Liu and Yefang Chen contributed equally to this work.

Abstract

Graph node classification is an important field in network representation learning algorithms. With the development of deep learning, the focus has shifted to models like graph convolutional network (GCN) and graph attention network (GAT). GCN and GAT only consider direct adjacent nodes in node embeddings, so the receptive field of node embeddings is constricted which cannot fully express the network structure. To expand the receptive fields of node embeddings and strengthen the effect of graph structure on node embedding, the dual graph embedding (DGE) framework is proposed. In this framework, an attention mechanism is adopted to transform the original graph into a weighted graph, and a dual graph transform strategy is proposed to convert the weighted graph into a dual graph. The edges in the original graph indicate the nodes in the dual graph. Performing node embedding methods such as GAT/GCN in the dual graph is performing edge embeddings in the original graph. Node embeddings in the original graph can be realized by aggregating the edge embeddings for each node. According to the DGE framework and multi-head model, we designed DGE-GCN, DGE-GCN-GAT, DGE-GCN-MULTI, and DGE-GAT-MULTI methods. Experimental results show that DGE methods outperform various node classification methods, especially in graphs with high link density.

1. INTRODUCTION

Artificial intelligence (AI) has played a very important role in many fields such as medical treatment, transportation, biology, industrial production, and so on, which has brought a lot of convenience to human life and work. With the Internet, social media, mobile devices, and sensors extensively used, AI has been accumulating vast amounts of data from various industries. The effect of AI on various industries is continually improving. With the development of artificial intelligence technology, AI has made great progress in image, audio, natural language processing, and other aspects. AI becomes the development direction of information technology in the future. Among them, graph embedding is an important field of AI technology. The graph embedding technology can represent code objects as vectors, according to entity relationships. Fuzzy relationships between entities can be coded, and the fuzzy relationship modeling problems can be solved. Currently, more and more scholars have conducted in-depth studies on graph embedding, solved key problems such as protein prediction [1], recommendation system [2], traffic prediction [3, 4], and achieved much social application by graph embedding.

Traditional network representation learning utilizes machine learning to express entity features, according to the temporal relation, dynamic evolution, similar relation, and interactive relation of entities. For the graph embedding model, the machine learning classification and clustering strategy are adopted to optimize the model parameters to achieve the coding of entity node features. The model of traditional machine learning method is relatively simple, which leads to problems such as high deviation, low precision, and poor prediction effect. With the improvement of computing capacity and the increment of training data volume, the deep learning method has higher accuracy than traditional machine learning. In terms of deep learning, with the success of CNN (convolutional neural network) on image-based tasks, CNN began to be applied to graph embedding field, then generating graph convolution neural networks (GCNs) [5]. After that, a large number of graph embedding methods based on GCNs were proposed, such as GAT [6] and GraphSage [7]. Compared with traditional methods, GCNs show good performance in graph processing such as node classification, link prediction, and graph classification, but also have the advantage of high precision in multiple applications such as recommendation systems [8], traffic prediction [9], and behavior recognition [10].

Current studies on GCNs can be mainly divided into two directions: spectral-based models [5, 11, 12] and spatial-based models [6, 7, 13]. The spectral models emphasize the mathematical convolution operation of graph-based Fourier transform, while spatial models emphasize message propagation and aggregation among nodes. The drawback of spatial models is that, it only considers the direct adjacency of graphs, could not fully reflect the deep latent graph structure. Although some improvement methods attempt to expand the receptive field through stacking multiple layers, this approach faces issues of overfitting and high computational cost, especially when dealing with large-scale graphs, where efficiency and scalability become significant challenges.

To address the limitation of spatial models in capturing deep structural information of the graph in node classification tasks. Our contributions can be summarized as follows:

Received: July 07, 2024. **Revised:** March 24, 2025. **Accepted:** May 19, 2025

- To strengthen the impact of deep latent graph structure on graph embedding, we propose DGE framework to extend the receptive field on graph convolution.
- In the dual graph embedding (DGE) framework, we carry out GCNs on dual graph to extend the embedding receptive field, and transform the obtained node embedding of dual graph into original graph.
- Based on the DGE framework, we propose variant methods such as DGE-GCN, DGE-GCN-GAT, DGE-GCN-MULTI, and DGE-GAT-MULTI.
- Extensive experiments have shown that DGE effectively improves the accuracy of node classification and enhances the receptive filed on graph convolution.

2. RELATED WORK

Currently, graph embedding methods are mainly divided into machine learning and deep learning methods. The machine learning methods focus on entity-relationship modeling and emphasize the interpretability of the model. Early graph embedding methods based on machine learning mainly adopt linear models, such as LLE [14] and GraRep [15].

With the rapid development of NLP, inspired by the great success achieved by word2vec [16], a great deal of random-walk-based methods on graph embedding have emerged recently, such as Deepwalk [17], LINE [18], node2vec [19], and SDNE [20]. These methods feed node sequences generated by random walks into the Skip-Gram model for training, and the obtained node embedding can be represented in low-dimensional space. Due to the simplicity of graph modeling, less trainable parameters, and poor prediction accuracy of massive samples, the limitations of the above machine learning methods have become increasingly obvious. With the development of deep learning, deep learning technologies represented by graph convolutional network (GCN) and graph attention network (GAT) have been widely applied in the field of graph embedding and have better application effects.

2.1. Graph convolution networks

Graph convolutional neural network is used to extract the characteristics of topological structure from graph network and embed nodes. Currently, researches on graph convolutional neural network mainly focus on spectral-based methods and spatial-based methods.

Generally, graph convolution utilizes convolution operators to embed nodes. In terms of spectral-based methods, the convolution operators are defined from the frequency domain. Defferrard [11] first defined the convolution operation in the Fourier domain. However, due to its heavy computation cost, it has difficulty scaling large graphs. To improve efficiency, Defferrard [12] proposed ChebNet, which can approximate the K-polynomial filters by Chebyshev expansion on graph Laplacian. Kipf and Welling [5] obtained first-order GCN by truncating the Chebyshev polynomial to the first-order neighborhood and further simplifying the parameters, which greatly reduced the space-time complexity.

In terms of spatial methods, an aggregation function is defined to aggregate each central node and its neighbors. MoNET [13] adopts different methods to assign weights to neighbor nodes. It adopts node pseudo coordinates to determine the relative position between the node and its neighbors but also maps the relative position to the weight between two nodes. GraphSage [7] utilizes 2 stage aggregation functions such as joins, maximum pools, and Long Short-Term Memory (LSTM) aggregators, after sampling a fixed number of neighbors, to embed nodes. HGCN [21] proposed a deep GCN to carry out the semi-supervised node classification task, adding the graph pooling mechanism into the GCN to expand the receptive field.

2.2. Graph attention networks

In image processing, increasing more attention to target areas, we can get more detailed information about the target areas. By which, the training accuracy of DNN can be improved. Inspired by this principle, Velickovic [15] introduced an attention mechanism into graph learning and proposed a GAT. Unlike GCN, which uses a fixed or learnable Laplace polynomial to aggregate node information, GAT uses attention mechanisms on graphs to aggregate node information. The essential differences between GAT and GCN are as follows: in GCN, the weights of aggregated neighbor nodes are obtained by the graph topological structure, and GCN trains parameters globally and indiscriminately, regardless of local structural differences. In GAT, the attention weight depends on the relation between adjacent nodes, and GAT emphasizes the independence of the local graph structure and avoids the influence of global indiscrimination on node embedding. The experimental results of graph node classification show that the adaptive capability of GAT enables it to integrate information from node features and graph topology more effectively, but also can capture the differences of node embedding better.

Currently, scholars have proposed a large number of graph embedding methods based on GAT. Park [22] applied the attention mechanism on heterogeneous networks with multiple attributes, integrating node embeddings in multiple graphs. ASAPool [23] applied the attention mechanism on graph classification, to calculate the attention weight by the features of nodes and supernodes so that it could better capture the global structure. By which the important nodes can be selected, perform graph pooling operations. AM-GCN [24] extracts specific and common embeddings from node features and topological structures and learns the adaptive importance of node embeddings using the attention mechanism. The gated attention network (GAAN) proposed by Zhang [3] also adopts the multi-head attention mechanism to update the hidden state of nodes. Unlike GAT, this model gives different weights to each head, so that the important parts can obtain more computational resources. In addition, GAM model [25] uses a cyclic neural network model and attention mechanism to solve the graph classification problem. Edge attention network EGAT [26] improved the attention weight calculation method of GAT and took edge features into attention weight calculation, so that edge information can also be fully utilized. DPGCN [27] constructs dual graphs and calculates attention on the dual graphs, to expand the receptive fields of graph convolution.

2.3. Dual graph

Given a graph $G = \{V, E, A\}$, its dual graph [27] is $G^D = \{V^D, E^D, A^D\}$. The nodes in G^D represent the edges in G. Therefore, the edges in G can be mapped to the nodes in G^D, namely the $V^D = \left\{v_{ij}^D \mid e_{i,j} \in E\right\}$. For instance, In G, e_{ij} and e_{jq} have a common vertex v_j, thus the mapped nodes v_{ij}, v_{pq} in G^D are linked. Figure 1 shows how to convert from the original G to dual graph G^D.

As shown in Fig. 1, the edges$\{e_{12}, e_{13}, e_{03}, e_{02}, e_{34}, e_{24}\}$ in G are transformed into the nodes$\left\{v_{12}^D, v_{13}^D, v_{03}^D, v_{02}^D, v_{34}^D, v_{24}^D\right\}$ in G^D after the dual graph transformation.

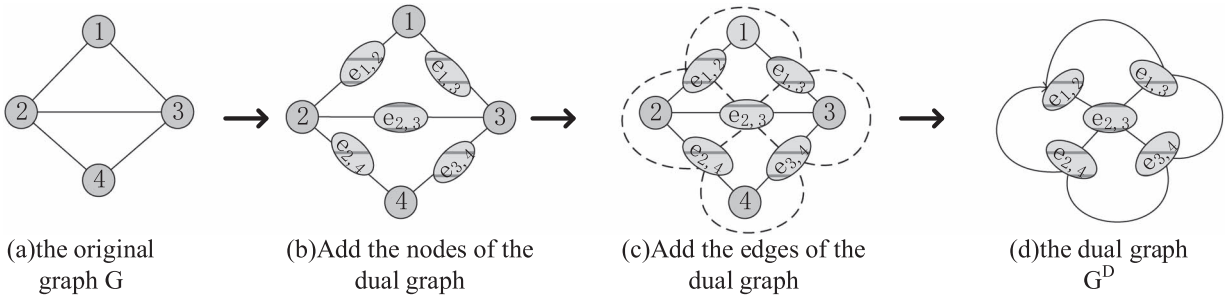


Figure 1. Dual graph transformation.

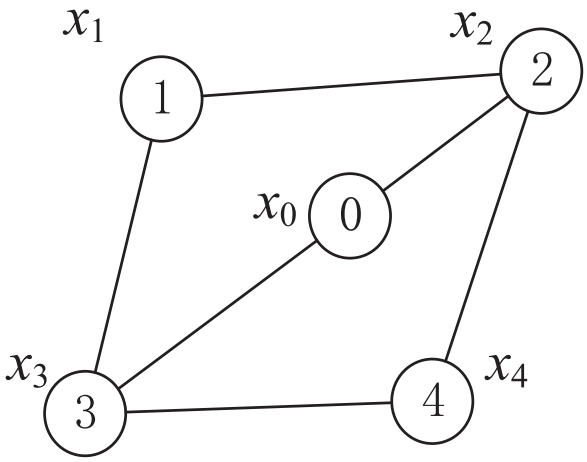


Figure 2. An example of input graph, where its $V = \{v_0, v_1, v_2, v_3, v_4\}$.

3. DUAL GRAPH EMBEDDING

Symbol definition. An undirected graph can be represented as $G = \{V, E, A\}$, where V and E represent the node set and the edge set. $|V| = n$ is the number of nodes and $|E| = m$ is the number of edges in G. A is the adjacency matrix. We take Fig. 2 as an example to illustrate the DGE method proposed in this paper. In Fig. 2, G is composed of five nodes $V = \{v_0, v_1, v_2, v_3, v_4\}$,and $X = \{x_0, x_1, x_2, x_3, x_4\}$ represents the features of v_{0-4}.

3.1. Graph link embedding based on attention mechanism

We use the attention mechanism to transform G into a weighted graph. The $\alpha_{i,j}$ is obtained by $[h_i, h_j]$ (h_i, the embedding vector of v_i), and $\alpha_{j,i}$ is obtained by $[h_j, h_i]$, so the attention weights obtained by attention mechanism are directed. The attention coefficient of nodes v_j to v_i is the following:

$$\alpha_{i,j} = \frac{exp\left(LeakyReLU(a^T[Wh_i \| Wh_j])\right)}{\sum_{v_k \in \tilde{N}(v_i)} exp\left(LeakyReLU(a^T[Wh_i \| Wh_K])\right)} \tag{1}$$

where $\alpha_{i,j}$ means the importance of v_j on v_i, and $\alpha_{i,j} \neq \alpha_{j,i}$, $\sum_{v_k \in \tilde{N}(v_i)} \alpha_{i,k} = 1$. W is a global parameter matrix, and $\|$ represents concatenation. $H = \{h_0, h_1, \ldots\}$ represents the embedded vector of node V, and initial $H^{(0)}$ is the following:

$$H^{(0)} = GNN(A, X) \tag{2}$$

Because $\alpha_{i,j}$, $\alpha_{j,i}$ represent the weight of edge $e_{i,j}$ and $e_{j,i}$, the attention can express the importance between nodes in G. The embedding $h_{i,j}$ of edge $e_{i,j}$ can be obtained by h_i, h_j, $\alpha_{i,j}$ and $\alpha_{j,i}$, and the transformation expression is shown in Equation 3.

$$h_{i,j} = \alpha_{i,j} h_i + \alpha_{j,i} h_j \tag{3}$$

In Fig. 3b, $(\alpha_{1,3}, \alpha_{1,2})$ represent the attention weight from nodes (v_3, v_2) to v_1, and $\alpha_{3,1}$, $\alpha_{3,4}$ represent the attention weight of nodes (v_0, v_1, v_4) to v_3, respectively. The calculation process of vectors $h_{1,3}$ of edges $e_{1,3}$ is shown in Fig. 3c according to Equation 3. The transformation from unweighted graph to weighted graph can be realized by Equation 1, and the edges can obtained embeddings from node embeddings by Equation 3. The proposed graph link embedding (GLE) is based on Equations 1 and 3, by which the edge embeddings of G can be obtained according to the node embeddings.

3.2. Dual graph transform

The dual graph transformation (DGT) takes the edges in the original graph as the nodes in the dual graph, and the adjacent edges in the original graph are the adjacent nodes in the dual graph. If G^D is a dual graph of G, e_{ij} in G is equal to v_{ij}^D in G^D, and the edge embedding $h_{i,j}$ in G, is equal to the node embedding h_{ij}^D in G^D.

$$h_{ij}^D = h_{i,j} \tag{4}$$

The adjacency matrix of G^D can be obtained from G according to Equation 5:

$$A^D = B^T B - 2I^2 \tag{5}$$

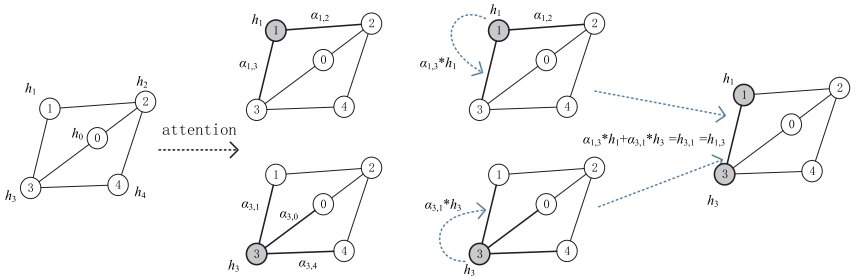
where $A^D \in R^{m \times m}$ is the adjacency matrix of a dual graph, and $I \in R^{m \times m}$ is the identity matrix. $B \in R^{n \times m}$ is the association matrix of G, which describes the relationship between nodes and edges, as defined below:

$$B_{ij} = \begin{cases} 1 & \text{if } v_i \text{ and } e_j \text{ connected} \\ 0 & \text{else} \end{cases} \tag{6}$$

For the graph G shown in Fig. 2, its adjacency matrix and association matrix are as follows:

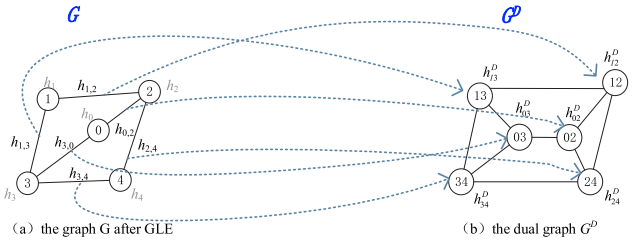
$$A = \begin{array}{c} \\ v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{ccccc} v_0 & v_1 & v_2 & v_3 & v_4 \end{array} \\ \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array} \quad B = \begin{array}{c} \\ v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \end{array} \begin{array}{c} \begin{array}{cccccc} e_{02} & e_{03} & e_{12} & e_{13} & e_{24} & e_{34} \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

where $B_{00} = 1$ means v_0 is connected to e_{02} while $B_{02} = 0$ means v_0 is not connected to e_{12}. According to Equation 5, the calculation



(a) the original embedded graph (b) the attention coefficient of v_1 and v_3 (c) the weighed embedded graph

Figure 3. The process of GLE.



(a) the graph G after GLE (b) the dual graph G^D

Figure 4. Dual embedding transform.

process of the adjacency matrix A^D of G^D can be obtained as follows:

$$A^D = B^T B - 2I^D = \begin{array}{c} \begin{matrix} e_{02} & e_{03} & e_{12} & e_{13} & e_{24} & e_{34} \end{matrix} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \end{pmatrix} \end{array} \begin{matrix} e_{02} \\ e_{03} \\ e_{12} \\ e_{13} \\ e_{24} \\ e_{34} \end{matrix}$$

The dual embedding transformation process of G is shown in Fig. 4, where Fig. 4a is the embedded G by GLE, and Fig. 4b represents the topological structure and node embedding vector of G^D after DGT transformation. According to DGT, the edges $\{e_{12}, e_{13}, e_{03}, e_{02}, e_{34}, e_{24}\}$ of G are equal to the nodes $\{v_{12}^D, v_{13}^D, v_{03}^D, v_{02}^D, v_{34}^D, v_{24}^D\}$ in G^D, The edge embedding vectors $\{h_{1,2}, h_{1,3}, h_{3,4}, h_{2,4}, h_{0,3}, h_{0,2}\}$ in Fig. 4a are the node embedding vectors $\{h_{12}^D, h_{13}^D, h_{03}^D, h_{3,4}, h_{02}^D, h_{34}^D, h_{24}^D\}$ in Fig. 4b.

3.3. Dual graph link embedding

Dual graph G^D obtained by DGT transformation can be seen as another form of G. The updating process of node embedding on G^D, can be seen as edge embedding on G. The edge embedding process have more extensive receptive fields than node embedding, so node embedding on G^D will embed topological structure of G better. In order to emphasize the influence of topology, we can use GNN method to update the node embeddings of G^D. In this paper, GAT is selected to update node embedding. The attention of GAT is directed, and the expression of attention for dual graph transformation is shown in Equation 7 according to DGT rule:

$$\alpha_{ij,jk} = \frac{exp\left(LeakyReLU(a^T[Wh_{ij} \| Wh_{jk}])\right)}{\sum_{v_{gk}^D \in \tilde{N}\left(v_{ij}^D\right)} exp\left(LeakyReLU(a^T[Wh_{ij} \| Wh_{gk}])\right)} \tag{7}$$

where $\alpha_{ij,jk}$ is the attention of directed edge $e_{ij,jk}$ in G^D, W is a trainable parameter. $\alpha_{ij,jk}$ can indicate the importance of node v_{jk}^D to node v_{ij}^D, and $\alpha_{ij,jk} \neq \alpha_{jk,ij}$, $\sum_{v_{gk}^D \in \tilde{N}(v_{ij}^D)} \alpha_{ij,gk} = 1$. The update expression for the node embeddings on dual graph G^D is the following:

$$h'_{ij} = \sigma\left(\sum_{v_{pq} \in \mathbf{N}(v_{ij})} \alpha_{ij,pq} W h_{pq}\right) \tag{8}$$

The expression of node embedding based on GCN is the following:

$$X^{(l+1)} = \sigma\left(\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{\frac{1}{2}} X^{(l)} W^{(l)}\right) \tag{9}$$

where $\hat{A} = A + I_n$, and $\hat{D}_{ii} = \sum_j \hat{A}_{i,j}$. $W^{(l)}$ is the parameter matrix of l-layer neural network.

The node embeddings in the dual graph G^D are equal to the edges embeddings in graph G, namely $h_{i,j} = h_{ij}^D$. The node embedding on G can be obtained by Equation 10.

$$h'_i = \sum_{v_j \in \mathbf{N}(v_i)} h_{i,j} \tag{10}$$

4. EXPERIMENTS

4.1. Datasets

To verify the performance of the model proposed in this paper, we carry out the proposed model for node classification tasks on three public graph datasets. These datasets belong to the citation network, where documents and citations are treated as nodes and edges, respectively. The statistics of these datasets are summarized in Table 1.

- Cora: This dataset consists of 2708 scientific publications and 5429 links. Each publication is described by a 1433-dimensional word vector as a node feature. And each node is labeled with the class it belongs to.
- CiteSeer: A citation network consists of 3312 scientific publications divided into six classes, and the dataset contains 4660 edges. Node attributes are bag-of-words representations of the papers and all nodes are divided into six areas.
- PubMed: A citation network consists of 19 717 scientific publications from the PubMed database about diabetes classified into one of three classes. Each node is described by a TF/IDF weighted word vector from a dictionary of 500 unique words.

4.2. Experimental setup

The experimental platform is Win10 operating system, and the hardware environment is Intel i7 10700 CPU 3.80GHz, GeForce RTX™ 2080 Ti, and 32GB RAM. The source code of DGE is based on

Table 1. Main parameters of experimental datasets

Dataset	Category	Nodes	Edges	Classes	Features	Label proportion
Cora	Citation network	2708	5429	7	1433	0.052
CiteSeer	Citation network	3327	4732	6	3703	0.036
PubMed	Citation network	19 717	44 338	3	500	0.003

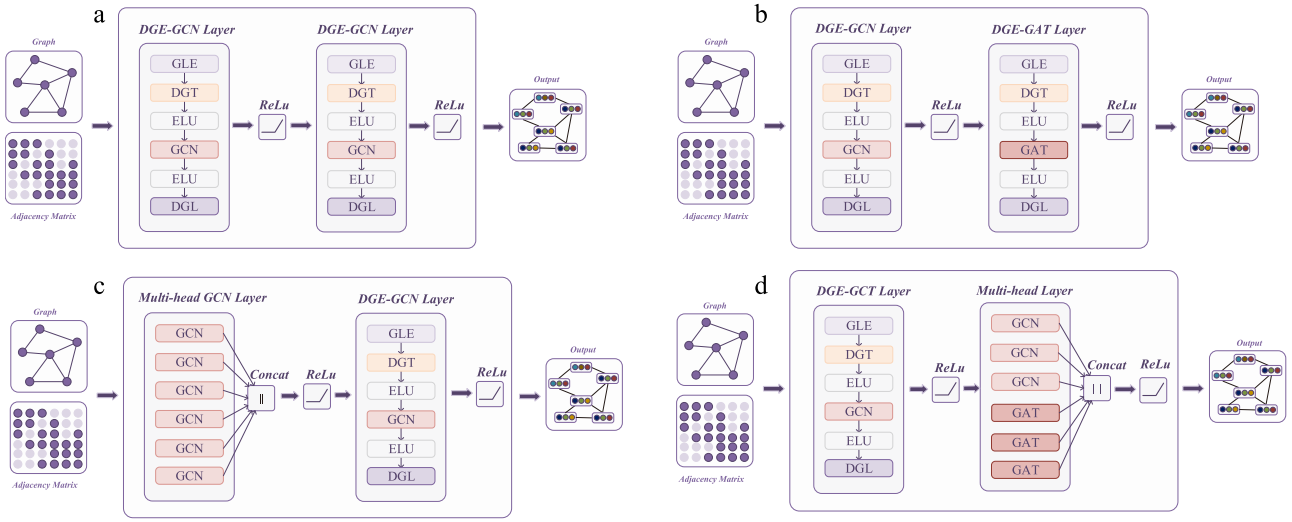


Figure 5. Networks framework based on DGE.

PyTorch and dual graph link (DGL) learning framework. Through a large number of experiments, we summarize four effective network frameworks based on DGE, and their structures are shown in Fig. 5.

DGE-GCN: The structure is shown in Fig. 5a. This model takes GCN as the dual graph embedding method and two-layer DGE-GCN as the backbone network. This model stacks the DAT layer and GCN layer in series, and the output of each layer uses the ELU activation function. The final output uses the softmax activation function to classify nodes.

DGE-GCN-GAT: The structure is shown in Fig. 5b. This structure contains two embedding layers. Firstly, GCN is used to embed nodes on the dual graph, to increase the embedding similarity between neighbor nodes. Secondly, DGE-GAT was used to enhance the weight of nodes with a high degree, to increase the specificity of the graph. After two embedding layers, the softmax activation function was used to classify nodes.

DGE-GCN-Multi: The structure shown in Fig. 5c, we design a two-layer model. To reduce parameters, the DGE network is only used in the second layer. The first layer consists of GCN attention headers ($K = 8$) with ELU activation functions. Each of the attention headers has 16-dims features. The second layer uses DGE-GCN to obtain node embedding and classifies nodes by the softmax activation function.

DGE-GAT-Multi: The structure shown in Fig. 5d, we also adopt a two-layer model. In the first layer, based on the multi-heads mechanism, GCN and GAT are mixed in parallel with $k = 8$ (4 GCNs, 4 GATs), and each attention head has 16-dims features. The second layer uses DGE-GAT to obtain node embedding and classifies nodes by softmax activation function.

All these models use cross-entropy as a loss function. The optimizer is Adam with an initial learning rate of 0.005, and the training epoch is 1000. Accuracy is used to evaluate model efficiency. Other hyperparameters are set using the values reported in GAT.

4.3. Contrast on DGE-based models

In this experiment, the four proposed DGE-based models, DGE-GCN, DGE-GCN-GAT, DGE-GCN-Multi, and DGE-GAT-Multi, were compared in the dataset Cora and CiteSeer [15]. The experiments are designed as follows:

1) **Training efficiency comparison.** The priori parameters of DGE based models are the number of heads h and the embeddings dimension n of hidden layer. We took the number of heads $h = [2, 4, 8, 16]$ and embeddings dimension $n = [4, 8, 16, 20]$ as input priori parameters for the four DGE based models, to conduct training in datasets Cora and CiteSeer. The changes of Accuracy-Epoch were shown in Fig. 6. When the priori parameters h and n increase, the amount of model parameters increase correspondingly, leading to the slow convergence rate of accuracy. That makes the training time longer. In addition, when h and n became larger ($h > 8$, $n > 16$), accuracy fluctuates sharply, and the training process has an over-fitting tendency. By this experiment, the effective parameters interval of DGE-based modes is $h \in [2, 8]$ and $n \in [4, 16]$.
2) **Accuracy comparison.** We used Cora and CiteSeer datasets to compare the accuracy of DGE-GCN, DGE-GCN-GAT, DGE-GCN-Multi, and DGE-GAT-Multi models to analyze the effect of each model. Sine DGE- GCN-Multi and DGE-GAT-Multi were multi-head models, we took $h = [2, 4, 8, 16]$ as the hyperparameter of multi-heads. Figure 7 shows the accuracy comparison of the four models. Except for ($h = 2$, $n = 4$) and ($h = 2$, $n = 8$), DGE-GCN-Multi has better performance than the other models. By this experiment, it can be seen

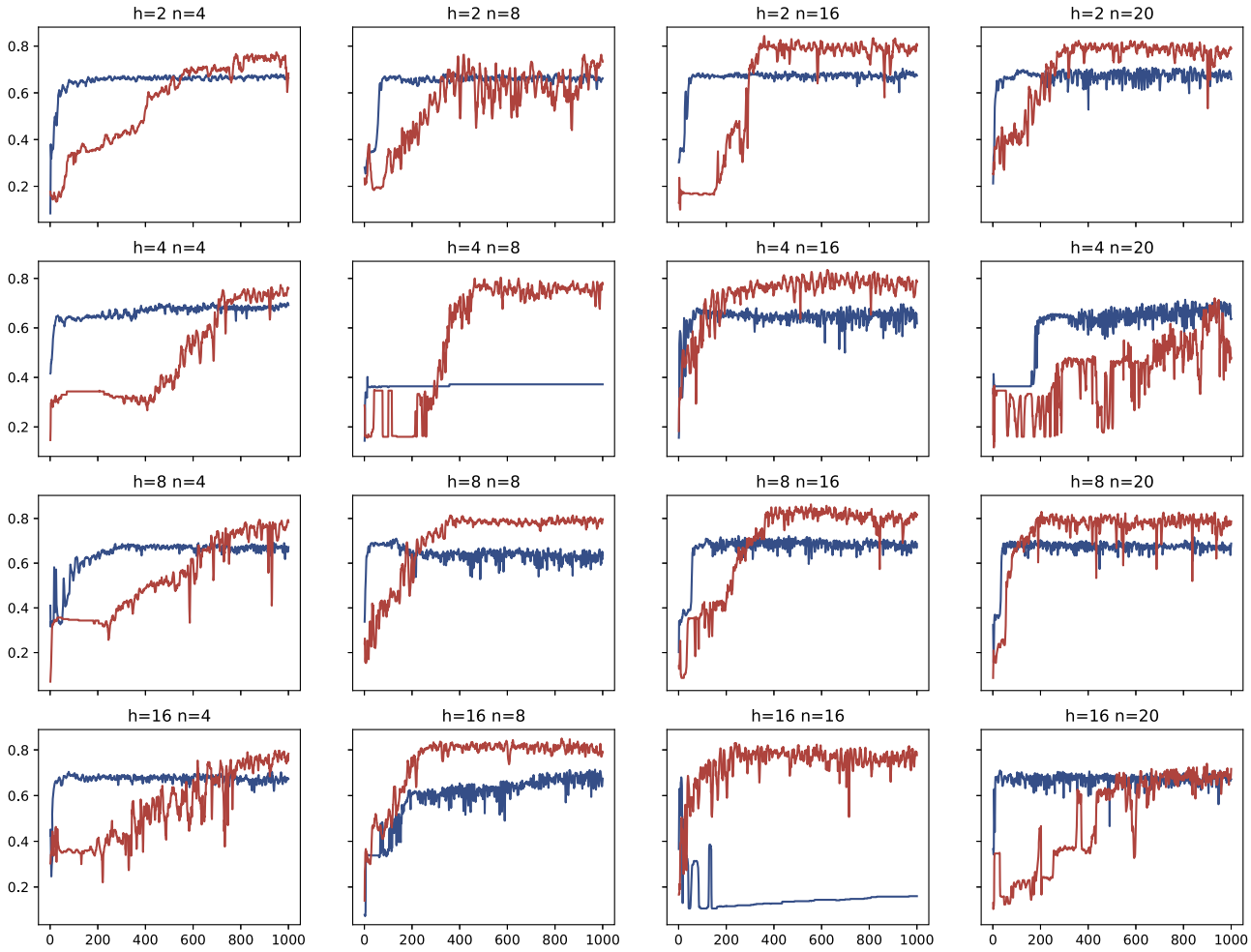


Figure 6. Accuracy changes with the epoch.

that the DGE-GCN-multi model has a superior classification effect.

4.4. Methods contrast

To evaluate the performance of DGE, the proposed DGE-based models are compared with the following representative methods:

- DeepWalk: The node embedding is generated by an unsupervised random walk, and then the embedding is input to the support vector machine classifier for node classification.
- Planetoid: It not only learns node embedding but also can predict the context in the graph. Besides it also uses label information to construct transgenic and induced formulations.
- GCN: The node embedding is generated by truncating Chebyshev polynomials to the one-hop neighborhood.
- GAT: The node embedding is generated by modeling the difference between a node and its one-hop neighbor.
- H-GCN: The global structure features are learned by repeatedly aggregating structurally similar nodes to hyper nodes and then the coarser graph is refined into the original graph to obtain the node embedding.

5. RESULTS

In this experiment, we selected MLP, Deepwalk, Planetoid, GCN, GAT, and H-GCN as the contrast methods, and the comparison is shown in Table 2. The link density (edges/nodes^2) of Cora, CiteSeer, and PubMed is 0.7%, 0.4%, and 0.1%, respectively. The accuracy of the DGE-based models in Cora and CiteSeer is better than other methods, but the advantages in PubMed are not obvious. That means the DGE-based modes are suitable for the network with relatively high link density.

Table 2. Node classification accuracy results

Methods	Datasets		
	Cora	CiteSeer	PubMed
MLP	55.1	46.5	71.4
DeepWalk	67.2	43.2	65.3
Planetoid	75.7	64.7	77.2
GCN	81.5	70.3	78.8
GAT	83.0	72.5	79.0
H-GCN	84.5	72.8	**79.8**
DGE-GCN	80.3	70.2	65.2
DGE-GCN-GAT	82.3	66.3	67.3
DGE-GCN-Multi	**86.33**	**73.2**	71.6
DGE-GAT-Multi	83.6	71.2	67.6

Bold values indicate the highest node classification accuracy across different datasets

6. DISCUSSION

Our experimental results indicate that the DGE framework is effective in node classification tasks, especially in high link density graphs such as the Cora dataset, superior to traditional GCN

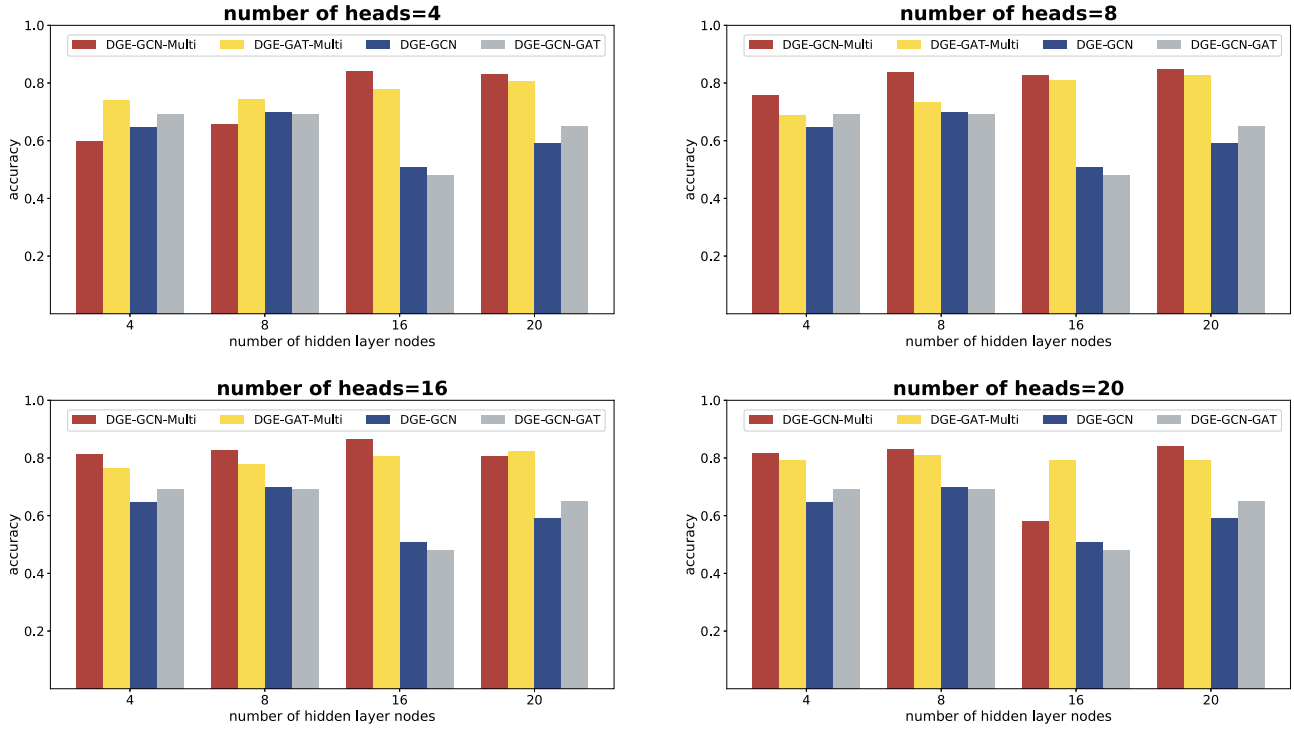


Figure 7. Accuracy comparison on DGE-based models.

and GAT methods, especially the DGE-GCN-MULTI model, the node classification accuracy is about 5% higher than GCN and about 3.5% higher than GAT.

Therefore, the DGE based model can expand the receptive field of node embeddings and can be used as an independent deep learning embedding layer.

The DGE framework can extend the receptive field and more effectively handle long-distance dependencies between nodes in the graph, avoiding the shortcomings of traditional methods in this regard. Therefore, the DGE framework not only solves the limitations of existing methods in expressing deep graph structures, but also provides new ideas and methods for the further development of graph neural networks.

7. CONCLUSION

In this paper, we proposed a DGE framework for the node classification tasks. The DGE framework offers significant advantages in capturing long-range dependencies and expanding the receptive field of graph convolutions. The dual graph structure is used as the representation of the original graph. Node embeddings in the dual graph can get the edge embeddings for the original graph, but also extend the receptive field in the embedding convolution process. The key problem is how to transform the original graph into a dual graph, and how to get the node embeddings on the original graph from the dual graph. To solve these problems, we proposed a DGE framework and designed four DGE-based models, DGE-GCN, DGE-GCN-GAT, DGE-GCN-MULTI, and DGE-GAT-MULTI. Moreover, this extension emphasizes the flexibility and effectiveness of the DGE framework in various applications.

We carried out these four models on Cora, CiteSeer, and PubMed datasets. The experimental analysis shows that the optimal prior parameters (multi-heads h, embeddings dimension n) of DGE based models are $h = 8$ and $n = 8$. In addition, if the link density of the dataset is higher, the advantage of the DGE-based models is more obvious.

The advantage of DGE-based models is that they can expand the receptive field of node embeddings, and can be used as an independent deep learning embedding layer.

The limitation of DGE-based models is that these DGE-based models are relatively complex. The follow-up work of this paper is to reduce the multi-layer DGE parameters to simplify the model structure. Based on DGE, a follow-up pooling strategy can be designed for establishing a multi-layers convolution neural network on a large-scale graph.

CONFLICT OF INTEREST

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

FUNDING

We acknowledge the support of the Natural Science Foundation of Zhejiang Province, China (Grant No. LY22F020001, No. LZ20F020001), the 3315 Plan Foundation of Ningbo (Grant No. 2019B-18-G), China Natural Science Foundation under Grant 62271274, the support of Research Application of A Multi Billion Parameter Monitoring Video Model for domestic full stack AI infrastructure, China (Grant No. 20242004), and Ningbo "Kechuang Yongjiang 2035" Key Technology (Grant No. 2024Z245).

DATA AVAILABILITY

The data that support the findings of this study are openly available in [28].

REFERENCES

1. Fan W, Ma Y, Li Q. *et al*. Graph neural networks for social recommendation. In: *The World Wide Web Conference, San Francisco, CA, 13–17 May*. New York: ACM, 2019, 417–26.

2. You J, Liu B, Ying R. *et al*. Graph convolutional policy network for goal-directed molecular graph generation. In: *Advances in Neural Information Processing Systems 31 (NIPS 2018), Montreal, Canada, 2–8 December*. New York: Neural Information Processing Systems (NIPS), 2018, 1049–5258.
3. Zhang J, Shi X, Xie J. *et al*. GAAN: gated attention networks for learning on large and spatiotemporal graphs. In: *Uncertainty in Artificial Intelligence, Monterey, CA, 6–10 August*. New York: AUAI Presseditorial Office, 2018, 339–49.
4. Hou Z, He Y, Cen Y. *et al*. Graphmae2: a decoding-enhanced masked self-supervised graph learner. In: *Proceedings of the ACM Web Conference 2023, 2 Penn Plaza, New York, 30–4 April*. New York: ACM, 2023, 737–46.
5. Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. *Proceedings of the 5th International Conference on Learning Representations (ICLR 2017)*, Toulon, France, April 24–26, 2017.
6. Velickovic P, Cucurull G, Casanova A. *et al*. Graph attention networks. *stat* 2017; **1050**:10–48550.
7. Hamilton WL, Ying R, Leskovec J. Inductive representation learning on large graphs. In: *Advances in Neural Information Processing Systems 30 (NIPS 2017), Long Beach, CA, 04–09 DEC, Advances in Neural Information Processing Systems*, Vol. **30**. California: Neural Information Processing Systems (NIPS), 2017, 1049–5258.
8. He X, Deng K, Wang X. *et al*. Lightgcn: simplifying and powering graph convolution network for recommendation. In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Electr Network, 25–30 July* New York: ASSOC Computing Machinery, 2020, 639–48.
9. Chen W, Chen L, Xie Y. *et al*. Multi-range attentive bicomponent graph convolutional network for traffic forecasting. In: *Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, 7–12 February*. CA: Assoc Advancement Artificial Intelligence, 2020, 3529–36.
10. Li M, Chen S, Chen X. *et al*. Actional-structural graph convolutional networks for skeleton-based action recognition. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, 16–20 June*. CA: IEEE Computer Soc, 2019, 3595–603.
11. Bruna J, Zaremba W, Szlam A, LeCun Y. Spectral networks and locally connected networks on graphs. *Proceedings of the 2nd International Conference on Learning Representations (ICLR 2014)*, Banff, Canada, April 14–16, 2014.
12. Defferrard M, Bresson X, Vandergheynst P. Convolutional neural networks on graphs with fast localized spectral filtering. In: *Advances in Neural Information Processing Systems 29 (NIPS 2016), Barcelona, Spain, 5–10 December*. Neural Information Processing Systems (NIPS), 2016, 3844–52. California.
13. Monti F, Boscaini D, Masci J. *et al*. Geometric deep learning on graphs and manifolds using mixture model CNNS. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, 21–26 July*. New York: IEEE, 2017, 5115–24.
14. Roweis S, Saul L. Nonlinear dimensionality reduction by locally linear embedding. *Science* 2000; **290**:2323–6.
15. Cao, S., Lu, W., and Xu, Q. (2015) Grarep: learning graph representations with global structural information. *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, Melbourne, Australia, 19–23 October*, 891–900. ACM, New York.
16. Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. *Proceedings of the 1st International Conference on Learning Representations (ICLR 2013)*, Scottsdale, Arizona, USA, May 2–4, 2013.
17. Perozzi B, Al-Rfou R, Skiena S. Deepwalk: online learning of social representations. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, 24–27 August*. New York: Assoc Computing Machinery, 2014, 701–10.
18. Tang J, Qu M, Wang M. *et al*. Line: large-scale information network embedding. In: *Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, 18–22 May*. New York: Assoc Computing Machinery, 2015, 1067–77.
19. Grover A, Leskovec J. node2vec: scalable feature learning for networks. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, 13–17 August*. New York: Assoc Computing Machinery, 2016, 855–64.
20. Wang D, Cui P, Zhu W. Structural deep network embedding. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, 13–17 August*. New York: Assoc Computing Machinery, 2016, 1225–34.
21. Hu F, Zhu Y, Wu S, Wang L, Tan T. Hierarchical graph convolutional networks for semi-supervised node classification. *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)* p. 4532–9.
22. Park C, Kim D, Han J. *et al*. Unsupervised attributed multiplex network embedding. In: *Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, 7–12 February*. CA: Assoc Advancement Artificial Intelligence, 2020, 5371–8.
23. Ranjan E, Sanyal S, Talukdar P. Asap: Adaptive structure aware pooling for learning hierarchical graph representations. In: *Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, 7–12 February*. CA: Assoc Advancement Artificial Intelligence, 2020, 5470–7.
24. Wang X, Zhu M, Bo D. *et al*. AM-GCN: adaptive multi-channel graph convolutional networks. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, ELECTR NETWORK, 23–27 August*. New York: Assoc Computing Machinery, 2020, 1243–53.
25. Lee JB, Rossi R, Kong X. Graph classification using structural attention. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, England, 19–23 August*. New York: Assoc Computing Machinery, 2018, 1666–74.
26. Gong L, Cheng Q. Exploiting edge features for graph neural networks. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, 16–20 June*. New York: IEEE, 2019, 9211–9.
27. Bianchi FM, Grattarola D, Alippi C. Dual-primal graph convolutional networks. *Advances in Neural Information Processing Systems (NeurIPS)*, **33**, 4663–75.
28. Sen P, Namata G, Bilgic M. *et al*. Collective classification in network data. *AI Mag.* 2008; **29**:93–3.

The Computer Journal, 2025, 1–8
https://doi.org/10.1093/comjnl/bxaf086
Original Article