Journal of Systems Engineering and Electronics
Vol. 34, No. 2, April 2023, pp.350 – 359

DHSEGATs: distance and hop-wise structures encoding enhanced graph attention networks

HUANG Zhiguo[1,2,*]

1. Sci-Tech Academy, Zhejiang University, Hangzhou 310030, China;
2. Post-Doctoral Research Center, Hundsun Incorporated, Hangzhou 310038, China

Abstract: Numerous works prove that existing neighbor-averaging graph neural networks (GNNs) cannot efficiently catch structure features, and many works show that injecting structure, distance, position, or spatial features can significantly improve the performance of GNNs, however, injecting high-level structure and distance into GNNs is an intuitive but untouched idea. This work sheds light on this issue and proposes a scheme to enhance graph attention networks (GATs) by encoding distance and hop-wise structure statistics. Firstly, the hop-wise structure and distributional distance information are extracted based on several hop-wise ego-nets of every target node. Secondly, the derived structure information, distance information, and intrinsic features are encoded into the same vector space and then added together to get initial embedding vectors. Thirdly, the derived embedding vectors are fed into GATs, such as GAT and adaptive graph diffusion network (AGDN) to get the soft labels. Fourthly, the soft labels are fed into correct and smooth (C&S) to conduct label propagation and get final predictions. Experiments show that the distance and hop-wise structures encoding enhanced graph attention networks (DHSEGATs) achieve a competitive result.

Keywords: graph attention network (GAT), graph structure information, label propagation.

DOI: 10.23919/JSEE.2023.000057

1. Introduction

Many works prove that neighbor-averaging graph neural networks (GNNs) cannot efficiently catch structure information. Such GNNs cannot even capture degree information in some cases. The reason is rather intuitive: as the neighbor-averaging GNNs can only aggregate the neighbors' features of a node, if the neighbors' features contain no structure information, then the hop-wise neighbor-averaging GNNs can only catch degree information at best [1−3]. Therefore, it is an intuitive idea that injecting structure information into nodes' feature vectors may improve the performance of GNNs.

Manuscript received August 15, 2022.
*Corresponding author.

Based on the abovementioned idea, numerous works have shown that injecting structure, distance, position, or spatial information can significantly improve the performance of neighbor-averaging GNNs [4−10]. However, existing works have their problems. Some of them are with incredibly high computation complexity, which cannot apply to large-scale graphs (MotifNet [4]). Some of them concatenate structure information with intrinsic feature vectors such as the ID-GNN [6], position GNN (P-GNN) [8], distance encoding GNN (DE-GNN) [9], which may confuse the signals of different features. For example, in ogbn-arxiv dataset, the intrinsic feature is a semantic embedding of headline or abstract, which provides a totally different signal with the structure information. Some of them are graph-level-task oriented and only deal with small graphs (Graphormer [7], SubGNN [10]). Moreover, current works consider only simple structure information such as degree and circles, remaining high-level structure information untouched. More importantly, hop-wise neighbor-averaging GNNs' sensitive field is restricted to only one hop and thus cannot catch multi-hop structure information. Injecting hop-wise structure information also remains untouched.

This work sheds light on efficiently injecting high-level multi-hop structure and distance information into graph attention networks (GATs). A combination of structure statistics with $O(|\boldsymbol{N}|+|\boldsymbol{E}|)$ computation complexity (where $|\boldsymbol{N}|$ denotes the number of nodes and $|\boldsymbol{E}|$ denotes the number of edges, respectively) is proposed by investigating the computation complexity of most node-level and graph-level structure statistics. The work first extracts k hop-wise ego-nets for every node to compute node-level and graph-level structure statistics. At the same time, a distance sequence of k-hop ego-net is extracted to compute distributional distance statistics. The distance statistics, hop-wise structure statistics, and node's intrinsic features are encoded into the same vector space and then added together, just like Transformer's initial

embedding. Derived feature vectors are fed into GATs to get initial predictions. These initial predictions are then fed into correct and smooth (C&S) to inject global label information to get final predictions. Experiments show that the scheme can significantly improve the performance of baseline GATs.

The contributions of this work are concluded as follows:

(i) By investigating the computation complexity of most structure statistics and the effectiveness of distance information in empowering GNNs, the work proposes a combination of structure statistics with $O(|\boldsymbol{N}|+|\boldsymbol{E}|)$ computation complexity and a distributional distance statistics extraction scheme that does not corrupt the graph's original structure.

(ii) The work proposes a hop-wise high-level distance and structure information injecting scheme which fits universal neighbor-averaging GNN models.

(iii) The work conducts extensive experiments to demonstrate the effectiveness of injecting distance and hop-wise structure information into GATs.

2. Related work

2.1 Structure and distance information in GNNs

ID-GNN argues that GNNs cannot outperform the 1-WL test without external features, and injecting the number of circles would make it more powerful. In-degree and out-degree embedding is an integral part of Graphormer for winning the OGB Large-Scale Challenge 2021. MotifNet significantly improves GNNs' expressive power by injecting the number of motifs of the graph. By subgraph isomorphism counting, graph substrecture network (GSN) [5] enlarges the expressive power of GNNs. Dasoulas et al. [11] proposed a method called colored local iterative procedure (CLIP) to inject structure information into GNNs, which also improves the performance of GNNs.

Besides in-degree and out-degree, spatial information is also an integral part of Graphormer. By injecting position information, P-GNN achieves significant improvement compared with baseline GNNs. By injecting distance encoding information, DE-GNN outperforms baseline GNNs. SubGNN achieves state-of-the-art (SOTA) performance on subgraph classification by injecting neighborhood, structure, and position information.

2.2 Disadvantage of neighbor-averaging GNNs

Early works that analyze the disadvantage of GNNs come from the aspect of the over-smoothing problem. Li et al. [1] analyzed the asymptotic behavior of over-smoothing and found that GNNs can only catch degree information without a nonlinear activation function. Oono and Suzuki [2] made further efforts and demonstrated that even with a nonlinear activation function, GNNs can also only catch degree information. Xu et al. [12] demonstrated that message-passing GNNs cannot outperform 1-WL in the aspect of expressive power. Dehmamy et al. [3] showed that messages passing GNNs cannot even catch degree information without proper convolution support.

Another idea to analyze the disadvantage of GNNs is to inspect the ability of GNNs to count simple local structure statistics such as circles and triangles. Their finds are consistent with graph isomorphism network (GIN) [12], which can be concluded that message-passing GNNs cannot count local structure features that the 1-WL test cannot [5,13−15]. Some works inspect the GNNs' spectral ability and find that GNNs are nothing but low-pass filters [16]. Scalable inception graph network (SIGN) [17] and scalable adptive graph network (SAGN) [18] show that by concatenating graph diffusion processed features, multi-layer perceptron (MLP) outperforms most complicated GNNs, which demonstrates the disadvantage of GNNs in another way.

2.3 Graph attention mechanism

The reason why GATs are preferred among a large number of GNN models lies in their two distinctive advantages. First, according to GIN, the key to the expressive power of a message-passing GNN is whether its aggregating function is multi-set injective. The attention mechanism of GATs intrinsically provides a multi-set injective aggregating function. Second, GNNs are troubled by the problem of over-smoothing. However, because of residual connection and attention mechanism, GATs can theoretically eliminate over-smoothing and make the number of layers go deep.

Because of the two advantages mentioned above, the attention mechanism on the graph arouses enormous interest. Representative works including GAT [19], graph tranformer network (GTN) [20], GPT-GNN [21], heterogeneous graph transformer (HGT) [22], Graphormer, and adaptive graph diffusion network (AGDN) [23]. GAT is the first work that combines message-passing GNNs with the attention mechanism. After that, GTN further combines GNN with Transformer, and GPT-GNN combines GNN with GPT. HGT adapts graph transformer for the heterogeneous graphs. Compared with GATs mentioned above, Graphormer adopts a full transformer and gets rid of the vanilla message-passing mechanism. AGDN combines graph attention mechanism and graph diffusion process, which outperforms most of the existing single GNN models on ogbn-arxiv dataset.

2.4 Label propagation in GNNs

Label propagation in graphs has been proven to be an

effective way to inject global label information in the context of semi-supervised learning. Its basic assumption is that the labels of adjacent nodes have a higher probability of being the same than those that are not adjacent. Recently, a large number of works have concentrated on combining GNNs with label propagation. Some works directly combine label propagation with GNNs, such as taking label propagation as an extra loss [24]. Some works inject label information by Markov random field [25,26]. Some works take label information as the extra features [27]. Some works first do error correction of label predictions of the base model and then use the error correction to smooth predictions, which gets surprisingly good performance even combining with MLP [28,29].

3. Methodology

3.1 Overall architecture

The overall architecture of the model is depicted in Fig. 1. The architecture is divided into four modules: module 1 denotes raw distance and hop-wise structure information extracting, module 2 denotes distance and hop-wise structure information encoding, module 3 denotes GATs, and module 4 denotes C&S.

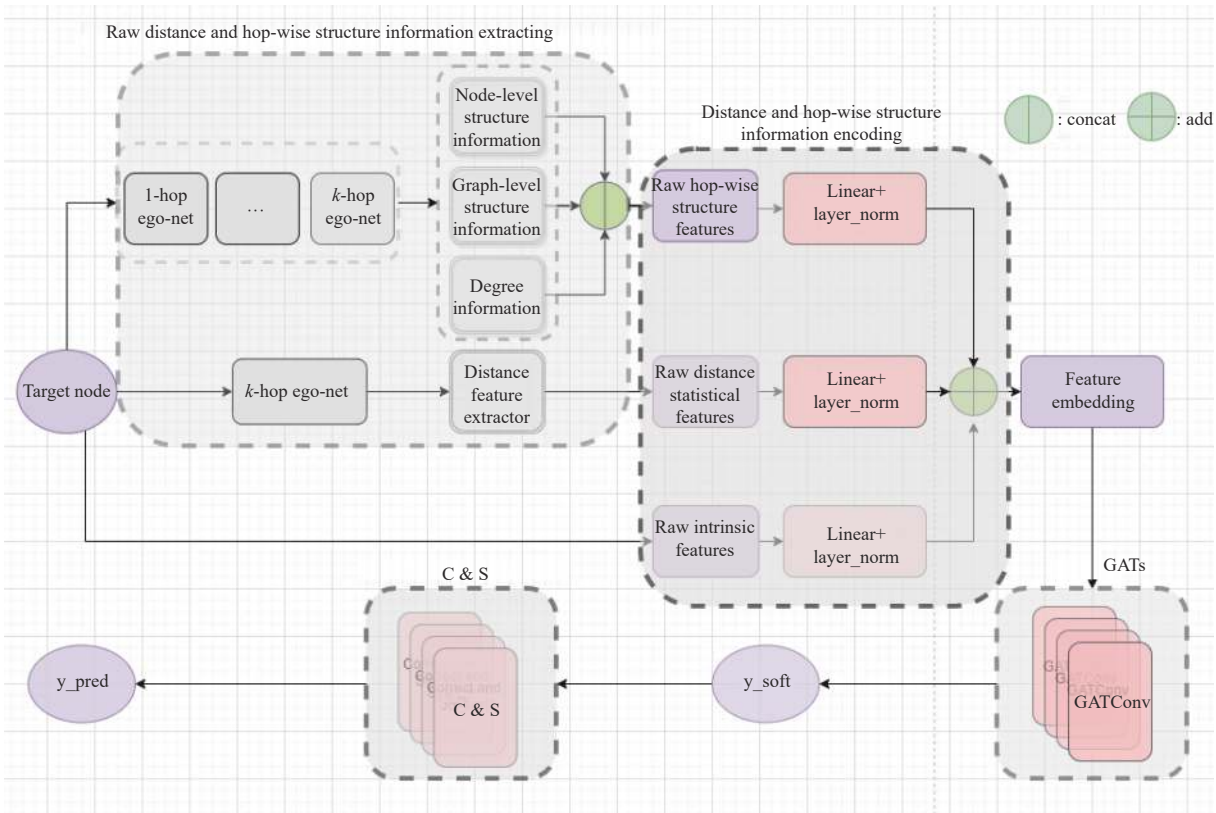

Fig. 1 Overall architecture of DHSEGATs

For every target node, module 1 first extracts its k hop-wise ego-nets around it (k denotes the predefined largest hop). After that, module 1 could be split into two parts. For every ego-net, part 1 of module 1 computes its node-level structure information, graph-level structure information, and degree information. The node-level structure information includes triangles, clustering, and square-clustering. The graph-level structure information includes density, number of self-loops, and transitivity (see Subsection 3.2.1 for more detail). The degree information includes degree only for undirected graphs and in-degree & out-degree for directed graphs (our case in the experiment). All the extracted statistics of part 1 are concatenated together as a vector (denoted as raw hop-wise structure features). Part 2 of module 1 only needs the k-hop ego-net. It is designed to extract distributional distance information of the k-hop ego-net (see Subsection 3.2.2 for more detail). The extracted distance information is also in the format of a vector (denoted as raw distance statistical features). The raw hop-wise structure features, raw distance statistical features, and the original features of the target node (denoted as raw intrinsic features) are fed into module 2 as its input.

Module 2 is designed to encode the input into the same dimension because the abovementioned three kinds of features are not in the same dimension and thus cannot be

added together. Module 2 inherits the idea of the transformer, which encodes different kinds of embeddings and then adds them together to fuse different types of information. Module 2 consists of three encoding layers. Each layer consists of a linear layer and a norm layer (denoted as linear + layer_norm). The three kinds of encoded features are then added together (denoted as feature embedding) and fed into module 3.

Module 3 is the graph attention module. It consists of several graph attention layers (denoted as GATConv). The choice of graph attention layer is relatively free. All GNN models that contain attention mechanisms are acceptable. Module 3 accepts the feature embedding and conducts neighbor-averaging to train the model. The trained model outputs the predictions of the test or valid samples (denoted as y_soft). The output of module 3 is fed into module 4 as its input.

Module 4 is the C&S module. C&S is a label propagation algorithm. It is so effective that even combined with MLP could make the derived model outperform most complicated GNN models. It consists of several C&S layers, each having a correct layer and a smooth layer. With the output of module 3, module 4 implements label propagation and gets the final predictions (denoted as y_pred). Once getting the final predictions, the procedure of distance and hop-wise structures encoding enhanced graph attention networks (DHSEGATs) is finished.

3.2 Preprocessing

3.2.1 Node-level and graph-level features of hop-wise ego-nets

As mentioned above, neighbor-averaging GNNs can only catch degree information at best, and degree information is merely one-hop node-level structure information. In order to catch multi-hop structure information of a node, k r-hop ego-nets ($r = 1, 2, \cdots, k$) are extracted so that their node-level and graph-level structural graph statistics can be computed. However, the computation complexities of many structural graph statistics are incredibly high. For example, the computation complexity of eigenvector centrality is $O(|\boldsymbol{N}|^3)$, and betweenness centrality is $O(|\boldsymbol{N}|\cdot|\boldsymbol{E}|)$. It is unrealistic to apply them to a large-scale graph if the preprocessing has high computation complexity. Therefore, the computation complexities of most structural graph statistics are investigated in the first place. According to the investigation, the following statistics are chosen to construct a combination of structural graph statistics. Their overall computation complexity is $\boldsymbol{O}(|\boldsymbol{N}|+|\boldsymbol{E}|)$, which could be easily scaled to a large-scale graph.

The abovementioned combination of structural graph statistics consists of three types of information: degree, node-level structure information, and graph-level structure information.

The node-level structure information contains Triangles, Clustering, and Square Clustering.

Triangles: the number of triangles that contain the target node in the ego-net.

Clustering: $c_u = \dfrac{2T(u)}{\deg(u)(\deg(u)-1)}$, where $T(u)$ is the number of triangles through node u and $\deg(u)$ is the degree of u.

Square-clustering: $C_4(v) = \dfrac{\sum_{u=1}^{k_v}\sum_{w=u+1}^{k_v} q_v(u,w)}{\sum_{u=1}^{k_v}\sum_{w=u+1}^{k_v}[a_v(u,w)+q_v(u,w)]}$,

where $q_{v(u,w)}$ is the number of common neighbors of u and w other than v (i.e., squares), and $a_v(u,w) = (k_u - (1+q_v(u,w)+\theta_{uv})) + (k_w - (1+q_v(u,w)+\theta_{uw}))$, where $\theta_{uw} = 1$ if u and w are connected and 0 otherwise.

The graph-level structure information contains Density, Number of self-loops, and Transitivity.

Density: $d = \dfrac{2m}{n(n-1)}$, where n is the number of nodes and m is the number of edges in graph.

Number of self-loops: number of self-loop edges, where a self-loop edge has the same node at both ends.

Transitivity: $T = 3\dfrac{\#\text{triangles}}{\#\text{triads}}$, where a triad is a graph that consists of two edges with a shared vertex.

3.2.2 Distributional distance features

Distance statistics are crucial information for the attention mechanism. In order to get a larger sensitive field of distance information, the k-hop ego-net around the target node is extracted first so that the unweighted distance of every neighbor node to the target node in the ego-net can be computed. By extraction and computation, a distance sequence is generated. However, the generated sequence could not be directly applied to GNNs because the sequence is variant and may be very long. Hence, it should be clipped to a fixed-length vector beforehand. Technically, we can adopt neighbor sampling or the padding technique. However, such methods may corrupt the graph's structure or introduce a virtual node, which may result in disastrous encoding consequences. Hence, an encoding method based on distributional statistics is adopted to compute sequence statistics. Seven sequence statistics are chosen to get comprehensive information: maximum, minimum, median, mean, standard deviation, kurtosis, and skewness.

3.3 Encoding distance and hop-wise structure features

In order to encode distance and hop-wise structure infor-

mation, we should first encode the distance features, hop-wise structure features, and intrinsic features to the same vector space. Such operation could be conducted by three linear layers:

$$\boldsymbol{H}_s = \boldsymbol{W}_s \boldsymbol{F}_s, \tag{1}$$

$$\boldsymbol{H}_d = \boldsymbol{W}_d \boldsymbol{F}_d, \tag{2}$$

$$\boldsymbol{H}_i = \boldsymbol{W}_i \boldsymbol{F}_i, \tag{3}$$

where $\boldsymbol{F}_s, \boldsymbol{F}_d$, and $\boldsymbol{F}_i$ denote the structure features vector, distance features vector, and intrinsic features vector, respectively, and $\boldsymbol{W}$ denotes the linear map layer.

Layer norm modules are followed to avoid one of the three kinds of features dominating the derived initial encoding vector:

$$\boldsymbol{H}_s = \text{layer\_norm}(\boldsymbol{H}_s), \tag{4}$$

$$\boldsymbol{H}_d = \text{layer\_norm}(\boldsymbol{H}_d), \tag{5}$$

$$\boldsymbol{H}_i = \text{layer\_norm}(\boldsymbol{H}_i). \tag{6}$$

The three encoded vectors are added together to get the initial encoding vector, and the spatial complexity is reduced as follows:

$$\boldsymbol{H}^{(0)} = \boldsymbol{H}_s + \boldsymbol{H}_d + \boldsymbol{H}_i. \tag{7}$$

3.4 Attention mechanism on graph

Two GAT models are investigated in this work: GAT and AGDN. The former is the most classical graph attention model and a widely used baseline model. The latter achieves the best performance on ogbn-arxiv dataset among single models by combining attention mechanisms and the graph diffusion process.

3.4.1 GAT

The general form of neighbor-averaging GNN could be written as

$$\boldsymbol{H}_i' = \text{UPD}_W(\boldsymbol{h}_i, \text{AGG}_{j\in N(i)} \text{MES}_W(\boldsymbol{H}_i, \boldsymbol{H}_j, \boldsymbol{e}_{j,i})) \tag{8}$$

where $\boldsymbol{H}$, AGG, MES, and UPD denote nodes' representation matrix, aggregating, message, and updating functions, respectively. GAT's AGG is a linear layer, and MES is a multi-head attention function. The attention mechanism in GAT could be described as follows:

$$\boldsymbol{H}_i'(K) = \text{Pool}_{k=1}^{K} \boldsymbol{H}_{ik}', \tag{9}$$

$$\boldsymbol{H}_i' = \sigma\left(\sum_{j\in N_i} \alpha_{ij} \boldsymbol{W} \boldsymbol{H}_j\right), \tag{10}$$

$$\alpha_{i,j} = \frac{\exp(\sigma(\boldsymbol{a}^{\mathrm{T}}[\boldsymbol{\Theta}\boldsymbol{x}_i \,\|\, \boldsymbol{\Theta}\boldsymbol{x}_j]))}{\sum_{k\in N(i)\cup\{i\}} \exp\left(\sigma\left(\boldsymbol{a}^{\mathrm{T}}[\boldsymbol{\Theta}\boldsymbol{x}_i \,\|\, \boldsymbol{\Theta}\boldsymbol{x}_k]\right)\right)}, \tag{11}$$

where Pool denotes the pooling function in multi-heads (our case in the experiments), K denotes the number of heads, α_{ij} denotes the normalized attention score of the source node j on the target node i, and σ denotes the activation function.

3.4.2 AGDN

By denoting transition matrix, parameters matrix, and attention vector as $\boldsymbol{T}^{(l)}$, $\boldsymbol{W}^{(l)}$ and $\boldsymbol{a}_{hw}^{(l)}$, respectively, a GAT-HA layer of the AGDN model could be defined as

$$\boldsymbol{H}^{(l)} = \text{AGDN}(\mathcal{G}, \boldsymbol{H}^{(l-1)}, \boldsymbol{W}^{(l)}, \boldsymbol{a}_{hw}^{(l)}), \tag{12}$$

$$\widehat{\boldsymbol{H}}^{(0,l)} = \boldsymbol{H}^{(l-1)} \boldsymbol{W}^{(l)}, \tag{13}$$

$$\widehat{\boldsymbol{H}}^{(k,l)} = \boldsymbol{T}^{(l)} \widehat{\boldsymbol{H}}^{(k-1,l)}. \tag{14}$$

To be more specific, the output of the GAT-HA layer could be written as

$$\boldsymbol{H}^{(l)} = \sum_{k=0}^{K} \boldsymbol{\Theta}^{(k,l)} \widehat{\boldsymbol{H}}^{(k,l)} + \boldsymbol{H}^{(l-1)} \boldsymbol{W}_r^{(l)}, \tag{15}$$

$$\boldsymbol{\Theta}^{(k,l)} = \text{diag}(\theta_1^{(k,l)}, \cdots, \theta_N^{(k,l)}), \tag{16}$$

$$\theta_i^{(k,l)} = \frac{\exp\left(\sigma\left(\left[\widehat{\boldsymbol{H}}_i^{(0,l)} \,\|\, \widehat{\boldsymbol{H}}_i^{(k,l)}\right] \cdot a_{hw}^{(l)}\right)\right)}{\sum_{k=0}^{K} \exp\left(\sigma\left(\left[\widehat{\boldsymbol{H}}_i^{(0,l)} \,\|\, \widehat{\boldsymbol{H}}_i^{(k,l)}\right] \cdot a_{hw}^{(l)}\right)\right)}, \tag{17}$$

where $\widehat{\boldsymbol{H}}_i^{(k,l)}$ denotes the representation vector of layer l in aggregation k on node i.

3.5 C&S

The C&S is chosen among a large number of label propagation algorithms on GNN because it has two advantages: First, C&S is such an effective label propagation algorithm that it could make the derived model outperform most complicated GNN models even combined with MLP. Second, as a post-processing module, C&S does not disturb the base model so that we can evaluate the base model properly. C&S contains two steps: the Correct module corrects the base predictions by modeling correlated error, and the Smooth module smooths the soft predictions by correlated error.

3.5.1 Correcting

The basic idea of the Correct module is that errors in the base predictions are positively correlated along edges in the graph. In other words, an error at node i increases the chance of a similar error at neighbor nodes of i. Denot-

ing the error matrix as $\boldsymbol{E}$, we can get

$$\begin{cases} \boldsymbol{E}_{L_t} = \boldsymbol{Z}_{L_t} - \boldsymbol{Y}_{L_t} \\ \boldsymbol{E}_{L_v} = 0 \\ \boldsymbol{E}_{U} = 0 \end{cases} \tag{18}$$

where $\boldsymbol{L}_t, \boldsymbol{L}_v, \boldsymbol{U}, \boldsymbol{Z}$, and $\boldsymbol{Y}$ denote training labeled data, validating labeled data, unlabeled data, base predictions, and ground-truth labels, respectively.

After that, the label spreading technique is used to smooth the error matrix as follows:

$$\widehat{\boldsymbol{E}} = \underset{\boldsymbol{W}\in\mathbf{R}^{n\times c}}{\operatorname{argmax}} \operatorname{trace}\left(\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{I}-\boldsymbol{S})\boldsymbol{W}\right) + \mu \parallel \boldsymbol{W}-\boldsymbol{E}\parallel_{\mathrm{F}}^{2} \tag{19}$$

where $\boldsymbol{S} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$, and $\boldsymbol{I}$ denotes the unit matrix.

Then, we can get the corrected predictions by adding the solution of (19) to $\boldsymbol{Z}$ with some scale method:

$$\boldsymbol{Z}^{(r)} = f(\boldsymbol{Z}, \widehat{\boldsymbol{E}}) \tag{20}$$

where f denotes scale and add operation, here we adapt scaled fixed diffusion (FDiff-scale). Then we get

$$\boldsymbol{Z}^{(r)} = \boldsymbol{Z} + s\widehat{\boldsymbol{E}} \tag{21}$$

where s is a hyper-parameter.

3.5.2 Smoothing

We can smooth $\boldsymbol{Z}^{(r)}$ to get final predictions based on the assumption that adjacent nodes in the graph are likely to have similar labels. Start with the best guess $\boldsymbol{G}$:

$$\begin{cases} \boldsymbol{G}_{L_t} = \boldsymbol{Y}_{L_t} \\ \boldsymbol{G}_{L_v,U} = \boldsymbol{Z}^{(r)}_{L_v,U} \end{cases}. \tag{22}$$

Implementing the following iterating procedure until convergence, we get the final predictions

$$\boldsymbol{G}^{(t+1)} = (1-\alpha)\boldsymbol{G} + \alpha\boldsymbol{S}\boldsymbol{G}^{(t)} \tag{23}$$

where $\boldsymbol{G}^{(0)} = \boldsymbol{G}$.

4. Experiments

4.1 Dataset and settings

The dataset used in this work is ogbn-arxiv. It is a directed graph representing the citation network between all computer science (CS) arXiv papers indexed by MAG, with 169 343 nodes and 1 166 243 edges. Each node represents an arXiv paper, and each directed edge represents a citation. Each paper has a 128-dimensional feature vector obtained by averaging the embeddings of words in the title and abstract. In addition, all papers are linked to the year in which the corresponding paper was published. The task of ogbn-arxiv is to predict the 40 subject areas to which arXiv CS papers belong (e.g., cs.AI, cs.LG, and cs.OS), whose labels are manually tagged by the authors of the papers and arXiv moderators. Ogbn-arxiv splits data based on publication date. The author of the dataset suggests taking the papers published before 2017 as the training set, the papers published in 2018 as the validation set, and the papers published in 2019 as the test set. The official evaluation indicator of ogbn-arxiv is accuracy.

The experimental programming language is Python, the deep learning framework is Pytorch, and the server configuration is CPU: Intel(R) Xeon(R) CPU E5-1 630 v4 @ 3.70 GHz, GPU: GeForce RTX 2080 Ti, memory: 128G. The hyper-parameters of the main model are shown in Table 1, and the hyper-parameters of the C&S module are shown in Table 2.

Table 1 Hyperparameters of main model

Hyperparameter	Value	Hyperparameter	Value
num_heads	3	scheduler	Cosine
learning_rate	0.002	num_layers	3
optimizer	Adam	num_hiddens	256

Table 2 Hyperparameters of C&S module

Hyperparameter	Value	Hyperparameter	Value
smooth_adj	DAD	correct_adj	DAD
smooth_alpha	0.756	correct_alpha	0.979
smooth_layers	50	correct_layers	50

The comparison models include GAT, AGDN (GAT-HA+3_heads), GAT+C&S, AGDN+C&S, and UniMP_v2 [30]. GAT is the most primitive and basic node-level attention GNN. AGDN is a GNN with two-level attention and three attention heads with GAT-HA as the primary neural network layer. GAT+C&S and AGDN+ C&S use C&S for further correction based on GAT and AGDN. UniMP_v2 is a unified label propagation GNN model developed by the Baidu Paddle Graph Learning team, which is the most powerful single GNN model.

4.2 Results

4.2.1 Performance comparison

The main results of the performance comparison are presented in Table 3. It shows that the performance of GAT and AGDN is significantly improved when combined with DHSE and C&S. It demonstrates that DHSE provides GATs with high-level multi-hop distance and structure information that GATs cannot catch with the original neighbor-averaging mechanism. DHSEGAT=DHSE+GAT+C&S, DHSEAGDN=DHSE+AGDN+C&S.

Table 3 Performance comparison

Model	Valid accuracy	Test accuracy
UniMP_v2	0.750 6 ± 0.000 9	0.739 7 ± 0.001 5
GAT + C&S	0.748 4 ± 0.000 7	0.738 6 ± 0.001 4
AGDN (GAT-HA+3_heads)	0.748 3 ± 0.000 9	0.737 5 ± 0.002 1
AGDN+C&S	0.747 1 ± 0.000 7	0.738 7 ± 0.001 8
DHSEGAT	0.744 1 ± 0.001 2	0.742 5 ± 0.001 6
DHSEAGDN	0.746 2 ± 0.001 3	**0.743 9 ± 0.001 9**

4.2.2 Performance of DHSEAGDN and DHSEGAT with and without C&S

This work argues that DHSE provides GATs with high-level multi-hop distance and structure information that the neighbor-averaging mechanism cannot catch. However, GAT's classifier may confound the intrinsic features with such information without C&S. As a result, the performance of the enhanced model may not be significantly improved. In order to test this hypothesis, this work compares the performance of DHSEGAT and DHSEAGDN with and without the C&S module. The models without C&S are denoted as DHSE+AGDN and DHSE+GAT. Since DHSE+AGDN and DHSE+GAT get their best performance in different hyper-parameter settings, we report the results separately. The results are presented in Table 4 and Table 5. They show that with the same hyper-parameter settings, the performance of enhanced models (i.e., DHSE+AGDN and DHSE+GAT) and base models (i.e., AGDN and GAT) are very close. However, with C&S, their differences are significant. It indicates that only if combined with C&S can those three types of information be effectively used.

Table 4 Best performance of DHSEAGDN with and without C&S

Model	Valid accuracy	Test accuracy
DHSE+AGDN	0.742 7 ± 0.000 7	0.728 3 ± 0.003 0
DHSE+AGDN+C&S	0.746 2 ± 0.001 3	**0.743 9 ± 0.001 9**
AGDN	0.742 9 ± 0.001 0	0.729 7 ± 0.002 1
AGDN+C&S	0.747 1 ± 0.000 7	0.738 7 ± 0.001 8

Table 5 Best performance of DHSEGAT with and without C&S

Model	Valid accuracy	Test accuracy
DHSE+GAT	0.741 2 ± 0.000 8	0.726 1 ± 0.002 1
DHSE+GAT+C&S	0.744 1 ± 0.001 2	**0.742 5 ± 0.001 6**
GAT	0.741 9 ± 0.000 5	0.727 3 ± 0.001 2
GAT+C&S	0.744 0 ± 0.001 2	0.737 1 ± 0.002 3

4.2.3 Marginal improvement with same hyper-parameters as AGDN

In order to demonstrate the marginal improvement of models with DHSE, this work tests the models with the same hyperparameters setting as AGDN. The results are presented in Table 6. It shows that DHSE can significantly improve the performance of base models with the combination of C&S.

Table 6 Marginal improvement of DHSE

Model	Valid accuracy	Test accuracy
DHSE+AGDN	0.746 4 ± 0.000 8	0.734 6 ± 0.000 7
DHSE+AGDN+C&S	0.748 5 ± 0.000 8	**0.741 4 ± 0.001 0**
AGDN	0.743 2 ± 0.000 7	0.734 4 ± 0.001 8
AGDN+C&S	0.744 7 ± 0.000 9	0.739 8 ± 0.001 0
DHSE+GAT	0.744 8 ± 0.000 7	0.730 8 ± 0.001 4
DHSE+GAT+C&S	0.747 1 ± 0.000 6	**0.738 8 ± 0.001 7**
GAT	0.741 4 ± 0.000 5	0.729 8 ± 0.002 2
GAT+C&S	0.743 8 ± 0.000 6	0.736 9 ± 0.001 8

4.3 Ablation study

In this section, the effectiveness of every component of DHSE in improving the base models' performance is tested. DHSE consists of three parts: encoding layers, structure information, and distance information. The ablation study of the three parts is presented below.

4.3.1 Encoding layers

In order to test the effectiveness of the encoding module, the encoding layers are removed from DHSEAGDN and DHSEGAT separately. The derived models are denoted as DHSEAGDN-encoding and DHSEGAT-encoding, respectively. In order to test the pure effectiveness of the encoding layer to DHSE, the C&S is removed from DHSEGAT-encoding and DHSEGAT-encoding. The derived models are denoted as DHSEAGDN-encoding-C&S and DHSEGAT-encoding-C&S, respectively. The results are presented in Table 7. It shows that encoding layers are an indivisible part of DHSEGATs. The performance of models without encoding layers is inferior to the corresponding base models.

Table 7 DHSEGATs without encoding layers

Model	Valid accuracy	Test accuracy
AGDN-encoding-C&S	0.739 8 ± 0.002 0	0.730 3 ± 0.002 2
DHSEAGDN-encoding	0.738 4 ± 0.002 3	**0.737 7 ± 0.002 6**
GAT-encoding-C&S	0.737 8 ± 0.000 6	0.724 4 ± 0.001 8
DHSEGAT-encoding	0.740 7 ± 0.001 4	**0.737 4 ± 0.002 0**

4.3.2 Structure information

The structure information of DHSE consists of three levels: degree, node level, and graph level. In order to test the effectiveness of these three types of structure information, they are removed from DHSEGAT and DHSEAGDN separately. These derived models are denoted as DHSEGAT-degree, DHSEGAT-node, DHSEGAT-graph, DHSEAGDN-degree, DHSEAGDN-node, and DHSEAGDN-graph, respectively. In order to test the pure effectiveness of the structure information to DHSE, the C&S is removed from above mentioned six derived models. These models are denoted as DHSEAGDN-degree-C&S, DHSEAGDN-node-C&S, DHSEAGDN-graph-C&S, DHSEGAT-degree-C&S, DHSEGAT-node-C&S, and DHSEGAT-graph-C&S, respectively. The results are presented in Table 8, Table 9, and Table 10. They show that degree, node-level structure, and graph-level structure information contribute a similar marginal improvement to DHSEGATs.

Table 8 DHSEGATs without degree information

Model	Valid accuracy	Test accuracy
DHSEAGDN-degree-C&S	0.743 7 ± 0.000 5	0.731 5 ± 0.001 3
DHSEAGDN-degree	0.744 3 ± 0.001 0	**0.741 7 ± 0.000 8**
DHSEGAT-degree-C&S	0.742 3 ± 0.000 5	0.728 6 ± 0.002 1
DHSEGAT-degree	0.744 7 ± 0.001 0	0.737 9 ± 0.002 2

Table 9 DHSEGATs without node-level structure information

Model	Valid accuracy	Test accuracy
DHSEAGDN-node-C&S	0.744 7 ± 0.000 8	0.726 9 ± 0.001 2
DHSEAGDN-node	0.747 4 ± 0.001 0	**0.742 5 ± 0.001 5**
DHSEGAT-node-C&S	0.743 8 ± 0.000 4	0.730 1 ± 0.002 2
DHSEGAT-node	0.745 0 ± 0.001 1	0.740 2 ± 0.001 2

Table 10 DHSEGATs without graph-level structure information

Model	Valid accuracy	Test accuracy
DHSEAGDN-graph-C&S	0.744 4 ± 0.000 8	0.729 0 ± 0.003 6
DHSEAGDN-graph	0.746 3 ± 0.001 6	**0.741 9 ± 0.002 8**
DHSEGAT-graph-C&S	0.742 2 ± 0.000 6	0.726 5 ± 0.001 2
DHSEGAT-graph	0.745 0 ± 0.001 6	0.739 9 ± 0.001 9

4.3.3 Distributional distance information

In order to test the effectiveness of the distance information, the distributional distance features are removed from DHSEAGDN and DHSEGAT. These two modified models are denoted as DHSEAGDN-distance and DHSEGAT-distance, respectively. In order to test the pure effectiveness of the distributional distance information of DHSE, C&S is removed from DHSEGAT-distance and DHSEGAT-distance. The derived two models are denoted as DHSEAGDN-distance-C&S and DHSEGAT-distance-C&S, respectively. The results are presented in Table 11. It shows that distributional distance information provides indivisible information to DHSEAGDN but not to DHSEGAT. Without distributional distance information, the performance of DHSEGAT even improves slightly. We can presume that different GNN models need different information. For GAT, distance information is not indivisible. However, for AGDN, distance information is indivisible because of its graph diffusion process.

Table 11 DHSEGATs without distributional distance information

Model	Valid accuracy	Test accuracy
DHSEAGDN-distance-C&S	0.746 5 ± 0.000 4	0.731 5 ± 0.003 2
DHSEAGDN-distance	0.747 3 ± 0.001 4	**0.740 8 ± 0.002 0**
DHSEGAT-distance-C&S	0.742 9 ± 0.000 6	0.730 3 ± 0.001 7
DHSEGAT-distance	0.744 3 ± 0.000 7	0.739 4 ± 0.002 0

5. Conclusions

The disadvantage of GNNs in catching structure information of graphs has been broadly proven, and the ability of structure and distance information to improve the performance of GNNs has been widely proven as well. Inspired by Graphormer, this work investigates popular structure statistics' computation complexity to propose a combination of structure statistics with $O(|\boldsymbol{N}|+|\boldsymbol{E}|)$ computation complexity and a distributional distance information encoding scheme. Based on the distance and hop-wise structure information, this work further proposes a method that enables GNN with expressive power beyond the 1-WL test and fits universal GNN models.

Through detailed experiments, this work demonstrates that the label information that C&S provides is indivisible for DHSE in improving the expressive power of GNNs. Without C&S, DHSE can only slightly improve the performance of GATs. However, with C&S, DHSE can significantly improve performance. It implies that DHSE provides GATs the information that neighbor-averaging mechanism cannot provide, and neighbor-averaging mechanism cannot effectively use such information. Moreover, the ablation study shows that encoding layers are key components of DHSEGATs in improving the performance of the base model, and every structure

information makes a similar contribution, while distributional distance information is an indivisible part of DHSE.

Although the power of DHSEGATs has been concretely proven, there are two shortcomings in it. Firstly, it could not significantly improve the performance of the base graph attention model without label propagation. In some cases, it may slightly reduce the performance when independently used. Secondly, the results of the ablation study may imply that some structure statistics are redundant. The effectiveness of every single feature of structure information in improving the based model is worthy of being discussed in detail, which is omitted in this work.

References

[1] LI Q M, HAN Z C, WU X M. Deeper insights into graph convolutional networks for semi-supervised learning. Proc. of the 32nd AAAI Conference on Artificial Intelligence, 2018: 3538−3545.

[2] OONO K, SUZUKI T. Graph neural networks exponentially lose expressive power for node classification. Proc. of the International Conference on Learning Representations, 2020: 1−37.

[3] DEHMAMY N, BARABASI A L, YU R. Understanding the representation power of graph neural networks in learning graph topology. Advances in Neural Information Processing Systems, 2019, 32: 15413–15423.

[4] MONTI F, OTNESS K, BRONSTEIN M M. Motifnet: a motif-based graph convolutional network for directed graphs. Proc. of the IEEE Data Science Workshop, 2018: 225−228.

[5] BOURITSAS G, FRASCA F, ZAFEIRIOU S P, et al. Improving graph neural network expressivity via subgraph isomorphism counting. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2022, 45(1): 657–668.

[6] YOU J X, GOMES-SELMAN J M, YING R, et al. Identity-aware graph neural networks. Proc. of the AAAI Conference on Artificial Intelligence, 2021, 35: 10737–10745.

[7] YING C X, CAI T L, LUO S J, et al. Do transformers really perform badly for graph representation? Advances in Neural Information Processing Systems, 2021, 34: 28877−28888.

[8] YOU J X, YING R, LESKOVEC J. Position-aware graph neural networks. Proc. of the International Conference on Machine Learning, 2019, 97: 7134–7143.

[9] LI P, WANG Y B, WANG H W, et al. Distance encoding: design provably more powerful neural networks for graph representation learning. Advances in Neural Information Processing Systems, 2020, 33: 4465–4478.

[10] ALSENTZER E, FINLAYSON S, LI M, et al. Subgraph neural networks. Advances in Neural Information Processing Systems, 2020, 33: 8017−8029.

[11] DASOULAS G, SANTOS L D, SCAMAN K, et al. Coloring graph neural networks for node disambiguation. Proc. of the 29th International Joint Conference on Artificial Intelligence and the 17th Pacific Rim International Conference on Artificial Intelligence, 2021, 294: 2126−2132.

[12] XU K, HU W H, LESKOVEC J, et al. How powerful are graph neural networks? Proc. of the International Conference on Learning Representations, 2019: 1−17.

[13] ARVIND V, FUHLBRUCK F, KOBLER J, et al. On Weisfeiler-Leman invariance: subgraph counts and related graph properties. Journal of Computer and System Sciences, 2020, 113: 42–59.

[14] CHEN Z D, CHEN L, VILLAR S, et al. Can graph neural networks count substructures? Advances in Neural Information Processing Systems, 2020, 33: 10383−10395.

[15] VIGNAC C, LOUKAS A, FROSSARD P. Building powerful and equivariant graph neural networks with structural message-passing. Advances in Neural Information Processing Systems, 2020, 33: 14143–14155.

[16] HOANG N, MAEHARA T, MURATA T. Revisiting graph neural networks: graph filtering perspective. Proc. of the 25th International Conference on Pattern Recognition, 2021: 8376−8383.

[17] ROSSI E, FRASCA F, CHAMBERLAIN B, et al. Sign: scalable inception graph neural networks. https://arxiv.org/abs/2004.11198v1.

[18] SUN C X, GU H M, HU J. Scalable and adaptive graph neural networks with self-label-enhanced training. https://arxiv.org/abs/2104.09376.

[19] VELICKOVIC P, CUCURULL G, CASANOVA A, et al. Graph attention networks. Proc. of the International Conference on Learning Representations, 2018: 1−12.

[20] YUN S J, JEONG M, KIM R, et al. Graph transformer networks. Advances in Neural Information Processing Systems, 2019, 32: 11983–11993.

[21] HU Z N, DONG Y X, WANG K S, et al. GPT-GNN: generative pre-training of graph neural networks. Proc. of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020: 1857−1867.

[22] HU Z N, DONG Y X, WANG K S, et al. Heterogeneous graph transformer. Proc. of the Web Conference, 2020: 2704−2710.

[23] SUN C X, HU J, GU H M, et al. Adaptive graph diffusion networks. https://arxiv.org/abs/2012.15024.

[24] WANG H W, LESKOVEC J. Unifying graph convolutional neural networks and label propagation. http://arxiv.org/abs/2002.06755.

[25] QU M, BENGIO Y, TANG J. Gmnn: graph Markov neural networks. Proc. of the International Conference on Machine Learning, 2019: 5241−5250.

[26] GAO H C, PEI J, HUANG H. Conditional random field enhanced graph convolutional neural networks. Proc. of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019: 276−284.

[27] ZHENG Q K, LI H Y, ZHANG P, et al. GIPA: general information propagation algorithm for graph learning. https://arxiv.org/abs/2105.06035v2.

[28] GASTEIGER J, BOJCHEVSKI A, GUNNEMANN S. Pre-

dict then propagate: graph neural networks meet personalized PageRank. Proc. of the International Conference on Learning Representations, 2019: 1−15.

[29] HUANG Q, HE H, SINGH A, et al. Combining label propagation and simple models out-performs graph neural networks. Proc. of the International Conference on Learning Representations, 2021: 1−21.

[30] SHI Y S, HUANG Z J, FENG S K, et al. Masked label prediction: unified message passing model for semi-supervised classification. Proc. of the 30th International Joint Conference on Artificial Intelligence, 2021: 1548−1554.

Biographies

HUANG Zhiguo was born in 1989. He received his Ph.D. degree in actuarial science from Nankai University, Tianjin, China, in 2019. Since 2019, he has been a post-doctoral researcher at the Sci-Tech Academy of Zhejiang University and the Post-Doctoral Research Center of Hundsun Incorporated, Hangzhou, Zhejiang, China. He is the author of more than 10 articles. His research interests include graph neural network algorithms, network embedding, and graph neural networks for risk management.

E-mail: hzg0601@163.com