Information Sciences 630 (2023) 222–234

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

High-order graph attention network

Liancheng He [a], Liang Bai [a,b,*], Xian Yang [c], Hangyuan Du [a], Jiye Liang [a,b]

[a] Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, School of Computer and Information Technology, Shanxi University, Taiyuan, 030006, Shanxi, China

[b] Institute of Intelligent Information Processing, Shanxi University, Taiyuan, 030006, Shanxi, China

[c] Alliance Manchester Business School, The University of Manchester, Manchester, UK

ARTICLE INFO

Keywords:
Graph neural network
Graph convolutional network
Attention mechanism
High-order information

ABSTRACT

GCN is a widely-used representation learning method for capturing hidden features in graph data. However, traditional GCNs suffer from the over-smoothing problem, hindering their ability to extract high-order information and obtain robust data representation. To overcome this limitation, we propose a novel graph model, the high-order graph attention network. Compared to other existing graph attention networks, our model can adaptively aggregate node features from multi-hop neighbors through an attention mechanism. Moreover, the edges in the original graph may not accurately represent the relationships between nodes. We implement a new approach to update the graph by using the aggregated node representation to adjust the edges with small step sizes. Additionally, we perform a theoretical analysis to demonstrate the relationships between our proposed model and other GCN models. Finally, we evaluate our proposed model against eight variants of GCN models on multiple widely-used benchmark datasets. The experimental results show the superiority of our proposed model over other models.

1. Introduction

Graph-structured data is prevalent in real-life applications, such as social networks, citation networks, and biological networks [32,13]. Due to its expressive nature, graph-structured data is widely used to represent complex systems in various fields, such as social networks [22], recommendation systems [50,25] and computer vision [26,28,48]. Graph data often includes both topological relationships between nodes and properties of nodes. However, many machine learning algorithms find it challenging to directly handle this type of data. To address this challenge, numerous graph representation learning or graph embedding methods have been developed, such as graph factorization, deepwalk, and graph neural networks, to find node embeddings such that similar nodes in the graph have similar embeddings.

Currently, as an effective tool for graph embedding, graph neural networks have achieved great success in graph machine learning [20,14,34]. Especially, graph convolutional network (GCN) [20] has attracted extensive attention, which plays a fundamental role in the development of follow-up graph neural networks. It is a two-layer neural network that aggregates information from neighbors to learn the representation of nodes. However, the traditional GCN is limited by the over-smoothing problem, leading to a lack of high-order graph information in the learned node representations. It is insufficient to only consider low-order neighbor information when representing a node, as high-order neighbor information contains valuable hidden relationships between nodes that can

* Corresponding author at: Institute of Intelligent Information Processing, Shanxi University, Taiyuan, 030006, Shanxi, China.
E-mail addresses: liancheng.he@foxmail.com (L. He), bailiang@sxu.edu.cn (L. Bai), xian.yang@manchester.ac.uk (X. Yang), duhangyuan@sxu.edu.cn (H. Du), ljy@sxu.edu.cn (J. Liang).

https://doi.org/10.1016/j.ins.2023.02.054
Received 22 May 2022; Received in revised form 10 February 2023; Accepted 12 February 2023

Available online 15 February 2023
0020-0255/© 2023 Elsevier Inc. All rights reserved.

contribute to more accurate node representation. For example, nodes belonging to the same class may not have a direct connection with each other. Thus, it is important to uncover the hidden relationships between them to obtain a better representation.

Furthermore, several graph neural networks have been designed, applying various techniques to enhance the aggregation of neighbor information for node representation. These technologies include sampling nodes or layers [14,4], reconstructing adjacency matrix [11,6], mutual information [35] and random walk [46,23]. However, these methods face challenges in incorporating high-order neighborhood information for node representation. The challenge of incorporating information from higher-order neighborhoods is a key issue that needs to be addressed in the advancement of graph convolutional networks. To address this challenge, several studies have attempted to aggregate node features from multi-hops neighbors through feature propagation and mapping [10,36]. However, they have a common limitation in that they can not dynamically recognize the significance of multi-hop neighbor information for node representation. The use of fixed coefficients for aggregating multi-hop neighbors restricts the expressive power of these models.

In terms of graph topology update, Similarity Metric is a commonly used technique, as seen in methods like spectral clustering [24] and kNN graph [17,16], to construct adjacency matrices. However, relying solely on the original feature information may overlook the diversity among nodes within the same class. For instance, two nodes belonging to the same class may have distinct features but similar neighbors, making it challenging to differentiate them based solely on node features. Thus we consider using the aggregated representation from neighbors to update the graph. Furthermore, we adopt an iteration method to update the graph topology in a more effective way by gradually finding better edge weights with a small step size.

In this work, we present a novel high-order graph attention network (HGRN) that consists of three components: generation of high-order feature tensor through feature propagation, weighting combination of multi-order features with attention mechanism, and updating graph topology. Compared to existing versions of GCN and its variants, the novel HGRN network adaptively integrates different-order feature information to obtain the node representation, thereby avoiding the over-smoothing problem. We provide a theoretical analysis to show the relationship between the proposed network and related deep neural graph networks. Our analysis shows that our model is a general form for aggregating node features, leading to improved node representation. Our experimental results illustrate that HGRN outperforms existing models and achieves state-of-the-art performance.

In summary, the contributions of the proposed paper are as follows:

1) We present a novel high-order graph attention network that effectively integrates multi-hop neighbor information for node representation. Our model incorporates an attention mechanism that adaptively learns the importance of different-order neighbors.
2) Our method can effectively use high-order neighbor information to alleviate the problem of over-smoothing.
3) The proposed HGRN method uses an innovative strategy that updates the graph topology through an iteration method with small step sizes, cumulatively improving the edge weights.
4) Our theoretical analysis demonstrates that the proposed HGRN is a general form of aggregating node features for better representation.

The structure of the rest of this paper is as follows: Section 2 covers related work in the field. Section 3 provides an overview of GCN and GAT, as well as their limitations. The framework of our proposed HGRN model is presented in Section 4. In Section 5, we analyze the relationship of the HGRN with other deep graph neural networks. The experimental results that demonstrate the effectiveness of the proposed HGRN are presented in Section 6. Finally, Section 7 concludes the paper with some final remarks.

2. Related work

For the task of node classification, we introduce two main types of graph convolutional networks: spectral graph convolutional networks and spatial graph convolutional networks. We also discuss the advancement of attention mechanism and graph attention models as follows.

Spectral Graph Convolutional Networks (GCN) models [3,9] define the convolution operation in the spectral domain to uncover the relationship between nodes. In fact, they prefer global information or multi-order information of graphs. These models can be explained by spectral graph theory. However, their drawback lies in the high computational cost during training. GCN [20] limits the layer-wise convolution operation to 1-hop neighborhoods around each node to construct a local spatial relationship of node features. However, this results in the over-smoothing problem as the GCN model can not effectively utilize high-order neighbor information [23]. As a result, Spectral Graph CNN models are not well-suited to address the challenge of incorporating high-order graph information.

Spatial graph approaches [14] define how to aggregate information of neighbor. They are designed to make use of both node features and graph topology to classify nodes or graphs. For most graph models [42,34], they would face the issue that when increasing the depth of layers, local information is lost and the learned features become indistinguishable. Several studies examine adjacency topology to enhance feature aggregation. LDS [11] and IDGL [6] methods merge low-level information and some high-level information by recreating the graph structure. Disentangled models (such as [29,27,45]) attempt to disentangle a graph into multiple factorized graphs as input. Thus, they can extract information from each subgraph to generate a new combination of features. However, they usually have a longer runtime. Models combining graph neural networks with adversarial learning have emerged [41,39], but they are designed to solve the robustness problem. Additionally, incorporating information from high-order neighbors remains challenging for these models. Furthermore, some regularization methods [19,44] have been proposed to examine high-order

information in graphs. For deep GCNs, some works [5,49,12] attempt to alleviate the over-smoothing problem by assigning weights to neighbors. However, the weights for multi-order neighbors are determined by hyperparameter tuning. Recently, an optimization algorithm [15] formulates the asymptotic behaviors of GNTK in the large depth, which shows that wide and deep GCNs experience an exponential drop in trainability in the optimization process.

The attention mechanism in sequential data ([33]) has shown its strong representation capabilities in importance distribution. It has also been applied in graph neural networks to learn the significance of neighbors and aggregate neighbor features. Graph Attention Networks (GAT) [34] utilize self-attention to learn and assign weights to neighbors. GaAN [47] introduces the control of the importance of each attention head. HAN [37] addresses heterogeneous graph problems. SuperGAT ([18]) is an advanced graph attention model for handling noisy graphs. Furthermore, a recent work [2] proposes that there are simple graph problems that GAT can not address due to the static attention mechanism. However, exploring high-order graph relationships remains difficult for these models. To address this limitation, we propose a novel model that leverages both low-order and high-order information in graphs.

3. Preliminary

Let $G = (V, E)$ denote an undirected graph with nodes V and edges E. The nodes are described by the feature matrix $X \in \mathbb{R}^{n \times f}$, where n is the number of nodes, and f is the dimension of their features. $A \in \mathbb{R}^{n \times n}$ is the adjacency matrix, which reflects the connection of the nodes linked by the edges with corresponding degree matrix $D_{ii} = \sum_j A_{ij}$. $\widetilde{A} = A + I$ stands for the adjacency matrix $\widetilde{A}$ for a graph with self-loops and $\widetilde{D} = D + I$ is degree matrix of $\widetilde{A}$. $\hat{A} = \widetilde{D}^{-\frac{1}{2}} \widetilde{A} \widetilde{D}^{-\frac{1}{2}}$ denotes the symmetric normalized adjacency matrix with self-loops.

3.1. Graph convolutional networks (GCN)

The vanilla GCN is proposed by Kipf & Welling [20]. It aggregates information of neighbors by symmetric normalized adjacency matrix $\widetilde{A}$ to generate new node representation. The GCN model can be described by

$$H = Softmax\left(\hat{A} ReLU\left(\hat{A} X W^{(0)}\right) W^{(1)}\right), \tag{1}$$

where $W^{(0)}$ and $W^{(1)}$ denote the trainable parameter matrices of the first and second layers of GCN, respectively. H is the final prediction of the GCN model. Each GCN layer can aggregate the features of the 1-hop neighborhoods around nodes. However, different weights need to be considered for different nodes in a neighborhood.

3.2. Graph attention networks (GAT)

In contrast to GCN, GAT ([34]) uses an attention mechanism to calculate edge weights. GAT updates both the adjacency matrix and features by aggregating feature information from neighbors with node-specific weights. The weights of edges can be calculated by the following formula

$$\alpha_{ij} = \frac{exp\left(LeakyReLU\left(\vec{a}^T [W\vec{h}_i \| W\vec{h}_j]\right)\right)}{\sum_{k \in \mathcal{N}_i} exp\left(LeakyReLU\left(\vec{a}^T [W\vec{h}_i \| W\vec{h}_j]\right)\right)}, \tag{2}$$

where α_{ij} is the weight of edge between the ith and jth nodes. W is a trainable weight matrix that transforms previous node features. $\vec{h}_i$ and $\vec{h}_j$ denote the feature for the ith and jth nodes. $\vec{a}$ is a weight vector that measures and compares node features. $\mathcal{N}_i$ is the direct neighborhood of the ith node in the graph. The feature information is processed by combining with the weighted neighbor information from the attention adjacency matrix and then transformed through a nonlinear function, which is described by

$$\vec{h}'_i = \sigma\left(\frac{1}{K} \sum_{k=1}^{K} \sum_{j \in \mathcal{N}_i} \alpha_{ij}^k W^k \vec{h}_j\right), \tag{3}$$

where K is the number of multi-heads. σ denotes the final nonlinear function (usually a softmax). Finally, make a prediction using the output of the final layer.

GCN and GAT have achieved significant success in classification tasks. However, they are limited to only considering local neighbors for each node and are unable to capture higher-order relationships among neighborhoods. For instance, a two-layer graph model can only consider the information of two-hop neighbors. To address this limitation and explore higher-order relationships, we introduce a novel model that aggregates information from nodes at various distances in the neighborhood.

4. High-order graph attention network

4.1. Algorithm framework

In this study, we introduce a High-Order Graph Attention Network (HGRN) that utilizes high-order graph information to enhance node representation generation. The framework of HGRN is shown in Fig. 1. It is made up of three key modules as follows.

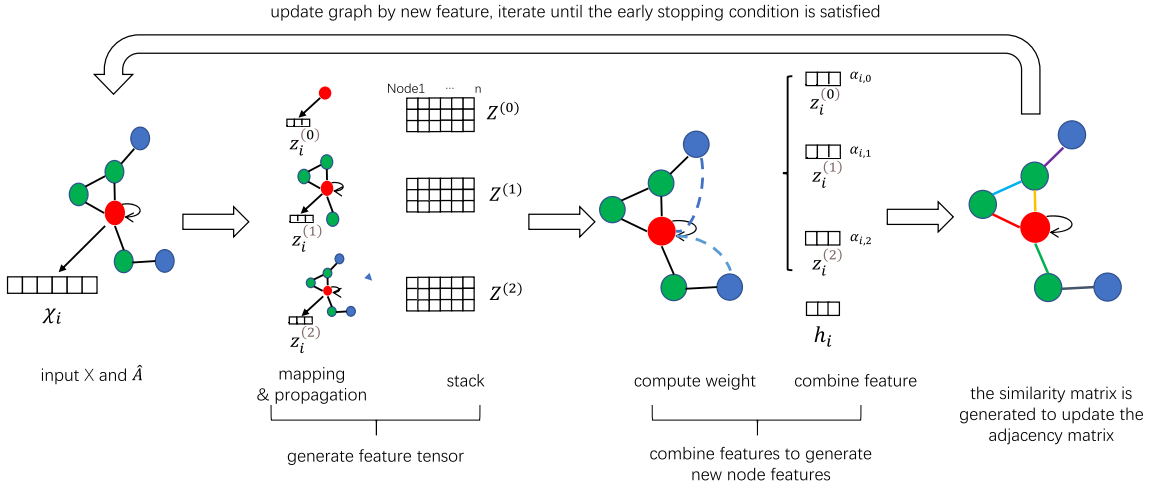

Fig. 1. Illustration diagram of HGRN.

1) *Generation of feature tensor from multi-order neighbors*. In this module, we employ feature mapping and propagation to obtain a feature tensor that captures the node features from multi-order neighbors (including the node itself).
2) *Aggregation of the features from different-order neighbors*. In this module, new node features are generated by aggregating the feature matrices from different-order neighbors, weighted by attention coefficients.
3) *Updating graph topology*. In this module, the graph topology is refined by incorporating the newly obtained features, improving the representation of node relationships. This module enables iterative updates of the graph edge weights.

In the subsequent sections, we will provide a detailed explanation of each module's implementation.

4.2. Generation of feature tensor from multi-order neighbors

In the first module, the goal is to generate node representations from various-order neighbors and store them in a feature tensor. This requires two key steps, feature mapping and feature propagation.

Since the original node feature matrix X in a graph is often sparse, it can not be used directly to compute features from multi-order neighbors. Therefore, we first map the node feature matrix X to a label feature space that reflects the distribution of each node's label among classes. In this paper, the feature mappings are realized using a two-layer fully-connected neural network, which is described as follows:

$$f_\theta(X) = ReLU(XW^{(1)})W^{(2)}. \tag{4}$$

Following the feature mappings, we compute new representations for each node based on its different-order neighbors using a feature propagation mechanism on the normalized adjacency matrix $\hat{A}$ of the graph. The process of generating the feature tensor T is described as follows:

$$Z^{(0)} = f_\theta(X), \tag{5}$$

$$Z^{(l+1)} = \hat{A}Z^{(l)}, \tag{6}$$

$$T_{[:,j,:]} = Z^{(j)}, T \in \mathbb{R}^{n\times(L+1)\times c}, \tag{7}$$

where $Z^{(0)} \in \mathbb{R}^{n\times c}$ is the feature matrix which X is mapped to, c is the number of classes, L is the number of orders. Note that the length of the vector obtained by feature mapping is equal to the number of classes. Therefore, $Z^{(0)}$ also can be seen as the original label features of each node. $Z^{(l)}$ is the label feature matrix for each node from lth-order neighbors by the lth feature propagation. The label feature matrix of each order is stacked to generate a feature tensor T. To improve the generalization ability of neighborhood expansion, we adopt the DropEdge [31] strategy to randomly dropout edges of the graph to get different $\hat{A}$ in each propagation.

4.3. Aggregation of the features from different-order neighbors

After obtaining the feature tensor, we aggregate the label features from multi-order neighbors to derive a representation for each node. In the second module of the proposed framework, we use a feedforward neural network with an attention mechanism to perform the aggregation task. This module aggregates the label features of neighbors from different orders, with adaptive weights computed through the attention mechanism, which learns coefficients to weight the feature information of neighbors from each hop. It is important to note that the proposed module differs from GAT, as illustrated in Fig. 2. GAT only assigns weights to itself and

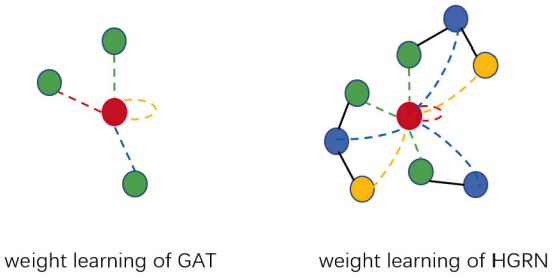

Fig. 2. The difference between GAT and HGRN in learning weight.

its immediate neighbors, while our proposed method learns weights for neighbors of various orders. This allows for learning more high-order information to represent nodes as the number of layers increases.

The attention coefficient can be calculated as

$$m_{i,l} = \frac{exp\left(\vec{u}_i \vec{t}_{i,l}\right)}{\sum_{k=0}^{L} exp\left(\vec{u}_i \vec{t}_{i,k}\right)}, \tag{8}$$

where $U \in \mathbb{R}^{n \times c}$ is the parameter matrix of the attention network and $\vec{u}_i$ is the ith row in U for the ith node. $m_{i,l} \in \mathbb{R}^{L+1}$ denotes the weight for the lth-order neighborhood of the ith node with self-loop. $\vec{t}_{i,l} = T_{[i,l,:]}$ is a vector with c elements in T, which reflects the lth-order feature of ith node.

Next, we need to aggregate the features of different-order neighbors with the obtained weights to get the new representation of nodes. Let $H \in \mathbb{R}^{n \times c}$ be the representation matrix of nodes after aggregating the features of L-order neighbors in the graph, and $\vec{h}_i$ is the ith row of H which represents the feature of the ith node. $\vec{h}_i$ is computed by the following equation

$$\vec{h}_i = \sigma(\sum_{l=0}^{L} m_{i,l} \vec{t}_{i,l}), \tag{9}$$

where σ is an activation function. In the proposed framework, if we do not update graph topology, σ is the final nonlinear function (softmax). Otherwise, σ is a direct output.

4.4. Updating graph topology

We know that the quality of node representation produced by graph neural networks is influenced by the graph topology. The original edges in the graph may not fully reflect the relationships between nodes. Updating the graph topology during network training can sometimes improve the learning of node representation. We assume that the learned node feature may contain highly reliable label information. Thus, if a node is particularly similar to one of its neighbors, the weights of the natural edge need to be increased to highlight its importance. In this module, we use cosine similarity between the learned node features to enhance the graph topology as follows:

$$\hat{A}_{i,j} = (1-\lambda)\hat{A}_{i,j} + \lambda S_{i,j}, \tag{10}$$

where

$$S_{i,j} = cos\left(\vec{h}_i, \vec{h}_j\right). \tag{11}$$

$S_{i,j}$ denotes the similarity score between the ith and jth nodes and λ is a parameter to control the role of updating graph topology.

4.5. Description of algorithm

The overall process of the HGRN framework is summarized in Algorithm 1. If $\lambda = 0$, this means that the graph topology is not updated. In this paper, we call the HGRN with $\lambda = 0$ as SHGRN. Its storage cost is $\mathcal{O}(nf + |E| + Lnc)$, where $O(nf)$ is the size of the original feature matrix X, $|E|$ is the number of edges, and $O(Lnc)$ is used to save the feature tensor including L-order matrices after feature propagation. If $\lambda = 0$, the computational cost is $\mathcal{O}(nfc + L|E|c + Lnc)$, where $O(nfc)$ is the cost of feature mapping, $O(L|E|c)$ is the cost of feature propagation and $O(Lnc)$ is used for feature aggregation from tensor to the matrix. If $\lambda > 0$, there is an additional step for updating the graph topology. In this case, the time complexity is $\mathcal{O}\left(nfc + L|E|c + Lnc + n^2c\right)$.

Algorithm 1: The framework of HGRN.

Input: Feature matrix X, adjacent matrix A, number of classes c, neighborhood range (number of layers) L, parameter λ
Output: Final label feature matrix H

Map X to Z by $Z^{(0)} = f_\theta(X)$
while *The function loss is not satisfied* **do**
  for $l = 0$ *to* L **do**
    Get the node features of the lth order by $Z^{(l+1)} = \hat{A}Z^{(l)}$
  Generate feature tensor $T = [Z^{(0)}, \cdot, Z^{(L)}]$
  Compute attention coefficient by $m_{i,l} = \frac{exp(\vec{u}_i \vec{t}_{i,l})}{\sum_{k \in \mathcal{N}_i} exp(\vec{u}_i \vec{t}_{i,k})}$
  Get the node representation by $\vec{h}_i = \sigma(\sum_{l=0}^{L} m_{i,l} \vec{t}_{i,l})$
  if $\lambda > 0$ **then**
    Compute the similarity between nodes $S_{i,j} = cos\left(\vec{h}_i, \vec{h}_j\right)$
    Update the graph topology by $\hat{A}_{i,j} = (1-\lambda)\hat{A}_{i,j} + \lambda S_{i,j}$
Return H

5. Theoretical analysis

To explore high-order relationships on a graph for node representation and classification, some spatial methods utilize knowledge from multi-hop neighbors. In practice, these methods can be considered as specific instances of our model under certain circumstances. We examine the relationship between our model and these methods in detail below.

Relation with SGC. SGC simply collects information from K-hop neighbors in a graph by using the K-th power of the normalized adjacency matrix in a single-layer neural network, but it fails to consider the significance of information from different-hop neighbors. It is equivalent to a part of SHGRN if SHGRN only performs K times in Eq. (5) and Eq. (6). The formula of SGC can be expressed as

$$Z^{(L)} = softmax\left(\hat{A}^l X W\right). \tag{12}$$

Relation with APPNP and GPRGNN. APPNP applies a Personalized PageRank matrix for feature propagation to relieve the over-smoothing problem, but the fixed hyperparameters are not universal for all datasets. Similarly, this strategy can find traces in lazy random walk (with restart) and label propagation. Recently, a novel model called GPRGNN can learn the weight of different layers to distribute the proportion from different-hop neighbors. However, sharing a parameter in the same propagation layer will lack discrimination for each node. From the perspective of generality, we can prove that GPRGNN is a general form of APPNP as 5.1 and GPRGNN is a special case of our model SHGRN under certain conditions as described in 5.2. Through comparison and analysis, we demonstrate that our model SHGRN which operates in the node-wise neighborhood is a more general form to describe neighbor relationships.

Theorem 5.1. *GPRGNN is a general form of APPNP to adjust the weights of layers after propagation adaptively.*

Proof. Define $Z^{(0)}$ as the output after feature mapping and set each model to propagate L times for comparison. In APPNP and GPRGNN, the coefficient of layers after feature propagation are different as follows:

$$Z^{(L)}_{APPNP} = softmax\left((1-\alpha)\hat{A}Z^{(L-1)} + \alpha Z^{(0)}\right), \tag{13}$$

$$Z^{(L)}_{GPRGNN} = softmax\left(\sum_{l=0}^{L} \gamma_l Z^{(l)}\right). \tag{14}$$

We can find APPNP is a special case of GPRGNN when $\gamma_l = \alpha(1-\alpha)^l$ and $\gamma_L = (1-\alpha)^L$. □

Theorem 5.2. *SHGRN is the general version of GPRGNN to adjust neighbor weights for each node adaptively.*

Proof. First of all, we need to note that GPRGNN is suitable for layer-level feature processing, whereas SHGRN can process feature information according to each node. The formula of GPRGNN can be described as follows:

$$Z = softmax(\sum_{l=0}^{L} \gamma_l Z^{(l)}). \tag{15}$$

We can find that for each node, GPRGNN can learn the same parameters γ_l. So, if the output of node i need to be obtained, the formula can be written as:

$$\vec{z}_i = \sigma(\sum_{l=0}^{L} \gamma_l \vec{z}_i^{(l)}). \tag{16}$$

SHGRN can learn the weights of neighbors for each node as follows:

$$\vec{h}_i = \sigma(\sum_{l=0}^{L} m_{i,l}\vec{t}_{i,l}). \tag{17}$$

We can find that γ_l is fixed in each propagation layer, but $m_{i,l}$ is different for each node. Thus, we can conclude that GPRGNN is a special form of SHGRN for whole nodes in a layer. The only difference is that the weights of GPRGNN can be negative, whereas the weights of SHGRN are between 0 and 1 after normalization. □

Relation with spectral methods. We also discuss some spectral methods using multi-order neighbors for aggregation as follows. Consider the normalized graph Laplacian matrix $L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ with eigendecomposition $U\Lambda U^T$, where Λ is a diagonal matrix of the eigenvalues of L and $U \in \mathbb{R}^{n\times n}$ is a matrix that consists of the eigenvectors of L. A signal x is filtered by $g_\theta(L) = diag(\theta)$ as $g_\theta(\Lambda) * x = U g_\theta(\Lambda)U^T x$, where the parameter $\theta \in \mathbb{R}^n$ is a vector of Fourier coefficients.

[9] proposes polynomial parametrization for localized filters as

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k, \tag{18}$$

where the parameter $\theta \in \mathbb{R}^K$ is a vector of polynomial coefficients. The filter represented by K-order polynomials of the Laplacian includes K-localized neighborhood information.

MGCN [38] proposes a new method with a multi-scale wavelet filter, described by

$$g_\theta(\Lambda) = \sum_{k=0}^{K} \theta_k g_{t_k}(\Lambda), \tag{19}$$

where

$$g_{t_k}(\lambda_j) = \begin{cases} h(\lambda_j), & \text{if } m = 0 \\ g(t_m\lambda_j), & \text{if } m > 0 \end{cases} \tag{20}$$

Specifically, the Mexican hat functions are chosen as the filters of the graph wavelet, which are given by

$$g(t_m\lambda_j) = A(t_m\lambda_j)^2 e^{\left(1-(t_m\lambda_j)^2\right)},\ h(\lambda_j) = Be^{\left(-\left(\frac{C\lambda_j}{\lambda_{max}}\right)^3\right)}, \tag{21}$$

where $t_m = e^{linspace\left(log\left(\frac{D}{\lambda_{max}}\right),log\left(\frac{E}{\lambda_{max}},K\right)\right)}$. A, B, C, D and E are parameters. MGCN uses the graph wavelet to adjust the search range and the value of the filter. The two spectral methods explore graph higher-order information from the perspective of filters. Unlike the two methods, SHGRN is a spatial method for multi-order neighborhood aggregation. Furthermore, HGRN can update graph topology iteratively based on SHGRN to explore the relationship between nodes. It is a general form to exploit high-order graph relationship.

6. Experiment

6.1. Experimental setup

Datasets. We conduct experiments on four homophilic benchmark graphs: CiteSeer, Cora-ML, PubMed and MS Academic graph. Furthermore, we also do experiments on four heterophilic benchmark datasets from Geom-GCN [30]: Chameleon and Squirrel from Wikipedia network, and webpage dataset WebKB including Cornell and Texas. For homophilic datasets, we follow the same experimental procedure as PPNP [21]. Specifically, each experiment is run 100 times on 20 random splits and initializations. Additionally, the largest connected component of each graph is utilized and the abstracts of the papers are represented as features using a bag-of-words representation. For heterophilic datasets, nodes of each class are randomly split into training, validation, and testing sets at a ratio of 0.48:0.32:0.2. It should be noted that the datasets are initially divided into training and testing sets at a ratio of 0.8:0.2. Then, the training set is further divided into training and validation sets at the ratio of 0.6:0.4. The homophily of datasets is defined as the average ratio of direct neighbors of a node that have the same label as the center node [30]. More intuitively, we can express homophily as $\mathcal{H} = \frac{1}{|V|}\sum_{i\in V}\sum_{j\in\mathcal{N}_i} 1_{l_i=l_j}/|\mathcal{N}_i|$, where $\mathcal{N}_i$ denotes the direct neighbor of node i (have edges), l_i is the label of node i and V denotes all node collections.

Our settings of homophilic datasets are the same as PPNP [21] for fairness purposes. For data splits, we use visible settings in which 1500 nodes are sampled for the citation graphs and 5000 nodes are sampled for MS Academic. It should be noted that the visible set includes both the training and validation sets. The remaining nodes form the test set. Furthermore, the training set consists of 20 nodes per class and the validation set consists of 500 nodes. For fair comparison, the same early stopping mechanism is adopted across all methods. For heterophilic datasets, the handling of node features and the adjacency matrix is done in the same manner as Geom-GCN [30]. The Adam optimizer and cross-entropy loss are used for all datasets, and both models are initialized using Glorot Initialization. The details of dataset statistics are summarized in Table 1.

Baselines. We compare HGRN with eight models: GCN [20], GAT [34], SGC [40], MixHop [1], PPNP (& APPNP) [21], JKNet [43] and GPRGNN [7]. For HGRN(SHGRN), we use a variant of two-layer MLP without bias and set L_2 regularization to 5×10^{-3} on the first layer for feature mapping. Then the hidden representation is transformed using the ReLU nonlinear function. The second

Table 1
Dataset statistics.

Dataset	Classes	Features	Nodes	Edges	Label rate	Homophily
CiteSeer	6	3703	2110	3668	0.057	0.71
Cora-ML	7	2879	2810	7981	0.050	0.80
PubMed	3	500	19717	44324	0.003	0.79
MS Academic	15	6805	18333	81894	0.016	0.83
Texas	5	1703	183	309	0.475	0.06
Cornell	5	1703	183	295	0.475	0.11
Chameleon	4	2325	2277	36101	0.480	0.25
Squirrel	4	2089	5201	217073	0.480	0.22

layer maps directly to the label feature. After conducting a line search on hyper-parameters, the number of propagation layers is set to 11, 15, 10, and 5 respectively for the different graph structures in homophilic datasets. For all heterophilic datasets, 1 propagation layer is set. In addition, the dropout rate of edges is set to 0.5 for all datasets except HGRN in CiteSeer 0.4. For the feature, we set the dropout rate to 0 for Citeseer, Cora-ML datasets and PubMed, and 0.5 for Ms-academic. For different layers, the dropout rate is set to 0.2 for Citeseer and Cora-ML datasets, 0.4 for PubMed, and 0 for Ms-academic. For all heterophilic datasets, the dropout rate of features is set to 0.5 and the dropout rate of layers is set to 0.2. Regarding the learning rate, we set 0.01 for all datasets. For HGRN, the hyperparameter $\lambda = 0.004$ on Cora-ML and PubMed, $\lambda = 0.005$ on CiteSeer and $\lambda = 0.001$ on Ms-Academic. We set the patience $p = 150$ and the maximum of $n = 10000$ epochs for homophilic datasets due to the slow convergence of the model. In addition, we set the patience $p = 100$ and the maximum of $n = 2000$ epochs for all heterophilic datasets on all models.

To ensure a fair comparison of performance among models, the number of hidden units in all models is set to 64. The early stopping criterion on homophilic datasets uses the patience of p = 100 and the maximum of n = 10000 epochs. The dropout rate is set to $d = 0.5$ except GAT $d = 0.6$. We apply learning rate $lr = 0.01$ except GAT $lr = 0.005$. For L_2 regularization, GCN, GAT, JKNet and MixHop use $\lambda = 5 \times 10^{-4}$, and PPNP, APPNP and GPRGNN apply $\lambda = 5 \times 10^{-3}$ on first linear layer. In detail, for GAT, the first layer consists of $K = 8$ attention heads computing $F' = 8$ features each (for a total of 64 features), followed by an exponential linear unit (ELU) [8] nonlinearity. For SGC, we use one layer to transform features and then propagate 5 times to get final results with L_2 regularization $\lambda = 0$. For JKNet, we set the concatenation variant with three layers as the original paper. MixHop uses two layers which can concatenate three hops for each layer to combine features of different orders. PPNP and APPNP follow the original paper completely. For feature propagation, the dropout rate is $d = 0.5$ for the adjacency matrix. And we set teleport probability $\alpha = 0.1$ on Cora-ML, CiteSeer and PubMed, $\alpha = 0.2$ on Ms-academic. For APPNP, we set the number of power iteration steps to $K = 10$. GPRGNN applies the same parameters as APPNP except for teleport probability. We use initializations with $\alpha = 0.1$ for all datasets.

6.2. Experimental results

Prediction performance. For homophilic datasets, we compare the models with eight methods. The results show that HGRN and SHGRN consistently outperform the compared methods in terms of accuracy. Different models have diverse feature combinations, resulting in varying prediction capabilities. Our goal is to learn better feature representation. The experimental results are summarized in Table 2. All experiments are carried out according to the algorithm of the original paper. The highest prediction accuracy for each dataset in the table has been bold. The results show that our model surpasses the state-of-the-art baseline models. There is an interesting phenomenon that GCN outperforms GAT in several datasets. It indicates that the edge weights learning of GAT may be not optimal and can sometimes lead to performance degradation. Models like APPNP and GPRGNN that can utilize high-order information tend to have superior classification ability compared to shallow models. The results also demonstrate that selecting a suitable measurement method for weight distribution calculation can enhance representation learning. Our method outperforms other depth graph neural networks, demonstrating its ability to effectively learn better neighbor weights for neighborhood aggregation.

For heterophilic datasets, we compare SHGRN with GCN, GAT, SGC, JKNet, APPNP and GPRGNN under 100 random splittings of datasets. Due to the low homophily, the performance of all models on heterophilic datasets is usually not good enough. The results in Table 3 show that GPRGNN and our method SHGRN attain state-of-the-art performance on heterophilic datasets. The performance of SHGRN and GPRGNN is comparable on all datasets. The experimental results indicate that GPRGNN is well-suited for node classification in heterophilic datasets, but not as much in homophilic datasets. Our model SHGRN also outperforms other graph neural networks of the same type, such as SGC and APPNP. Therefore, it can be concluded that our weight learning method is effective. In the experimental results, bold text indicates the best results, while underlined bold text represents results within the confidence interval of the top result. The results reveal that GAT struggles to learn appropriate edge weights in heterophilic datasets, resulting in the lowest performance among all models. As the optimization of graph topology update by HGRN relies on the accuracy of SHGRN, low accuracy would cause the graph update to move in the incorrect direction. Hence, HGRN was not considered a comparison model.

t-SNE visualization. To demonstrate the improvement of graph node representation through multi-order neighborhood aggregation, we use t-SNE to visualize the results of MLP, GCN, and our algorithm on the Cora-ML dataset in Fig. 3. It is important to note that the 2-layer MLP variant is generated from Eq. (4). In the MLP visualization (Fig. 3a), most node representations are distinct.

Table 2
The accuracy results for the homophilic datasets are calculated by bootstrapping with a 95% confidence level based on 100 runs. Note that some models were unable to run on certain datasets due to memory constraints, as indicated by "-". Under strict experimental conditions, both HGRN and SHGRN demonstrate better performance than other compared models across all datasets.

	CiteSeer	Cora-ML	PubMed	MS Academic
GCN	74.20 ± 0.47%	83.19 ± 0.27%	77.95 ± 0.38%	92.09 ± 0.08%
GAT	75.07 ± 0.45%	83.90 ± 0.25%	77.79 ± 0.65%	91.55 ± 0.13%
JKNet	72.13 ± 0.44%	81.88 ± 0.32%	77.38 ± 0.40%	89.81 ± 0.17%
MixHop	73.87 ± 0.42%	82.24 ± 0.30%	77.35 ± 0.41%	92.36 ± 0.09%
SGC	72.11 ± 0.39%	81.37 ± 0.28%	76.65 ± 0.36%	89.37 ± 0.13%
PPNP	75.74 ± 0.29%	85.15 ± 0.25%	-	-
APPNP	75.82 ± 0.26%	85.01 ± 0.24%	79.66 ± 0.34%	93.17 ± 0.07%
GPRGNN	75.45 ± 0.33%	84.86 ± 0.32%	79.27 ± 0.35%	92.63 ± 0.08%
SHGRN	76.13 ± 0.26%	85.68 ± 0.22%	80.39 ± 0.27%	**93.35 ± 0.06%**
HGRN	**76.25 ± 0.28%**	**85.79 ± 0.21%**	**80.42 ± 0.28%**	93.31 ± 0.06%

Table 3
The average accuracy of heterophilic datasets with uncertainties shows the 95% confidence level calculated by bootstrapping under 100 runs. Under a rigorous experimental setup, SHGRN matches or outperforms the compared models on all datasets.

	Texas	Cornell	Chameleon	Squirrel
GCN	59.22 ± 1.51%	62.27 ± 1.54%	36.23 ± 0.53%	26.22 ± 0.27%
SGC	54.38 ± 1.54%	56.62 ± 1.73%	36.20 ± 0.94%	25.03 ± 0.28%
GAT	54.41 ± 1.99%	55.11 ± 1.62%	27.88 ± 0.84%	23.89 ± 0.48%
JKNet	58.03 ± 1.35%	62.49 ± 1.46%	34.51 ± 0.59%	26.41 ± 0.50%
APPNP	59.43 ± 1.49%	57.59 ± 1.57%	37.28 ± 0.53%	27.05 ± 0.32%
GPRGNN	**77.03 ± 1.70%**	**79.92 ± 1.38%**	**44.23 ± 0.52%**	**29.95 ± 0.29%**
SHGRN	**77.84 ± 1.81%**	**79.49 ± 1.27%**	**44.46 ± 0.48%**	**29.70 ± 0.31%**

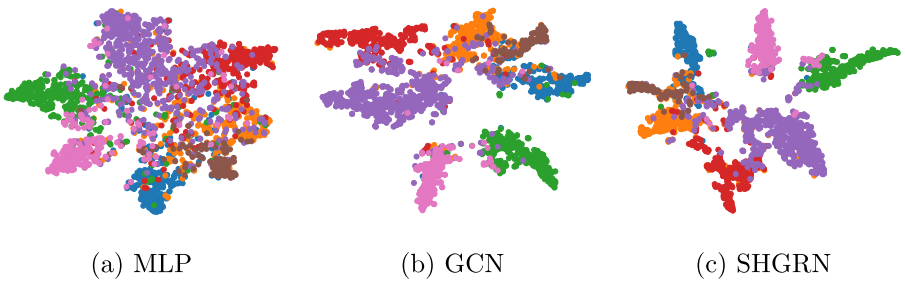

Fig. 3. The t-SNE visualization of MLP, GCN and SHGRN outputs on the Cora-ML dataset.

However, the model output is spread out and the class boundaries are not clearly defined. In the GCN visualization (Fig. 3b), the output is more compact compared to MLP and the model performance has greatly improved. In the SHGRN visualization (Fig. 3c), the model has reduced misclassification compared to GCN, resulting in closer proximity of node features of the same class. The results demonstrate that GCNs can be applied to a variety of graph-structured data, and the effective utilization of high-order information in graphs can improve node representation even further.

The impact of neighborhood range. It is known that as the depth of GCN and GAT increases, they may experience over-smoothing, resulting in a decline in performance. Thus, they are unable to gather information from a wider neighborhood range, leading to insufficient representation capacity. As demonstrated in Fig. 4, our model has been proven through experiments to maintain high accuracy even with an increasing number of propagation layers, effectively avoiding over-smoothing. As the number of layers increases, the prediction accuracy of SGC remains stable but significantly lower compared to our model. In addition, we find an interesting phenomenon where GCN and GAT models exhibit the steepest decline in performance when the number of layers falls within the range of 5 to 7. Furthermore, we found several unique features of our model. Firstly, as the number of layers increases, the model's performance will increase until reaching a peak, then slightly decrease and generally stabilize. This is a common trend among models, which typically have a specific number of layers (parameters) that result in optimal performance. Additionally, we introduce a novel idea that incorporating dropout in the propagation layers (new tensor) can enhance performance and robustness.

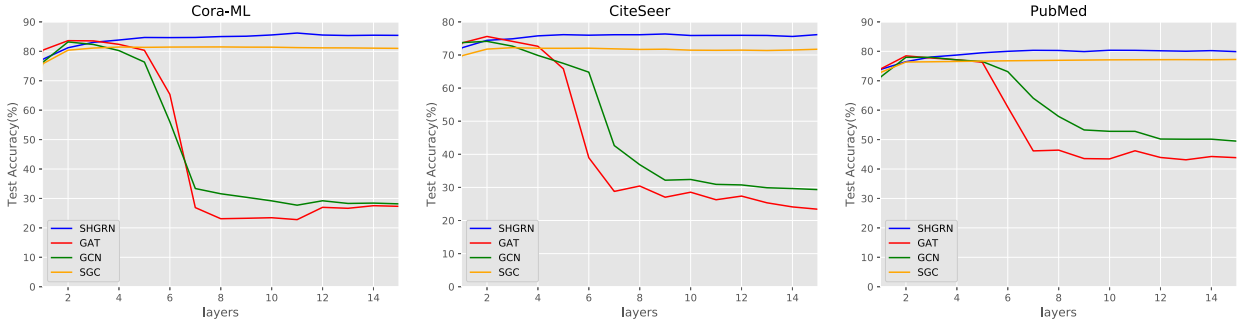

Fig. 4. Comparison of the test accuracy of various models as the number of propagation layers increases. The comparison involves three datasets (Cora-ML, CiteSeer, and PubMed) and four models (GCN, GAT, SGC, and SHGRN). In SGC and SHGRN, the layers represent the feature matrix after feature propagation, allowing for the exploration of neighbors from various distances.

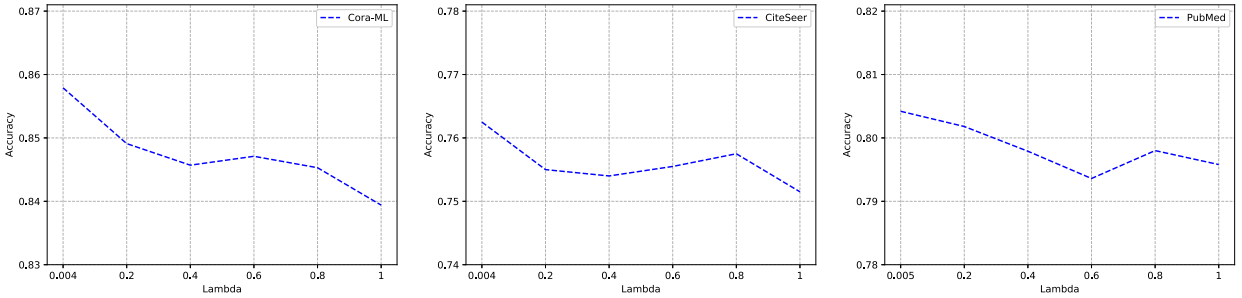

Fig. 5. Sensitivity analysis of λ on three datasets (Cora-ML, CiteSeer and PubMed).

Table 4
The relationship between WLNP and the label rate in each dataset.

Dataset	WLNP	Label rate
CiteSeer	12	3.41%
	16	4.55%
Cora-ML	12	2.99%
	16	3.99%
PubMed	12	0.18%
	16	0.24%
MS Academic	12	0.98%
	16	1.31%

In heterophilic datasets, we discovered that SHGRN achieves optimal performance when the propagation layer is set to one. In other words, using only the node's own features and those of its direct neighbors provides the highest accuracy. This is due to the fact that increasing the number of layers leads to the introduction of numerous noisy features through propagation, which negatively impacts the original features.

The impact of label rates. To examine the effects of models with varying labeling rates, particularly lower labeling rates, we have established a performance comparison of each model under different labeling rates. The experiment of the above standard is to set 20 labeled nodes per class in homophilic datasets. For convenience, we refer to the number of labeled nodes per class as WLNP. In Table 5, we also tested with 12 and 16 labeled nodes per class (the highest prediction in each dataset is bolded). The relationship between WLNP and the label rate in each dataset is depicted in Table 4. The experimental results indicate that PPNP, GPRGNN, and our model perform well even when the labeling rate decreases. We can conclude that models with multiple propagations are beneficial for feature representation learning, even when the labeling rate is low. Our model adaptively learns high-order graph information to achieve better feature representation for each dataset. Despite a decrease in the labeling rate, our model still has superior prediction accuracy compared to other models. In contrast, the simple SGC model performs poorly under different labeling rates due to its inability to learn the importance of the neighborhood. This highlights the role played by the neighbor importance distribution used by PPNP (APPNP), GPRGNN, and HGRN (SHGRN) in feature combination and representation.

Sensitivity analysis. To examine the impact of the HGRN hyperparameter λ on our experiments, we conduct a sensitivity analysis on three datasets: Cora, CiteSeer, and PubMed. As seen in Fig. 5, the experiment results show that the accuracy peaks at

Table 5
The average accuracy, along with its uncertainty (represented by 95% confidence level), was calculated through 100 runs using the bootstrapping method. Each class has 12 to 16 labeled labels. For convenience, we refer to the number of labeled nodes per class as WLNP.

	WLNP	CiteSeer	Cora-ML	PubMed	MS Academic
GCN	12	72.71 ± 0.38	81.49 ± 0.30	76.58 ± 0.53	91.55 ± 0.10
	16	73.40 ± 0.47	82.17 ± 0.28	77.43 ± 0.45	91.92 ± 0.09
GAT	12	73.91 ± 0.32	82.63 ± 0.34	75.09 ± 0.84	91.06 ± 0.26
	16	74.60 ± 0.42	83.25 ± 0.37	77.00 ± 0.85	91.34 ± 0.15
JKNet	12	71.24 ± 0.42	80.55 ± 0.33	75.34 ± 0.63	89.27 ± 0.27
	16	71.57 ± 0.47	81.26 ± 0.31	77.18 ± 0.42	89.54 ± 0.21
MixHop	12	72.12 ± 0.42	80.17 ± 0.34	74.67 ± 0.48	91.36 ± 0.13
	16	73.34 ± 0.39	81.08 ± 0.32	76.60 ± 0.52	91.96 ± 0.09
SGC	12	70.89 ± 0.38	79.82 ± 0.25	74.25 ± 0.45	88.85 ± 0.16
	16	71.50 ± 0.35	80.49 ± 0.26	75.86 ± 0.49	89.28 ± 0.15
PPNP	12	74.85 ± 0.32	84.06 ± 0.29	-	-
	16	75.18 ± 0.36	84.42 ± 0.29	-	-
APPNP	12	75.13 ± 0.30	83.64 ± 0.31	77.53 ± 0.42	92.10 ± 0.11
	16	75.36 ± 0.30	84.30 ± 0.28	78.99 ± 0.43	92.41 ± 0.07
GPRGNN	12	74.28 ± 0.31	83.48 ± 0.29	77.69 ± 0.45	92.14 ± 0.11
	16	74.77 ± 0.34	84.23 ± 0.29	78.80 ± 0.42	92.43 ± 0.10
SHGRN	12	75.28 ± 0.24	83.95 ± 0.33	**78.40 ± 0.42**	92.59 ± 0.10
	16	75.71 ± 0.28	84.97 ± 0.25	79.64 ± 0.36	**93.10 ± 0.07**
HGRN	12	**75.49 ± 0.23**	**84.12 ± 0.30**	78.35 ± 0.42	**92.62 ± 0.08**
	16	**75.81 ± 0.27**	**84.98 ± 0.26**	**79.76 ± 0.40**	93.09 ± 0.07

Table 6
The average training time per epoch. SHGRN is comparable to APPNP and GPRGNN and faster than the advanced attention network GAT.

	CiteSeer	Cora-ML	PubMed	MS Academic
GCN	11.56 ± 0.20 ms	13.73 ± 0.18 ms	24.47 ± 0.14 ms	35.90 ± 0.15 ms
GAT	134.99 ± 1.53 m	205.78 ± 1.73 ms	1009.51 ± 3.79 ms	1510.90 ± 7.26 ms
JKNet	43.89 ± 0.62 ms	51.14 ± 0.62 ms	97.72 ± 1.39 ms	634.57 ± 2.25 ms
MixHop	17.33 ± 0.30 ms	22.95 ± 0.33 ms	68.47 ± 0.93 ms	85.15 ± 0.26 ms
SGC	10.26 ± 0.18 ms	11.09 ± 0.09 ms	10.43 ± 0.13 ms	23.95 ± 0.09 ms
PPNP	13.63 ± 0.27 ms	16.67 ± 0.16 ms	-	-
APPNP	13.59 ± 0.23 ms	15.25 ± 0.20 ms	25.44 ± 0.22 ms	41.57 ± 0.15 ms
GPRGNN	16.92 ± 0.28 ms	17.54 ± 0.80 ms	29.53 ± 0.83 ms	43.94 ± 0.38 ms
SHGRN	14.05 ± 0.25 ms	16.87 ± 0.25 ms	28.07 ± 0.19 ms	40.60 ± 0.47 ms
HGRN	14.25 ± 0.22 ms	17.15 ± 0.21 ms	28.18 ± 0.22 ms	43.40 ± 0.15 ms

smaller values of λ and tends to decrease as λ increases. This indicates that the optimal edge weights may be around the graph Laplace matrix so changing the edge weights by fine-tuning is a feasible way to explore the neighborhood for aggregation. To further improve performance, future work will focus on adjusting edges by adding or removing edges, specifically removing inter-class edges and increasing intra-class edges.

Efficiency analysis. Despite using the attention mechanism to learn weights from multi-hop neighbors, our model does not significantly increase computational costs. We compare the average running time per epoch for different models in Table 6. To ensure fair comparison and reduce the overall computational cost, all models utilize sparse matrices (including the initial feature matrix and adjacency matrix) for feature mapping and propagation processing. Although sparse GCN is fast in training time, it is not the fastest. Our model, HGRN, has a similar training time per epoch to APPNP and GPRGNN. Our model, as well as APPNP and GPRGNN, have several propagation layers, but their training time is not excessively high. This is due to the use of dimension reduction in feature mapping for propagation, which reduces the training cost. As a result, GCN (2 layers) does not run faster than SGC, which has 5 layers. Moreover, compared with the complex GAT model which also uses the attention mechanism, SHGRN and HGRN can not only ensure the improvement of performance but also reduces the consumption of training time.

7. Conclusion

In this paper, we present an attention-based graph convolutional network model to combat over-smoothing in GCN. It aggregates features from multi-order neighbors and uses an attention mechanism to examine the node feature relationships. Compared to conventional graph attention models, our model optimally leverages high-order information to represent nodes. Additionally, we introduce a new strategy that iteratively updates the graph with small step sizes to improve edge weights. Furthermore, we provide a theoretical analysis exploring the relationship between our proposed model (HGRN) and existing models. Experimental results

demonstrate the efficacy of our proposed model compared to other graph models. In conclusion, edge weight adjustment impacts neighborhood aggregation, but for heterophilic graphs (where nodes tend to connect with different classes), the presence or absence of edges holds greater significance. In future work, we plan to investigate adding intra-class edges and reducing inter-class edges to reconfigure the graph structure.

CRediT authorship contribution statement

Liancheng He: Conceptualization, Data curation, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft. **Liang Bai:** Conceptualization, Funding acquisition, Project administration, Resources. **Hangyuan Du:** Writing – review & editing. **Jiye Liang:** Supervision.

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.

Acknowledgement

The authors express their gratitude to the editors and reviewers for their insightful comments and suggestions. This work is supported by National Key Research and Development Program of China (No. 2021ZD0113303), the National Natural Science Foundation of China (Nos. 62022052, 62276159).

References

[1] S. Abu-El-Haija, B. Perozzi, A. Kapoor, H. Harutyunyan, N. Alipourfard, K. Lerman, G.V. Steeg, A. Galstyan, Mixhop: higher-order graph convolutional architectures via sparsified neighborhood mixing, in: International Conference on Machine Learning, 2019, pp. 21–29.
[2] S. Brody, U. Alon, E. Yahav, How attentive are graph attention networks?, in: International Conference on Learning Representations, 2022.
[3] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and deep locally connected networks on graphs, in: International Conference on Learning Representations, 2014.
[4] J. Chen, T. Ma, C. Xiao, Fastgcn: fast learning with graph convolutional networks via importance sampling, in: International Conference on Learning Representations, 2018.
[5] M. Chen, Z. Wei, Z. Huang, B. Ding, Y. Li, Simple and deep graph convolutional networks, in: International Conference on Machine Learning, PMLR, 2020, pp. 1725–1735.
[6] Y. Chen, L. Wu, M.J. Zaki, Iterative deep graph learning for graph neural networks: better and robust node embeddings, in: Advances in Neural Information Processing Systems, 2020, pp. 19314–19326.
[7] E. Chien, J. Peng, P. Li, O. Milenkovic, Adaptive universal generalized pagerank graph neural network, in: International Conference on Learning Representations, 2021.
[8] D.A. Clevert, T. Unterthiner, S. Hochreiter, Fast and accurate deep network learning by exponential linear units (elus), in: International Conference on Learning Representations, 2016.
[9] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Neural Information Processing Systems, 2016, pp. 3844–3852.
[10] W. Feng, J. Zhang, Y. Dong, Y. Han, H. Luan, Q. Xu, Q. Yang, E. Kharlamov, J. Tang, Graph random neural networks for semi-supervised learning on graphs, in: Advances in Neural Information Processing Systems, 2020, pp. 22092–22103.
[11] L. Franceschi, M. Niepert, M. Pontil, X. He, Learning discrete structures for graph neural networks, in: International Conference on Machine Learning, 2019, pp. 1972–1982.
[12] S. Fu, S. Wang, W. Liu, B. Liu, B. Zhou, X. You, Q. Peng, X.Y. Jing, Adaptive graph convolutional collaboration networks for semi-supervised classification, Inf. Sci. 611 (2022) 262–276.
[13] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: International Conference on Machine Learning, 2017, pp. 1263–1272.
[14] W.L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Neural Information Processing Systems, 2017, pp. 1025–1035.
[15] W. Huang, Y. Li, R. Xu, J. Yin, L. Chen, M. Zhang, et al., Towards deepening graph neural networks: a gntk-based optimization perspective, in: International Conference on Learning Representations, 2022.
[16] D. Jin, Z. Yu, C. Huo, R. Wang, X. Wang, D. He, J. Han, Universal graph convolutional networks, Adv. Neural Inf. Process. Syst. 34 (2021).
[17] W. Jin, T. Derr, Y. Wang, Y. Ma, Z. Liu, J. Tang, Node similarity preserving graph convolutional networks, in: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, 2021, pp. 148–156.
[18] D. Kim, A. Oh, How to find your friendly neighborhood: graph attention design with self-supervision, in: International Conference on Learning Representations, 2021.
[19] K.I. Kim, F. Steinke, M. Hein, Semi-supervised regression using Hessian energy with an application to semi-supervised dimensionality reduction, in: Proceedings of the 22nd International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, 2009, pp. 979–987.
[20] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations, 2017.
[21] J. Klicpera, A. Bojchevski, S. Günnemann, Predict then propagate: graph neural networks meet personalized pagerank, in: International Conference on Learning Representations, 2019.
[22] S. Kumar, A. Mallik, A. Khetarpal, B. Panda, Influence maximization in social networks using graph embedding and graph neural network, Inf. Sci. 607 (2022) 1617–1636.

[23] Q. Li, Z. Han, X.M. Wu, Deeper insights into graph convolutional networks for semi-supervised learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2018.
[24] Z. Li, F. Nie, X. Chang, Y. Yang, C. Zhang, N. Sebe, Dynamic affinity graph construction for spectral clustering using multiple features, IEEE Trans. Neural Netw. Learn. Syst. 29 (2018) 6323–6332.
[25] J. Liao, W. Zhou, F. Luo, J. Wen, M. Gao, X. Li, J. Zeng, Sociallgn: light graph convolution network for social recommendation, Inf. Sci. 589 (2022) 595–607.
[26] W. Liu, X. Ma, Y. Zhou, D. Tao, J. Cheng, p -Laplacian regularization for scene recognition, IEEE Trans. Cybern. 49 (2019) 2927–2940.
[27] Y. Liu, X. Wang, S. Wu, Z. Xiao, Independence promoted graph disentangled networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2020, pp. 4916–4923.
[28] M. Luo, X. Chang, L. Nie, Y. Yang, A.G. Hauptmann, Q. Zheng, An adaptive semisupervised feature analysis for video semantic recognition, IEEE Trans. Cybern. 48 (2018) 648–660.
[29] J. Ma, P. Cui, K. Kuang, X. Wang, W. Zhu, Disentangled graph convolutional networks, in: International Conference on Machine Learning, 2019, pp. 4212–4221.
[30] H. Pei, B. Wei, K.C.C. Chang, Y. Lei, B. Yang, Geom-gcn: geometric graph convolutional networks, in: International Conference on Learning Representations, 2020.
[31] Y. Rong, W. Huang, T. Xu, J. Huang, Dropedge: towards deep graph convolutional networks on node classification, in: International Conference on Learning Representations, 2020.
[32] M. Schlichtkrull, T.N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, M. Welling, Modeling relational data with graph convolutional networks, in: European Semantic Web Conference, 2018, pp. 593–607.
[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: Neural Information Processing Systems, 2017, pp. 5998–6008.
[34] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations, 2018.
[35] P. Velickovic, W. Fedus, W.L. Hamilton, P. Liò, Y. Bengio, R.D. Hjelm, Deep graph infomax, in: International Conference on Learning Representations, 2019.
[36] J. Wang, J. Liang, J. Cui, J. Liang, Semi-supervised learning with mixed-order graph convolutional networks, Inf. Sci. 573 (2021) 171–181.
[37] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, P.S. Yu, Heterogeneous graph attention network, in: The World Wide Web Conference, 2019, pp. 2022–2032.
[38] Y. Wang, J. Ren, D.M. Yan, J. Guo, X. Zhang, P. Wonka, Mgcn: descriptor learning using multiscale gcns, ACM Trans. Graph. 39 (2020) 122.
[39] Q. Wei, J. Wang, X. Fu, J. Hu, X. Li, Aic-gnn: adversarial information completion for graph neural networks, Inf. Sci. 626 (2023) 166–179.
[40] F. Wu, T. Zhang, A.H. de Souza, C. Fifty, T. Yu, K.Q. Weinberger, Simplifying graph convolutional networks, in: International Conference on Machine Learning, 2019, pp. 6861–6871.
[41] Y. Wu, W. Liu, X. Yu, The identical distribution hypothesis is equivalent to the parameter discrepancy hypothesis: adversarial attacks on graph neural networks, Inf. Sci. 623 (2023) 481–492.
[42] K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks, in: International Conference on Learning Representations, 2018.
[43] K. Xu, C. Li, Y. Tian, T. Sonobe, K. ichi Kawarabayashi, S. Jegelka, Representation learning on graphs with jumping knowledge networks, in: International Conference on Machine Learning, 2018, pp. 5449–5458.
[44] H. Yang, K. Ma, J. Cheng, Rethinking graph regularization for graph neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2021, pp. 4573–4581.
[45] Y. Yang, Z. Feng, M. Song, X. Wang, Factorizable graph convolutional networks, in: Advances in Neural Information Processing Systems, 2020, pp. 20286–20296.
[46] R. Ying, R. He, K. Chen, P. Eksombatchai, W.L. Hamilton, J. Leskovec, Graph convolutional neural networks for web-scale recommender systems, in: ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 974–983.
[47] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, D.Y. Yeung, Gaan: gated attention networks for learning on large and spatiotemporal graphs, in: Uncertainty in Artificial Intelligence, 2018, pp. 339–349.
[48] R. Zhou, X. Chang, L. Shi, Y.D. Shen, Y. Yang, F. Nie, Person reidentification via multi-feature fusion with adaptive graph learning, IEEE Trans. Neural Netw. Learn. Syst. 31 (2020) 1592–1601.
[49] H. Zhu, P. Koniusz, Simple spectral graph convolution, in: International Conference on Learning Representations, 2021.
[50] X. Zhu, G. Tang, P. Wang, C. Li, J. Guo, S. Dietze, Dynamic global structure enhanced multi-channel graph neural network for session-based recommendation, Inf. Sci. 624 (2023) 324–343.