Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 3911137, 10 pages
https://doi.org/10.1155/2021/3911137

Research Article

Semisupervised Classification with High-Order Graph Learning Attention Neural Network

Wu-Lue Yang,[1] **Xiao-Ze Chen**,[2] **and Xu-Hua Yang**[2]

[1]*School of Optical and Electronic Information, Huazhong University of Science and Technology, Wuhan 430074, China*
[2]*College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China*

Correspondence should be addressed to Xu-Hua Yang; xhyang@zjut.edu.cn

Received 6 September 2021; Revised 17 November 2021; Accepted 18 November 2021; Published 7 December 2021

Academic Editor: Jude Hemanth

At present, the graph neural network has achieved good results in the semisupervised classification of graph structure data. However, the classification effect is greatly limited in those data without graph structure, incomplete graph structure, or noise. It has no high prediction accuracy and cannot solve the problem of the missing graph structure. Therefore, in this paper, we propose a high-order graph learning attention neural network (HGLAT) for semisupervised classification. First, a graph learning module based on the improved variational graph autoencoder is proposed, which can learn and optimize graph structures for data sets without topological graph structure and data sets with missing topological structure and perform regular constraints on the generated graph structure to make the optimized graph structure more reasonable. Then, in view of the shortcomings of graph attention neural network (GAT) that cannot make full use of the graph high-order topology structure for node classification and graph structure learning, we propose a graph classification module that extends the attention mechanism to high-order neighbors, in which attention decays according to the increase of neighbor order. HGLAT performs joint optimization on the two modules of graph learning and graph classification and performs semisupervised node classification while optimizing the graph structure, which improves the classification performance. On 5 real data sets, by comparing 8 classification methods, the experiment shows that HGLAT has achieved good classification results on both a data set with graph structure and a data set without graph structure.

1. Introduction

Real-world data sets have many manifestations. From a macro point of view, we divide them into two data types with graph structure and nongraph structure in this paper. Graph structure data refers to data that is in the form of a network. The network is a way of representing relationship information between entities. Many real-world data sets can be represented by networks. For example, the citation network that reflects the citation relationship between scientific papers [1], the social network that facilitates the association and social activities between users [2], the protein interaction network that involves complex biological processes [3], etc. Nongraph structure data refers to data that does not have a network form, such as image data in computer vision or economic data of a city.

At present, there are many research results on two types of data classification of graph structure and nongraph structure. For example, for data with nongraph structure, SVM [4], random forest [5], logistic regression [6], and other machine learning methods and clustering methods [7] can achieve excellent classification results. In a network with a graph structure, node classification usually uses the given categories of some nodes to predict the category of unlabeled nodes. This task is also called semisupervised node classification. For example, the traditional label propagation algorithm [8, 9] learns the models that transform node features into node labels and add regularization items to classify nodes; the Skipgram model [10] on the text corpus inspired modern graph embedding methods, such as DeepWalk and random walk. These methods generally calculate the network node embedding first and then perform node classification. Data classification is to merge data with certain common attributes or characteristics and distinguish the data through the attributes or characteristics of its categories.

In recent years, graph neural networks (GNNs) [11] have received more and more attention in solving the problem of semisupervised node classification in noneuclidean space. In addition to node classification, GNNs are also widely used in knowledge graphs [12], adversarial attacks [13], combinatorial optimization [14], computer vision [15], and many other practical problems. As researchers discovered that convolutional neural networks [16] can significantly improve computer vision tasks by learning layered filters, how to extend convolution operations from European spatial data (such as images) to graph structure data has become a research hotspot. The graph convolutional network (GCN) [17] proposed by Kipf and Welling in 2014 has made great progress in the classification of graph structure data. It is the basis of many subsequent complex graph neural network models.

On the basis of GCN, Veličkovićet al. [18] proposed a graph attention neural network (GAT). GAT mainly uses node features and 1-hop neighbors to calculate attention scores. By assigning different weights to different nodes in the neighborhood, it can effectively combine the discrete structure and node features between data points. Compared to graph convolutional neural networks (GCNs), GAT enhances the ability of processing noise and greatly reduces dependence on graph structure. Although many successes have been achieved, it should be noted that the attention score in GAT is mainly calculated based on the content of the nodes, the graph structure is only used for the calculation of the attention coefficient, and only one-hop neighbors will participate in the calculation, which does not make good use of the high-order neighbor information of the graph topology.

Although the graph neural network can better handle those network topology data with complete and available relationships, its application scenarios are greatly restricted when there is no network topology structure or the network structure data is incomplete and noisy. Therefore, transforming nongraph structure data into graph structure data and optimizing incomplete or noisy graph structure data are challenging problems in the current application field of graph neural networks.

For data sets without network topology, a feasible solution is to artificially construct a graph structure based on the similarity between the data [19–21], for example, k-nearest neighbor (kNN) with Gaussian kernel and ε-radius network construction technology. However, these models rely on the selection of the number of neighbors k, the radius ε, and similarity metrics. The values of k and ε are very sensitive to the local structure of the data and the network topology obtained by artificially selecting the values and may not correctly reflect the original data distribution characteristics. Therefore, in this paper, we use a Bayesian optimization search to obtain the appropriate number of neighbors k. In addition, Li et al. [22] used a block-diagonal graph structure, which can easily obtain a reliable number of clusters and retain the internal cluster structure of the subspace. This method focuses more on the improvement of clustering performance but lacks the optimization of network reconstruction. Henaff et al. [23] proposed that a fully connected supervised graph can be learned from a separate network. Since graph learning is performed separately, it cannot guarantee that the learned graph structure can be well adapted to semisupervised classification. Franceschi et al. proposed a new model LDS [2], which parameterizes the entire graph and approximately solves bilevel programming to jointly learn the graph structure and GNNs parameters. The number of parameters to be optimized by this method increases quadratically with the increase of nodes, the memory and time costs are too large, and it is difficult to be used for the calculation of large data sets.

In order to solve these limitations, in this paper, we propose a high-order graph learning attention neural network model (HGLAT) that can effectively improve classification performance. The main contributions of this model are as follows:

(1) We propose the method for learning graph structure based on an improved variational graph autoencoder, which can learn and optimize graph structure for data sets without topological graph structure and data sets with missing topological structure.

(2) We propose the semisupervised classification method that extends the attention mechanism in GAT to higher-order neighbors in the network topology, which can effectively capture the information of high-order neighbors by attenuating the attention of high-order neighbors in accordance with the increasing order.

(3) We propose the high-order graph learning attention neural network, which jointly optimize graph learning and semisupervised classification to improve the performance of node classification.

The rest of this paper is organized as follows. Section 2 shows the related work. Section 3 describes the high-order graph attention neural network proposed in this paper. Section 4 is the numerical simulation and result of the analysis. Section 5 gives conclusions.

2. Related Work

At present, there are many semisupervised classification algorithms. Our proposed algorithm is obviously different from the existing algorithms. We use an improved variational autoencoder for graph learning to obtain richer hidden information and propose a high-order graph attention neural network for semisupervised classification. The following is the relevant research basis.

2.1. Graph Convolutional Neural Network. Network embedding aims to use low-dimensional and dense vectors to represent high-dimensional, sparse networks while retaining the topological structure of the network [24]. Researchers have proposed many network embedding models ranging from unsupervised (such as DeepWalk [25]) to supervised learning methods (such as SemiGraph [26]). In general, similar or close nodes in the network have similar embedding representations [27]. Graph neural network (GNN)

is a series of neural network models specially developed for learning network data. GNN learns the structural characteristics of the network through efficient neighborhood node information aggregation [11] to obtain the embedded representation of the network.

Graph convolutional network (GCN) [17] is generally used to deal with irregular graph structure data. Recently, many related algorithms have been proposed, which can be generally divided into four categories: graph convolutional neural networks based on the convolution theorem, graph convolutional neural networks based on aggregation functions, deep graph convolutional neural networks, large-scale graph convolutional neural networks. GCN uses a spectral-based convolution filter, by which nodes can aggregate features from their respective local map neighborhoods for representation learning. This convolution learning mechanism has been proven successful in many analysis tasks, such as link prediction [28], image recognition [29], and new drug discovery [30]. Following a similar convolutional graph embedding scheme, researchers have proposed multiple GNN models with more efficient information aggregators. For example, the graph attention network (GAT) [18] learns to assign different importance weights to each node so that nodes can highlight important neighborhoods when aggregating features. GraphSAGE [31] learned a set of aggregation functions for each node to flexibly aggregate information from neighborhoods in different hops. Recently, APSEGAT [32] explored a graph attention network based on adaptive progressive scaling.

2.2. Graph Topology Structure Optimization. Graph structure data with noise or incompleteness and other uncertainties often lead to poor performance of graph analysis methods [33]. Therefore, in recent years, researchers have proposed optimizing graph structure to improve the performance of graph analysis algorithms. Rong et al. [34] thought that overfitting and oversmoothing are the two main obstacles to the development of multilayer graph convolutional networks for node classification. Randomly deleting a certain number of edges from the input graph during each training period can prevent oversmoothing. Franceschi et al. proposed an LDS model [2], which parameterizes the entire graph and approximately solves two-level programming to jointly learn graph structure and GNN parameters. Li et al. [35] proposed the EACI model, which proposed a zero-shot complex event detection method and paid more attention to the semantic correlation between concepts and events. Haija et al. [36] pointed out that it is possible to learn the neighborhood hybrid relationship of general categories by repeatedly mixing the feature representations of neighbors at various distances. Liang et al. optimized the graph topology by adjusting the edges between and within the community [37] and obtained potential information by jointly improving the network topology and learning network parameters. Shi et al. [38] maximized the consistency of aggregated information by aligning the network in topological and semantic aspects.

3. High-Order Graph Learning Attention Neural Network (HGLAT)

In view of the problem that the graph neural network cannot process data without graph structure, and the effect is not good when the graph structure data is missing and with noise, we can learn the corresponding graph topology for the data without graph structure through the improved variational graph autoencoder, the graph structure can be optimized for missing or noisy graph structure data, and a high-order attention mechanism is proposed to perform more efficient semisupervised classification of nodes in the graph structure.

3.1. Problem Description. Given a data set without graph structure or a complex network $G = (V, E)$ without weights and directions, where $V = \{v_1, v_2, \ldots, v_N\}$ represents the set of nodes and E represents the set of edges, we use N, M to denote the number of nodes and edges, respectively, and A to denote the adjacency matrix of $N \times N$. If there is an edge between v_i and v_j, then $A_{ij} = 1$; otherwise, $A_{ij} = 0$. In the module of the learning graph structure we proposed, $\widehat{A}_{ij}$ represents the probability of the edge between v_i and v_j, and $\widehat{A}_{ij} \in [0, 1]$.

Our goal is to extend the attention mechanism from direct neighbors of nodes to higher-order neighbors, making full use of the higher-order graph topology information to classify data points or network nodes.

3.2. Algorithm Framework. The model consists of a graph learning module and a semisupervised classification module. The model architecture is shown in Figure 1 and consists of the following four steps.

(1) If the object classified is a data set without a graph structure, we use the k-nearest neighbor algorithm (kNN) to transform the data set into an initialized sparse network, denoted by "initial graph structure A." If the object to be classified is nodes of the network, no processing is done, which is the "initial graph structure A."

(2) Graph structure learning: the graph structure is reconstructed through our improved variational graph autoencoder IVGAE. By adding a regularization term to the reconstruction loss, the generated graph structure is guaranteed to be sparse and connected, and a new graph structure $\widehat{A}$ is generated.

(3) High-order graph attention neural network (HGAT) model: it is proposed to extend the attention mechanism to high-order neighbors and effectively aggregate features from high-order neighbors.

(4) Semisupervised classification training: the two modules of graph structure learning and HGAT (semisupervised module) are jointly optimized to complete semisupervised classification.

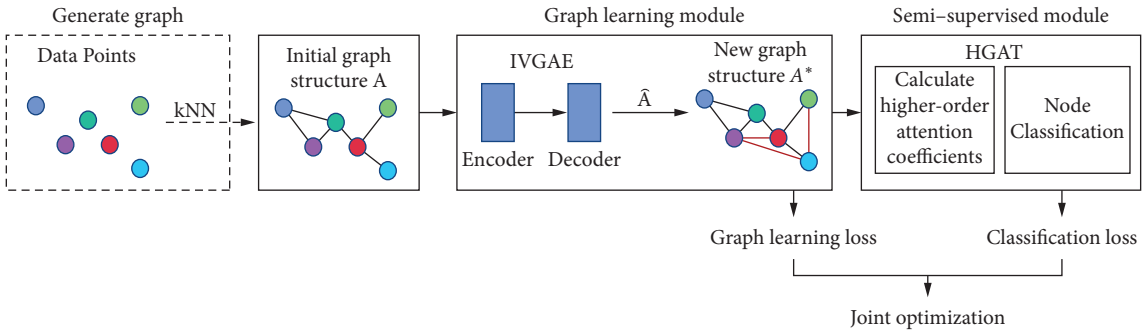

Figure 1: HGLAT model architecture. The dashed box is a data set without a graph structure. The initial graph structure needs to be generated through kNN clustering to be used as the input part of IVGAE. This step is not required for a data set with a graph structure. IVGAE is a graph learning module, which generates a new graph structure $\widehat{A}$, and calculates the graph learning loss. $\widehat{A}$ is symmetrized and combined with the initial graph structure to obtain A^*, and then, A^* is used as the input of HGAT. HGAT calculates high-order neighbor attention coefficients, performs semisupervised node classification, and calculates the classification loss. The two modules are jointly optimized to obtain semisupervised classification results.

3.3. Learning Graph Structure Based on Improved Variational Graph Autoencoder. In this section, we propose an improved variational graph autoencoder (IVGAE). IVGAE is based on the structure of the variational graph autoencoder (VGAE) [11], and we propose a new objective function and optimization method to learn graph structure. Using GCN as an encoder, the propagation mode between GCN layers can be expressed as

$$H^{(l+1)} = \mathrm{ReLu}\left(\widetilde{D}^{-(1/2)}\widetilde{A}\widetilde{D}^{-(1/2)}H^{(l)}W^{(l)}\right), \tag{1}$$

where $H^{(l)}$ represents the output of layer l; $H^{(0)} = X$. $\widetilde{A} = A + I$ is an adjacency matrix with self-loops, I is the identity matrix, $\widetilde{D}$ is the degree matrix of $\widetilde{A}$, and $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$. $W^{(l)}$ is the parameter of the first layer. The output layer result obtained by an L-layer GCN is

$$Z_{\mathrm{out}} = H^{(L)}W^{(L)}, \tag{2}$$

where $Z_{\mathrm{out}} \subseteq R^{N\times c}$, where c represents the dimension of the embedding vector of the output layer node. GCN encoding is used to obtain the variance and mean of the embedding of the node. This process can be expressed as

$$\mu = GCN_{\mu}(X, A), \tag{3}$$

$$\log \sigma = GCN_{\sigma}(X, A), \tag{4}$$

where $\mu \subseteq R^{N\times \mathrm{f}}$ and $\log \sigma \subseteq R^{N\times \mathrm{f}}$, respectively, represent the mean and variance of the embedding vector output by the encoder, f represents the dimension of the vector, $X = \{x_1, x_2, \ldots, x_N\} \subseteq R^{N\times d}$ represents the feature matrix, x_i represents the feature vector of node i, and d represents the dimension of node feature. GCN_{μ} and GCN_{σ} share the first layer parameter $W^{(0)}$. After obtaining the mean and variance, the embedding vector Z can be obtained by sampling, and the reconstructed graph structure

$$\widehat{A} = \left(ZZ^T\right) \tag{5}$$

can be obtained by decoding the embedding by the decoder.

Before semisupervised classification, the original graph structure is optimized, and the generation of a new graph structure can be regarded as an unsupervised task. Different from only using the similarity measure as the method of learning graph structure, here, we use IVGAE to supplement the original topological structure with edge information and, at the same time, implement the model for unsupervised learning according to the original graph structure and node features, so that we can better find some hidden information that is not reflected in the existing structure. Specifically, we take the node feature X and the original graph structure A as input, and the encoder and decoder are defined in formulas (3)–(5). After decoding the encoder, we can get a new graph structure as

$$\widehat{A} = \mathrm{Decoder}\left(\mathrm{Encoder}\left(X, A\right)\right). \tag{6}$$

Because our graph learning model is to learn the hidden parts that are not included in the original graph structure, the learned graph structure does not need to use the objective function to ensure that the graph structure generated by the training is the same as the original structure. So, our objective function does not need to calculate the cross-entropy between the new graph structure and the original graph structure. In addition, we hope that the distribution calculated by GCN is as similar as possible to the standard Gaussian. So, the loss function is composed of KL divergence as

$$\ell_{ivgae} = KL[q(Z|X, A)\|p(Z)], \tag{7}$$

where $q(Z|X, A)$ is the distribution calculated by GCN and $p(Z)$ is the standard Gaussian distribution. In order to ensure that the generated graph is meaningful and usable, we need to ensure that each node has at least one edge connected to other nodes, and we want the generated graph structure to be as sparse as possible. In order to meet these expectations, according to a general model proposed by Kalofolias et al. [39] that can learn graph structure without prior information available, we add constraints to formula (7) to obtain IVGAE, which is the objective function of the graph learning module as

$$\ell_{GL} = \ell_{ivgae} = KL[q(Z|X,A)\|p(Z)] - \left(\frac{\alpha}{n}1^T\log(\widehat{A}1) - \frac{\beta}{n^2}\|\widehat{A}\|_F^2\right), \tag{8}$$

where the parameters $\alpha > 0$ and $\beta \geq 0$ are used to control the penalties for connectivity and sparsity, respectively, and n represents the number of nodes in the graph. After getting our new adjacency matrix A^*, we need to symmetrize it to ensure that the values at the corresponding positions of the upper and lower triangles are consistent as

$$\widehat{A}_{ij}, \widehat{A}_{ji} = \begin{cases} \widehat{A}_{ij}, & \left|\widehat{A}_{ij}\right| < \left|\widehat{A}_{ji}\right|, \\ \widehat{A}_{ji}, & \left|\widehat{A}_{ij}\right| \geq \left|\widehat{A}_{ji}\right|. \end{cases} \tag{9}$$

We need to fuse the original structure with the new graph structure obtained and use the parameter η to control the proportion of the new and old structures during the fusion as

$$A^* = (1-\eta)A + \eta\widehat{A}. \tag{10}$$

3.4. High-Order Graph Attention Neural Network Model. The graph neural network generally learns the embedding representation of a node through its neighbors and combines the attribute value of the node with the graph structure. The strategy of aggregating node neighbors in GCN is to integrate its node representation with all first-order neighbors equally, regardless of neighbor differences. GAT considers the importance of different neighbors and assigns different weights to different neighbor nodes. First, the graph attention layer of GAT calculates the similarity coefficient.Between vertex i and its neighbor j according to the input node feature vector set $X = \{x_1, x_2, \ldots, x_N\} \subseteq R^{N\times d}$, where $f(\cdot)$ represents the mapping of $R^{N\times d} \times R^{N\times d} \longrightarrow R^{N\times d}$, this paper uses a single-layer feedforward neural network, and $W \in R^{d\times d}$ is a shared weight matrix, which increases the dimension of the vertex feature, which is a common feature enhancement method. $[\cdot\|\cdot]$ means to splice the transformed features of nodes i and j. In order not to lose the graph structure information, GAT uses a masked attention method, which only allocates attention to the set of neighboring nodes of the node. If not doing this, when performing self-attention calculations, attention will be distributed to all nodes in the graph, which will result in the loss of graph structure information. In order to make the coefficients comparable between different nodes, GAT uses the softmax function to normalize them:

$$a_{ij} = \text{softmax}_j\left(e_{ij}\right) = \frac{\exp\left(e_{ij}\right)}{\sum_{r\in\Gamma_i}\exp\left(e_{ir}\right)}, \tag{12}$$

where Γ_i represents the first-order neighbor set of nodes i. However, the propagation process of information in the graph is related not only to its first-order neighborhood but also to its high-order neighborhood. In both GCN and GAT, only the features of the first-order neighborhood node are aggregated, which to a certain extent hinders the capture of high-order interactive information between nodes in the graph and also limits the ability of the model. Therefore, we assign attention to the high-order neighborhood of the node according to the node weight matrix, and at the same time, we can make better use of the graph structure information, as shown in Figure 2. We propose to consider the k-order neighbor nodes in the graph to obtain an approximate high-order attention decay matrix:

$$Q = \text{Atte}(1)M + \text{Atte}(2)M^2 + \ldots + \text{Atte}(k)M^k, \quad \text{Atte}(k) = e^{\left(-k^2/2h^2\right)}, \tag{13}$$

where M is the transition matrix of $N * N$, the value of M_{ij} is determined by the ratio of the weight of the edge between nodes i and j to the sum of the weights of all connected edges of node i, $M_{ij} = A^*_{ij}/D_i$, where A^* is determined by equation (11), and $D_i = \sum_{j\in\Gamma_i} A^*_{ij}$, where Γ_i represents the set of neighbors of node i. Therefore, M^k can represent the k-order topological correlation between node j and node i, and different k can be selected for different data sets, so as to balance the accuracy and efficiency of the model. $\text{Atte}(k)$ represents the attenuation function. We can think that as the order k increases, the influence of the k-order neighbors on the node will gradually attenuate. Considering the effects of the three attenuation functions Logistic, Log-Logistic, and Gaussian through experiments, the best attenuation function Gaussian is obtained. h represents the parameter in the Gaussian decay function, which can be optimized. Integrating Q into equation (12), we get

$$a_{ij} = \frac{\exp\left(\text{Leaky ReLU}\left(Q_{ij}f\left(\left[W\overrightarrow{x_i}\|W\overrightarrow{x_j}\right]\right)\right)\right)}{\sum_{r\in\Gamma_i}\exp\left(\text{Leaky ReLU}\left(Q_{ir}f\left(\left[W\overrightarrow{x_i}\|W\overrightarrow{x_r}\right]\right)\right)\right),} \tag{14}$$

where LeakyReLU is the activation function.

$$e_{ij} = f\left(\left[W\overrightarrow{x_i}\|W\overrightarrow{x_j}\right]\right). \tag{11}$$

3.5. Algorithm Framework. The attention coefficients of nodes have been obtained in the previous section, and the embedding of each node can be obtained by aggregation according to the attention coefficient, so the output of each graph attention layer is

$$z_i^l = \sigma\left(\sum_{j\in\Gamma_i} a_{ij}Wz_j^{l-1}\right), \tag{15}$$

where z_i^l is the output representation of the l-th layer of node i, Γ_i represents the set of neighbors of i, and $\sigma(\cdot)$ represents the activation function. The multihead attention mechanism can be used to splice the results obtained in equation (15). Finally, the cross-entropy loss function

$$\ell_{GC} = -\sum_{i\in y_L}\sum_{j=1}^{C} Y_{ij}\ln\widehat{Y}_{ij} \tag{16}$$

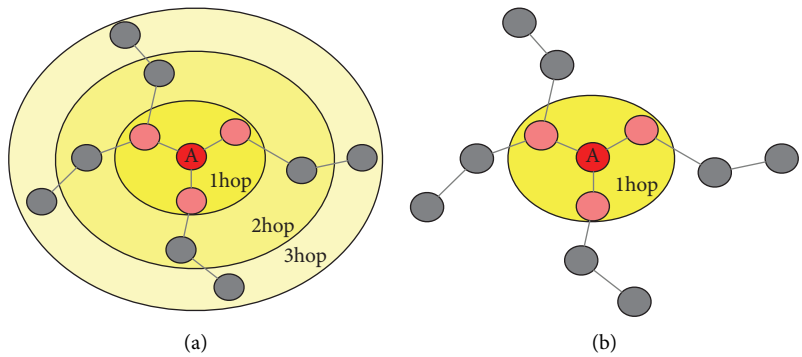

Figure 2: Comparison of high-level attention mechanism and GAT attention mechanism. Node A is the node that we need to request a hidden representation. (a) shows the attention mechanism of HGAT. It not only refers to the 1 hop neighbors but also calculates the attention coefficients of higher-order neighbors, and as the order increases, the color gradually becomes lighter; that is, the attention coefficient is distributed less. (b) shows the attention coefficient calculation method of GAT, and the attention coefficient is obtained only by calculating the neighbors of 1 hop.

is used for semisupervised classification, where $\widehat{Y}$ is the output softmax(z) of the last layer, and Y_{ij} is the label of the node. y_L represents the labeled nodes in the node set. After obtaining the loss of the node classification module, we can jointly optimize graph structure learning and semi-supervised node classification and define our objective function as

$$\ell_G = \delta \ell_{GL} + \ell_{GC}, \tag{17}$$

where ℓ_{GL} and ℓ_{GC} are reconstruction loss and classification loss, respectively, and $\delta \geq 0$ is a coefficient that controls the balance between the two parts.

4. Experiments

The 3 main objectives of the experiment are as follows. First, for a data set that has an available graph structure, we compare the HGLAT's performance on node classification with graph-based learning algorithms under the missing part of the graph structure. Second, we conduct node classification experiments on the data sets without graph structure to judge whether HGLAT can obtain good results on the semisupervised classification problem. We compare HGLAT with 8 well-known semisupervised classification methods. Third, we judge the influence of each part of the HGLAT on the result through the ablation experiment.

4.1. Datasets. The new algorithm proposed in this paper is numerically simulated on 5 data sets, which are divided into two parts: graph structure data and nongraph structure data.

4.1.1. Graph Structure Data

(1) Cora is a citation network data set [40], nodes correspond to documents, edges (nondirected) correspond to citations, and node features correspond to elements represented by the document's bag of words. Each node has a class label, which contains 2708 nodes, 5429 edges, 7 classes, and 1433 features for each node.

(2) Citeseer is also the benchmark data set of citation networks [40], which contains 3327 nodes, 4732 edges, 6 classes, and 3703 features per node. To evaluate the robustness of HGLAT on incomplete graphs, for Cora and Citeseer, we construct graphs with missing edges by randomly sampling 25%, 50%, and 75% of the edges.

4.1.2. Nongraph Structure Data. We use the following benchmark dataset [41] to evaluate HGLAT.

(1) Wine dataset: this data set is the composition data of three different types of wine produced in the same area in Italy, containing 178 data; each data has 14 characteristics.

(2) Cancer dataset: it comes from the Cancer Institute of the University of Ljubljana Medical Center in Yugoslavia. The dataset has 2 categories, 9 attributes, and a total of 286 instances.

(3) Digits dataset: it is a picture database of handwritten digits, each picture is a single digit from 0 to 9, it contains 1797 pictures, and each picture is an 8∗8 matrix.

4.2. Baseline Algorithms. We compare HGLAT with GCN and GAT. For the data set without graph topology, we first construct the kNN as the preprocessing step before applying GCN and GAT. In Table 1, we denote these two algorithms as kNN-GCN and kNN-GAT. GCN and GAT can be used directly for data with graph structure. To further evaluate HGLAT, we also compared HGLAT with well-known semisupervised learning methods, including label propagation (LP) [42] and semisupervised embedding (SemiEmb) [43]. At the same time, we also compared HGLAT and machine learning methods that do not use graph structures, including logistic regression (LogReg), random forest (RF), and two SVM methods, RBF SVM and Linear SVM.

We set the parameters of the HGLAT algorithm as follows. For the Cora and Citeseer datasets, we use the

Table 1: Comparison of results of various classification algorithms. Test accuracy (±standard deviation) on various classification data sets, expressed as a percentage. All experiments were repeated using 5 different random seeds. For each data set, the highest accuracy score is marked in bold. We compare HGLAT with several supervised benchmarks and semisupervised learning methods. HGLAT achieved the best results on 4 of the data sets.

Alg\dataset	Wine	Cancer	Digits	Citeseer	Cora
LogReg	92.1(1.3)	93.3(0.5)	85.5(1.5)	62.2(0.0)	60.8(0.0)
RF	93.7(1.6)	92.1(1.7)	83.1(2.6)	60.7(0.7)	58.7(0.4)
RBF SVM	94.1(2.9)	91.7(3.1)	86.9(3.2)	60.2(0.0)	59.7(0.0)
Linear SVM	93.9(1.6)	90.6(4.5)	87.1(1.8)	58.3(0.0)	58.9(0.0)
LP	89.8(3.7)	76.6(0.5)	**91.9(3.1)**	23.2(6.7)	37.8(0.2)
SemiEmb	91.9(0.1)	89.7(0.1)	90.9(0.1)	68.1(0.1)	63.1(0.1)
kNN-GCN	93.2(3.1)	93.8(1.4)	91.3(0.5)	70.9(0.5)	81.2(0.5)
kNN-GAT	94.5(2.5)	93.1(1.2)	90.5(0.8)	72.1(0.7)	82.6(0.7)
HGLAT	**97.5(0.3)**	**94.8(1.1)**	91.0(0.7)	**74.4(0.5)**	**84.7(0.4)**

experimental settings of [17, 18]. For Wine, Cancer, and Digits data sets, we use the experimental settings of [2]. The experimental results are the average of 10 runs with different random seeds. In the graph learning module, we initialize the weights as described in [43], using a 32-dimensional hidden layer and 16-dimensional output layer. The symmetry threshold ρ is chosen to be 0.6. The choice of hyperparameters in different data sets is not the same. We have listed hyperparameters suitable for each data set as shown in Table 2.

The semisupervised classifier in HGLAT uses a two-layer GAT model. The first layer consists of $K = 8$ attention heads, and each attention head calculates 8 features (64 features in total). The second layer is used for classification, calculating a single attention head of C features where C is the number of classes, and then performing softmax activation. In order to cope with the small training set size, regularization is freely applied in the model. During training, we apply L2 regularization with $\lambda = 0.0005$. In addition, we set dropout to 0.6 and apply it to the two-layer input and the normalization of the attention coefficient. Glorot initialization is used for initialization [3], and Adam SGD optimizer [44] is used for 300 epoch training to minimize the cross-entropy on the training node. The initial learning rate of the graph learning module is 0.005. And the highest order in Q is 2, which means we consider at most the second-order neighbors of each node. HGLAT is implemented in Pytorch [45] geometric deep learning extension library [1]. The implementation of the supervised baseline method and LP comes from scikit-learn python [46]. When kNN needs to be used for initialization in the algorithm, we recommend using a Bayesian optimization search for the selection of K value.

4.3. Experimental Results

4.3.1. Node Classification. On the five data sets of Wine, Cancer, Digits, Citeseer, and Cora, we compared the HGLAT algorithm with 8 well-known classification algorithms. The results are shown in Table 1. The data sets with graph structure including Citeseer and Cora retain the complete graph structure. Among them, the datasets with graph structure including Citeseer and Cora retain the complete graph structure, and the datasets without graph structure include Wine, Cancer, and Digits, which are generated by kNN before the learning method that needs to transfer the graph structure. First of all, the results show that in 4 of the 5 data sets, including Wine, Cancer, Citeseer, and Cora, HGLAT has the best performance and the third performance on the Digits data set. Overall, HGLAT has the best performance. Second, it can be seen that supervised machine learning methods, including LogReg, RF, RBF SVM, and Linear SVM, only perform well on nongraph structured data sets (Wine, Cancer, Digits), while in graph-structured data sets, Citeseer and Cora results are not good, and semisupervised learning methods LP and SemiEmb have not obtained better results than supervised machine learning methods. Third, the results of kNN-GCN, kNN-GAT, and HGLAT show that graph neural networks can achieve good results on all data sets, which also shows that it is meaningful to extend graph neural networks to nongraph structure data sets.

Table 2: Hyperparameters related to HGLAT on different data sets.

Dataset	K	α	β	η	δ
Wine	21	0.2	0.1	0.2	0.5
Cancer	15	0.45	0.15	0.2	0.5
Digits	25	0.35	0.1	0.15	0.3
Cora	—	0.3	0	0.1	0.3
Citeseer	—	0.3	0	0.3	0.2

To further test the superiority of HGLAT, we considered the experimental results when a certain percentage of edges in the graph structure were randomly deleted. We give the edge retention rate and randomly sample the edges. Figure 3 shows the accuracy of classification under different percentages of retained edges. First, it can be observed from Figure 3 that HGLAT shows the best classification effect in all cases, especially when the edge retention rate is low. Second, we found that the effect of GAT is not good when the missing edges are severe. This shows that although GAT does not depend on the graph structure to a certain extent, it is not enough to get good results just by reweighting the edges and node features. It also shows that the HGLAT learning graph structure before classification is meaningful for GAT.

4.3.2. Ablation Experiment. In order to further verify the effectiveness of each module of HGLAT, we conduct ablation studies by deleting some of the modules of HGLAT, including three types of No-Reg, No-Fus, and GLAT, as shown in Figure 4.

We conduct ablation experiments on the Cora dataset with different edge retention rates and reported the results in Figure 4. The comparison between No-Reg and HGLAT shows that the addition of regularization constraints can ensure that the generated graph structure is more reasonable and the result is better. The comparison of GLAT and HGLAT shows that when we consider the high-order

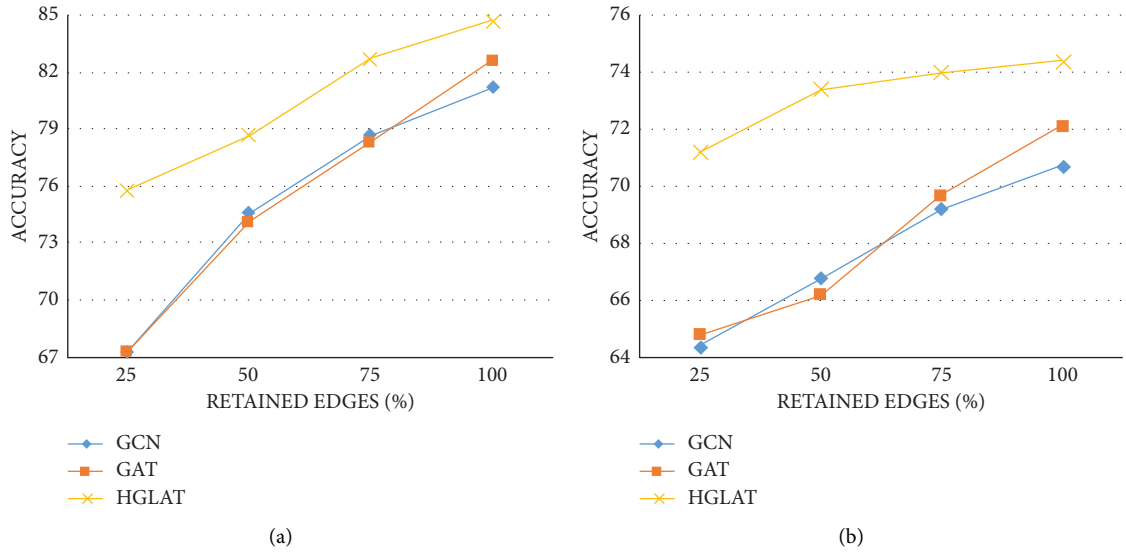

Figure 3: Comparison of classification accuracy of HGLAT, GCN, and GAT in the Cora (a) and Citeseer (b) datasets when retaining different proportions of edges.

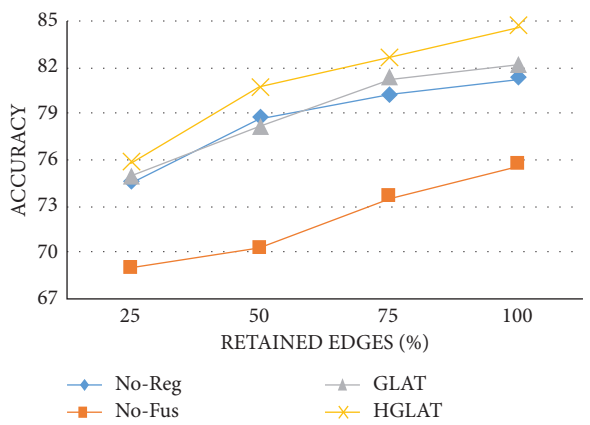

Figure 4: Classification accuracy of ablation experiments on Cora datasets with different edge retention rates. On the Cora dataset with different edge retention rates, the classification accuracy of HGLAT and the three No-Reg, No-Fus, and GLAT ablation experiments are compared, where No-Reg means that the graph structure obtained by the graph learning module is not symmetrized and regularized constraints; No-Fus means that the decoder graph structure and the original graph structure are not allowed to merge; that is, formula (10) is not used. GLAT means that the edges are reweighted only by the relationship between the first-order neighbors.

neighbor relationship, it can help us improve the classification accuracy to a certain extent. The comparison between No-Fus and HGLAT shows that fusing the original graph structure with the generated graph structure is helpful to the accuracy of the model.

4.3.3. Running Time Statistics. We record the running time of HGLAT on each data set to more comprehensively analyze the model proposed in the paper. The Tensorflow version is 1.15.0, CPU is i9-9900kf, GPU is RTX 3090, and operating system is Windows 10. The running time statistics are shown in Table 3, where the value is the average time for the model to complete training one time, and the unit is seconds.

Table 3: Running time statistics under different data sets.

Algorithm	Wine	Cancer	Digits	Citeseer	Cora
HGLAT	0.1482	0.5664	6.668	31.57	20.36

5. Conclusion

In this paper, we propose a new high-order graph learning attention neural network (HGLAT). Through the improved variational graph autoencoder, HGLAT can learn the corresponding graph topology for nongraph structure data and optimize graph structure for data with missing or noisy graph structure. HGLAT extends attention to high-order neighbors, effectively aggregates the features from high-order neighbors, and makes full use of high-order graph topology information. It has achieved good semisupervised classification results for nongraph structure and graph structure data. In the future, we plan to improve our preprocessing process, for example, iterative updating the loss function to obtain a more reasonable initial graph structure. We will also conduct extended research on the semisupervised network model and generate network clusters through some improved clustering algorithms to make up for the lack of node labels in the network. And we consider introducing more network information, such as edge information and heterogeneous network node information. to make the network model have a broader application prospect.

Data Availability

The datasets used in the experiments of this work are publicly available.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This research was supported by the National Natural Science Foundation of China (No. 61773348).

References

[1] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," 2019, https://arxiv.org/abs/1903.02428.

[2] L. Franceschi, M. Niepert, M. Pontil, and X. He, "Learning discrete structures for graph neural networks," 2019, https://arxiv.org/abs/1903.11960.

[3] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256, Sardinia, Italy, May 2010.

[4] S. Shan, "Support vector machine," *Machine Learning Models and Algorithms for Big Data Classification*, pp. 207–235, Springer US, Manhattan, NY, USA, 2016.

[5] A. Liaw and M. Wiener, "Classification and regression by random forest," *R News*, vol. 2, no. 3, pp. 18–22, 2002.

[6] M. P. LaValley, "Logistic regression," *Circulation*, vol. 117, no. 18, pp. 2395–2399, 2008.

[7] X.-H. Yang, Q.-P. Zhu, Y.-J. Huang, J. Xiao, L. Wang, and F.-C. Tong, "Parameter-free Laplacian centrality peaks clustering," *Pattern Recognition Letters*, vol. 100, pp. 167–173, 2017.

[8] J. Weston, F. Ratle, H. Mobahi, and R. Collobert, "Deep learning via semi-supervised embedding," in *Neural Networks: Tricks of the Trade*, pp. 639–655, Springer, Berlin, Germany, 2012.

[9] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: a geometric framework for learning from labeled and unlabeled examples," *Journal of Machine Learning Research*, vol. 7, no. 11, pp. 2399–2434, 2006.

[10] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, Lake Tahoe, NV, USA, December 2013.

[11] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2008.

[12] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proceedings of the European Semantic Web Conference*, pp. 593–607, Portorož, Slovenia, June 2018.

[13] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial attacks on neural networks for graph data," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2847–2856, London, UK, August 2018.

[14] Z. Li, Q. Chen, and V. Koltun, "Combinatorial optimization with graph convolutional networks and guided tree search," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 539–548, Montréal, Canada, December 2018.

[15] S. Qi, W. Wang, B. Jia, J. Shen, and S. C. Zhu, "Learning human object interactions by graph parsing neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 401–417, Munich, Germany, September 2018.

[16] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[17] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, https://arxiv.org/abs/1609.02907.

[18] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," 2017, https://arxiv.org/abs/1710.10903.

[19] T. C. Silva and L. Liang Zhao, "Network-based high level data classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 6, pp. 954–970, 2012.

[20] T. C. Silva and L. Zhao, "High-level pattern-based classification via tourist walks in networks," *Information Sciences*, vol. 294, pp. 109–126, 2015.

[21] Z. Li, F. Nie, X. Chang, Y. Yang, C. Zhang, and N. Sebe, "Dynamic affinity graph construction for spectral clustering using multiple features," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 12, pp. 6323–6332, 2018.

[22] Z. Li, F. Nie, X. Chang, L. Nie, H. Zhang, and Y. Yang, "Rank-constrained spectral clustering with flexible embedding," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 12, pp. 6073–6082, 2018.

[23] M. Henaff, J. Bruna, and Y. L. Cun, "Deep convolutional networks on graph structured data," 2015, https://arxiv.org/abs/1506.05163.

[24] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833–852, 2018.

[25] B. Perozzi, R. AlRfou, and S. Skiena, "Deepwalk: online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701–710, New York, NY, USA, August 2014.

[26] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: a survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.

[27] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1225–1234, Philadelphia, PA, USA, August 2006.

[28] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American Society for Information Science and Technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[29] D. Keysers, T. Deselaers, C. Gollan, and H. Ney, "Deformation models for image recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 8, pp. 1422–1435, 2007.

[30] G. Sliwoski, S. Kothiwale, J. Meiler, and E. W. Lowe, "Computational methods in drug discovery," *Pharmacological Reviews*, vol. 66, no. 1, pp. 334–395, 2014.

[31] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 1024–1034, Long Beach, CA, USA, December 2017.

[32] P. N. Chowdhury, P. Shivakumara, S. Kanchan, R. Raghavendra, T. Lu, and D. Lopresti, "Graph attention network for detecting license plates in crowded street scenes," *Pattern Recognition Letters*, vol. 140, no. 1, 2020.

[33] B. Müller, J. Reinhardt, and M. T. Strickland, *Neural Networks: An Introduction*, Springer Science & Business Media, Berlin, Germany, 2012.

[34] Y. Rong, W. B. Huang, X. T. Yang, and H. J. zhou, "Dropedge: towards deep graphconvolutional networks on node classification," in *Proceedings of the International Conference on Learning Representations (ICLR)*, Addis Ababa, Ethiopia, April 2020.

[35] Z. Li, L. Yao, X. Chang, K. Zhan, J. Sun, and H. Zhang, "Zero-shot event detection via event-adaptive concept relevance mining," *Pattern Recognition*, vol. 88, pp. 595–603, 2019.

[36] S. A. E. Haija, B. Perozzi, A. Kapoor et al., "Mixhop: higher-order graph convolutional architectures via sparsified neighborhood mixing," in *Proceedings of the 2019 International Conference on Machine Learning*, pp. 21–29, Long Beach, CA, USA, June 2019.

[37] Y. Liang, K. Z. Sheng, C. X. Chun, J. Di, Y. Bo, and G. Y. Fang, "Topology optimization based graph convolutional network," in *Proceedings of the 2019 International Joint Conference on Artificial Intelligence*, pp. 4054–4061, Beijing, China, August 2019.

[38] M. Shi, Y. Y. Fei, and Z. X. Quan, "Topology and content co-alignment graph convolutional learning," 2020, https://arxiv.org/abs/2003.12806.

[39] V. Kalofolias, "How to learn a graph from smooth signals," in *Proceedings of the Artificial Intelligence and Statistics*, pp. 920–929, Cadiz, Spain, May 2016.

[40] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, p. 93, 2008.

[41] D. Dua and C. Graff, *UCI Machine Learning Repository*, School of Information and Computer Science, University of California, Irvine, CA, USA, 2017, http://archive.ics.uci.edu/ml.

[42] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using Gaussian fields and harmonic functions," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 912–919, Washington, DC, USA, August 2003.

[43] T. N. Kipf and M. Welling, "Variational graph autoencoders," 2016, https://arxiv.org/abs/1611.07308.

[44] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," 2014, https://arxiv.org/abs/1412.6980.

[45] A. Paszke, S. Gross, S. Chintala et al., "Automatic differentiation in pytorch," in *Proceedings of the 31st Conference on Neural Information Processing Systems*, Long Beach, CA, USA, December 2017.

[46] F. Pedregosa, G. Varoquaux, A. Gramfort et al., "Scikit-learn: machine learning in python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.