ATPGNN: Reconstruction of Neighborhood in Graph Neural Networks With Attention-Based Topological Patterns

KEHAO WANG1, (Member, IEEE), HANTAO QIAN1, XUMING ZENG2, MOZI CHEN2, (Member, IEEE), KEZHONG LIU2, (Member, IEEE), KAI ZHENG2, PAN ZHOU3, (Senior Member, IEEE), AND DAPENG WU4, (Fellow, IEEE)

1School of Information Engineering, Wuhan University of Technology, Wuhan 430070, China
2School of Navigation, Wuhan University of Technology, Wuhan 430063, China
3Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China
4Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, USA

Corresponding author: Xuming Zeng (zengxvming@whut.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61911540481, Grant 52031009, and Grant 61972448.

ABSTRACT Graph Neural Networks (GNNs) have been applied in many fields of semi-supervised node classification for non-Euclidean data. However, some GNNs cannot make good use of positive information brought by nodes which are far away from each central node for aggregation operations. These remote nodes with positive information can enhance the representation of the central node. Some GNNs also ignore rich structure information around each central node’s surroundings or entire network. Besides, most of GNNs have a fixed architecture and cannot change their components to adapt to different tasks. In this article, we propose a semi-supervised learning platform ATPGNN with three variable components to overcome the above shortcomings. This novel model can fully adapt to different tasks by changing its components and support inductive learning. The key idea is that we first create a high-order topology graph, which is from similarity of node structure information. Specifically, we reconstruct the relationships between nodes in a potential space obtained by network embedding in graph. Second, we introduce graph representation learning methods to extract representation information of remote nodes on the high-order topology graph. Third, we use some network embedding methods to get graph structure information of each node. Finally, we combine the representation information of remote nodes, graph structure information and feature for each node by attention mechanism, and apply them to learning node representation in graph. Extensive experiments on real attributed networks demonstrate the superiority of the proposed model against traditional GNNs.

INDEX TERMS Semi-supervised node classification, GNNs, network embedding.

I. INTRODUCTION

In past few years, Convolutional Neural Networks (CNNs) have developed rapidly and achieved great success. However, the pixels in image or video data processed by CNNs are arranged into a very neat matrix, i.e., Euclidean data. In real world, such as social network, citation network, molecular structure and so on, most of data representation is not simple arrangement of sequence or plane, but more complex graph structure, which is non-Euclidean data. Now popular deep learning based GNNs [1], [2] are specially used to process such graph data composed of nodes and edges connecting nodes. In recent years, GNNs have been widely studied for node classification, link prediction and other tasks. By mining link information of nodes in graph data, as well as the attributes of nodes, GNNs and their variants perform better on benchmark datasets.

GNNs and their variants can be divided into spectral domain methods and spatial domain methods. The parameters that need to be learned in the spectrum based GNNs
methods are all related to the eigenvectors and eigenvalues of Laplacian matrix, that is, depending on the structure of graph. For example, [3] extended the operation of CNNs to signals defined on more general domains by using the characteristic decomposition of graph Laplacian, and proposed a neural network similar to CNNs in Fourier domain. In order to perform convolution operation by fast local spectral filtering method like CNNs, [4] used recursive Chebyshev polynomials to approximate the filter, and proposed a complete convolution pooling operation process. Reference [5] further proposed a semi-supervised learning method–Graph Convolutional Networks (GCN) on graph data. The neighbors of central nodes were weighted according to their degrees in [5], and the authors used a simplified spectral convolution operation based on the first-order approximation of spectral filter. For GNNs based on spatial domain, convolution operation is directly defined on the link relationships between nodes. For example, [6] built a feature representation for each node by scanning a diffusion process which stacked the features of different sampling distances into a matrix. Reference [7] mainly introduced an inductive learning method on graph, which generated node representation by sampling and aggregating the characteristics of neighborhood nodes of each node. Reference [8] proposed a new neural network model–Graph Attention Networks (GAT) based on attention mechanism. By introducing self-attention mechanism into GNNs, we can solve the problem that the weights of neighbor nodes are the same in the process of neighborhood aggregation in GCN.

Although the existing GNNs have achieved great success in a variety of scenarios, there still exist many obvious defects. First, the spectrum based GNNs learn new representation of each node by taking the neighborhood of each node as a whole, ignoring the subtle difference between different parts in the neighborhood, such as GCN. This learning operation is equivalent to smoothing the representations of nodes in the neighborhood and using Fully Connected Networks (FCN) to process them, so the spectrum based GNNs have poor flexibility, and do not fully use the potential of network topology. Therefore, the spectrum based GNNs are conductive and have poor scalability. The spatial based GNNs usually do not use the structure of graph when learning the new representation of each node, but only use the relationship between adjacent nodes. For example, GAT uses masked graph attention, and the operation of attention mechanism is only performed on neighbor nodes.

Second, GNNs cannot make good use of the positive information brought by the nodes which are far away from the central nodes to be aggregated, because these GNNs have no aggregation ability for multi-hop neighbors. When considering high-order neighborhood, if we simply increase the number of layers of the model to expand the range of aggregation neighborhood of each node, due to the problem of vanishing gradient, each node will fuse many negative information unrelated to the node itself. So the representation of each node will tend to be consistent, which is closely related to the excessive smoothing of GNNs [9]. As a result, most of the advanced GNNs models are no more than three or four layers.

Third, GNNs only use various aggregation operations to update the node representation in the first-order neighborhood of each node. However, each node has rich structure information around its surroundings or entire network in graph. GNNs could not mine and utilize this kind of structure information well. We believe that if the graph topology details are included in the training process of GNNs, the overall performance of the model will be greatly improved. Fourth, because the architectures of GNNs are fixed, they cannot adapt to different tasks by changing the components of the model for graph datasets with different sizes and structures. Therefore, we must consider the robustness of the model for graph datasets with different attributes.

In view of the above limitations, we propose a new model–ATPGNN, which makes full use of the topology information of graph datasets, mines the nodes in the high-order neighborhood that provide positive information for the aggregation of central nodes, and uses these nodes to expand the aggregation neighborhood of each node. Our model has three variable components, which can use different methods to improve the overall performance on various graph datasets with different topological patterns and sizes. Furthermore, our model is an end-to-end deep model, which supports inductive learning. Specifically, we first map the graph data to a potential space by network embedding methods, and then build the topology neighborhood of each node according to a certain geometric relationship, and reconstruct a high-order topology graph with structure information in the potential space. Through the above process, the remote nodes with similar structure information are contextualized in the topology neighborhood of each node. Next, we design a filter to select the nodes in the topology neighborhood, which improves the quality of representation information of nodes in the topology neighborhood according to feature smoothness and label smoothness metrics. Then, the aggregation operation is performed to produce a stable representation for each node in the topology neighborhood. Furthermore, we use some network embedding methods to get graph structure information of each node. The aggregation operation and filter operation are performed alternately. Finally, the attention mechanism is applied to the node embedding representation, the stable representation obtained by aggregating the topology neighbors, and the node feature to update the representation of each node. In our model, the three variable components are the network embedding method to get the high-order topology graph, the aggregation method in the topology neighborhood and the network embedding representations of nodes, which can be changed according to actual situation.

Our main contributions are summarized as follows:

• A New GNNs Design Scheme. By analyzing some existing GNNs and their variants, we summarize the defects of these models and classify them, and propose a new design scheme to guide the development of GNNs in the future.
• The Full Use of Graph Structure Information. We not only apply the abundant graph structure information around nodes to the aggregation operation in GNNs, but also reconstruct a high-order topology graph to obtain the remote high-order similar nodes' aggregating representation for each node by using the graph structure information. We combine these information by attention mechanism to enhance node representation learning.

• Semi-Supervised Classification Platform with Variable Components. We modularize the methods of extracting graph structure information and aggregating neighbors on the high-order topology graph, and propose a semi-supervised classification learning platform--ATPGNN with three variable components. We can change some components of the platform to adapt to different tasks perfectly. In addition, this platform supports inductive learning.

• Several Model Variants Based on the Proposed Platform. By changing the variable components of ATPGNN, we can get a lot of model variants. In this article, we propose four typical models. Extensive experiments have been carried out on several datasets to show the good performance of our four typical models.

The rest of this article can be divided into the following seven parts. We discuss related work in Section II. We introduce the definition of GNNs, analyze the limitations of some classical GNN variants and propose a new design scheme of GNNs in Section III. We introduce the model architecture of ATPGNN in Section IV. We introduce the overall structure and algorithm description of the model in Section V. We analyze the advantages of the model in Section VI. Extensive experiments are carried out in Section VII. We summarize the contribution of our work in Section VIII.

II. RELATED WORK
Recently, more and more applications need to process non-Euclidean data, which cannot be reliably processed by CNNs. GNNs can run directly on non-Euclidean data, so GNNs have achieved great success in recent years. They have been widely used to predict individual relationships in social networks [10], simulate chemical molecular structures to understand their biological activities [11], [12], and implement product recommendation and semantic search in knowledge graphs [13], [14]. Among them, semi-supervised node classification on graph is the most interesting and hot topic. The purpose of semi-supervised node classification is to infer the classes of remaining unlabeled nodes by using prior knowledge of graph.

Learning accurate low dimensional network embedding for each node is a key task in graph representation learning. The network embedding should capture graph topology, node to node relationship and other related information about graph, subgraph and node, and retain original network information as much as possible. In recent years, the development of network embedding method is very fast. Up to now, new algorithms emerge endlessly, among which some classical methods are based on Random Walk. For example, DeepWalk [15] is the most influential method in the early stage. Its main idea is to use Random Walk to sample nodes in graph, and then input co-occurrence relationships between nodes as training samples into Word2vec [16] to learn the vector representations of nodes. Node2vec [17] goes further on the basis of DeepWalk. By adjusting the weight of Random Walk, Node2vec makes the result of graph embedding trade-off between homogeneity and structural equivalence. For the process of Random Walk, in order to make the node embedding to express network structure, we need to make the process more inclined to Breadth First Search (BFS); on the other hand, in order to grasp the homogeneity of network, we need to make the process more inclined to Depth First Search (DFS). Node2vec controls the tendency of BFS and DFS through the jump probability between nodes. Struc2vec [18] uses a hierarchy to measure node similarity at different scales, and constructs a multilayer graph to encode structural similarities and generate structural context for nodes. It can better capture the structural similarity between two nodes which are far away from each other. GraphWave [19] is an extensible unsupervised method that represents each node’s network neighborhood via a low-dimensional embedding by leveraging heat wavelet diffusion patterns, instead of training on hand-selected features.

How to use these low dimensional network embedding vectors in GNNs, that is to say, how to make full use of network topology information to improve the performance of GNNs is a huge challenge. So far, there are many works in this field. Reference [20] proposed a novel Topology Optimization based Graph Convolutional Networks (TO-GCN) to fully utilize potential information by jointly refining the network topology and learning the parameters of the FCN and made GCN filter more flexible and learnable. Reference [21] proposed GraLSP, a new GNN model which explicitly incorporated local structural patterns into the neighborhood aggregation through random anonymous walks. With the adequate leverage of structural patterns, this model captured similarities between structures. Reference [22] proposed a Graph Inference Learning (GIL) framework to boost the performance of node classification by learning the inference of node labels on graph topology. This model formally defined a structure relation by encapsulating node attributes, between-node paths and local topological structures together, which can make inference conveniently deduced from one node to another node.

Although the above works can make use of local topology information, they fail to consider the impact of some nodes with positive information in the high-order neighborhood on the central node aggregation. The following works are about some solutions, such as expanding neighborhood size or increasing the number of network layers. Reference [23] introduced Graph RESidual NETwork (GRESNET) framework in their model, which created extensively connected highways to involve nodes’ raw features or intermediate representations throughout the graph for all the model layers.
Reference [24] proposed a new model, MixHop, that can learn a general class of neighborhood mixing relationships, including difference operators, by repeatedly mixing feature representations of neighbors at various distances. Reference [25] borrowed concepts from CNNs, specifically residual/dense connections and dilated convolutions, and adapted them to GCN architectures to train very deep GCN.

The above three works only expand the scope of each node’s receptive field of aggregation operation, without considering the quality of information carried by the nodes in the high-order neighborhood. At present, there are some works on how to select nodes in the neighborhood or the high-order neighborhood that are beneficial to the aggregation of central nodes. Reference [26] introduced a context-surrounding GNNs framework and proposed two smoothness metrics obtained from node feature and node label to measure the quantity and quality of node representation information in the neighborhood. The smoothness metrics were applied to the model to selectively aggregate node representations in the neighborhood to amplify useful information and reduce negative disturbance, but it is only applicable to one-hop neighbors. Reference [27] made good use of structural neighborhood continuous space underlying graph and captured long-range dependencies in disassortative graphs. However, the model is transductive, and lack of flexibility and scalability. Reference [28] contextualized each node with a weighted, learnable receptive field encoding rich and diverse local graph structures. But the model only performs best in two-hops neighbors. Although the above works have their shortcomings, they also brought us a lot of inspiration.

III. BACKGROUND AND MOTIVATION

In this section, we give a brief introduction to the notations and definitions of GNNs, and introduce some typical models and analyze their limitations.

A. NOTATIONS AND DEFINITIONS

Define a graph as $G=(\mathcal{V},\mathcal{E},\mathcal{X})$ with the set of $N$ nodes $\mathcal{V} = \{v_1,v_2,\ldots,v_N\}$, the set of $M$ edges $\mathcal{E} = \{e_1,e_2,\ldots,e_M\}$ and the set of representations $\mathcal{X} = \{\tilde{x}_1,\tilde{x}_2,\ldots,\tilde{x}_N\}$. Each node $v_n$ in the graph $G$ is represented as a vector $\tilde{x}_n \in \mathbb{R}^{1 \times S}$, which is the attributes of node $v_n$ and consists of $S$-dimensional vector. All the representations of nodes form a matrix $X \in \mathbb{R}^{N \times S}$. The network topology is represented by an adjacency matrix $A = \{a_{ij}\} \in \mathbb{R}^{N \times N}$, which is a binary one. If node $v_i$ is connected to node $v_j$, then $a_{ij} = 1$, otherwise $a_{ij} = 0$. $D = \text{diag}(d_1,d_2,\ldots,d_N)$ is a degree matrix, where $d_n = \sum_j a_{nj}$ is the degree of node $v_n$. $(\cdot)^T$ is the transpose of a vector or a matrix.

B. GRAPH NEURAL NETWORKS

For each node in the graph, the goal of GNNs is to learn a stable node representation which contains the information of neighbors. The representation can be applied to many downstream tasks, such as node classification, graph classification, link prediction and so on. In a GNN with $L$ layers, the input of its $l$-th layer ($l=1,\ldots,L$) is a feature vector $\tilde{h}_l^{l-1}$ for each node $v_i$, which is also the output of its $(l-1)$-th layer. The input $\tilde{h}_l^0$ of the first layer is the node feature $\tilde{x}_i^T$. For each node $v_i$, the $l$-th layer updates its output $\tilde{h}_l^i$ by aggregating feature vectors from the neighborhood $N_i$ of node $v_i$, possibly using a different weight $w_{ij}$ for neighbor $v_j$ in some special GNNs methods. By repeating this aggregation operation, GNNs can capture the node feature of the $L$-th-order neighborhood for each node $v_i$, and finally generate a stable representation $\tilde{h}_L^i$ for each node $v_i$. The general GNNs framework can be defined as follows:

$$\tilde{h}_L^i = \text{AGGREGATE}^{(l)}(\{\tilde{h}_l^{l-1},w_{ij} : v_j \in N_i\})$$

$$\tilde{h}_l^i = \text{COMBINE}^{(l)}(\tilde{h}_l^{l-1},\tilde{h}_{N_i}^{l-1})$$

where AGGREGATE represents the method of aggregating neighborhood, such as taking the mean or maximum value of neighbor node representation $\tilde{h}_l^{l-1}(v_j \in N_i)$, and COMBINE represents the method of merging $\tilde{h}_l^{l-1}$ and the node representation $\tilde{h}_{N_i}^{l-1}$, such as concatenation and so on.

C. GRAPH CONVOLUTIONAL NETWORKS

GCN is a variant of GNNs in spectral domain. In reference [3], the authors introduced convolution operation into GNNs and directly applied convolution operation to the spectrum of graph. The network framework in [3] can be defined as follows:

$$\tilde{h}_1^i = g_0 \ast \tilde{h}_1^{l-1} = Ug_\theta(\lambda)U^T\tilde{h}_1^{l-1}$$

where $g_\theta(\lambda)$ is convolution kernel, $U$ and $\lambda$ are the singular vectors and singular values obtained by spectral decomposition of graph Laplacian matrix $L = D^{-1/2}(D-A)D^{-1/2}$. However, the disadvantage of this method is that its time complexity is $O(N^2)$ and the convolution kernel does not have the characteristic of spatial localization. Considering these, an improved model proposed in [4] solves the above shortcomings by using the $K$-th order Chebyshev polynomials ($g_\theta(\lambda) = \sum_{k=0}^K \theta_k T_k(L)$) instead of the convolution kernel, where $T_k$ is Chebyshev polynomials, $\theta_k$ is Chebyshev coefficient. GCN is actually a special case derived from the first-order Chebyshev polynomials and the maximum singular value to be 2. The GCN framework can be defined as follows:

$$\tilde{h}_1^i = g_0 \ast \tilde{h}_1^{l-1} = \theta(I + D^{-1}\tilde{A}D^{-1/2})\tilde{h}_1^{l-1}$$

where $\theta$ is spectral filter, and $I$ is the identity matrix. Eq.(4) can be written in matrix form as follows:

$$H = \sigma(D^{-1/2}\tilde{A}D^{-1/2}XW)$$

where $\tilde{A} = A + I$, $\tilde{D}_{nn} = \sum_j \tilde{A}_{nj}$, $X \in \mathbb{R}^{N \times S}$ is the node feature matrix, $W \in \mathbb{R}^{S \times C}$ is learnable parameter matrix, which consists of $C$ spectral filters $\theta$, and $\sigma$ is activation function.
TABLE 1. Different variants of graph neural networks.

<table>
<thead>
<tr>
<th>Models</th>
<th>Aggregation and combination functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>General GNNs framework</td>
<td>$\tilde{h}^l_i = \text{COMBINING}^{(l)} \left( { \tilde{h}^{l-1}_i, \text{AGGREGATE}^{(l)} \left( { \tilde{h}^{l-1}<em>j, w^{l}</em>{ij} : v_j \in \mathcal{N}_i } } \right) \right)$</td>
</tr>
<tr>
<td>GCN</td>
<td>$\tilde{h}^l_i = \sigma \left( \sum_{v_j \in \mathcal{N}_i \cup {v_i}} \frac{1}{\sqrt{</td>
</tr>
<tr>
<td>GAT</td>
<td>$\tilde{h}^l_i = \sigma \left( \sum_{v_j \in \mathcal{N}<em>i \cup {v_i}} w^{l}</em>{ij} \cdot W^{l-1} \cdot \tilde{h}^{l-1}_j \right)$</td>
</tr>
<tr>
<td>ADSF</td>
<td>$\tilde{h}^l_i = \sigma \left( \sum_{v_j \in \mathcal{N}<em>i \cup {v_i}} w^{l}</em>{ij} \cdot W^{l-1} \cdot \tilde{h}^{l-1}_j \right)$</td>
</tr>
</tbody>
</table>

D. GRAPH ATTENTION NETWORKS

In reference [8], the authors proposed GAT to solve two problems in GCN:

- Unable to complete the inductive tasks, such as processing dynamic graphs.
- Unable to assign different weights to different neighbors of each node.

The core idea of GAT is the introduction of attention mechanism. The coefficient of attention mechanism between node $v_i$ and node $v_j$ can be defined as follows:

$$w^{l}_{ij} = \frac{\exp \left( \delta \left( W^l \tilde{h}_i^{l} || W^l \tilde{h}_j^{l} \right) \right)}{\sum_{v_k \in \mathcal{N}_i} \exp \left( \delta \left( W^l \tilde{h}_i^{l} || W^l \tilde{h}_k^{l} \right) \right)}$$  \hspace{1cm} (6)

where $w^{l}_{ij}$ is the attention coefficient of the $l$-th layer, $||$ is the concat operation, $W^l$ is the learnable parameter matrix of the $l$-th layer, and $\delta(x, y)$ maps the high-dimensional features to a real number, such as single-layer feed forward neural network.

E. ADAPTIVE STRUCTURAL FINGERPRINTS FOR GAT

In reference [8], the authors brought some local graph structure information into attention mechanism.

$$m^{l}_{ij} = \frac{\exp \left( \text{Jaccard}(\tilde{r}_i, \tilde{r}_j) \right)}{\sum_{\tilde{r}_k \in \mathcal{N}_i} \exp \left( \text{Jaccard}(\tilde{r}_i, \tilde{r}_k) \right)}$$  \hspace{1cm} (7)

$$w^{l(t)}_{ij} = \varphi \left( w^{l}_{ij}, m^{l}_{ij} \right)$$  \hspace{1cm} (8)

where $\tilde{r}_i$ is the convergence solution obtained by Random Walk with Restart (RWR) [29] for node $v_i$, which reflects the local structure around the node $v_i$. $\text{Jaccard}(x, y)$ is the Jaccard similarity [30], $m^{l}_{ij}$ is the attention coefficient which contains network structure information, $\varphi(x, y)$ is a function that combines $w^{l}_{ij}$ and $m^{l}_{ij}$, and $w^{l(t)}_{ij}$ is the final attention coefficient.

F. MOTIVATION

After summarizing the existing methods, we can list the core frameworks of these methods, as shown in TABLE 1.

According to the framework of GCN (5), we can find its graph convolution operation—Laplacian operator $\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$ is equivalent to Laplacian smoothing applied to the first-order neighborhood. In TABLE 1, comparing GCN with the General GNNs framework, we can see that the neighbor weight $w^{l}_{ij}$ in GNNs is embodied as $\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$ in GCN, that is, $1/\sqrt{(|\mathcal{N}_i| + 1) \times (|\mathcal{N}_j| + 1)}$. This shows that the graph convolution operation essentially uses the weight $1/\sqrt{(|\mathcal{N}_i| + 1) \times (|\mathcal{N}_j| + 1)}$ to average the node representation in the first-order neighborhood. From the above analysis, for each node $v_j$ in $\mathcal{N}_i$, the weights of its neighbor $w^{l}_{ij}$ are equal. The training process of GCN depends on the specific graph structure (i.e. Laplacian matrix, once the node changes, the Laplacian matrix will change), and the testing process can also only be conducted on the same graph. Therefore, GCN is transitive and cannot assign different weights to different neighbor nodes.

It can be seen from (6) that GAT is equivalent to calculating neighbor weights (i.e. edge weight) according to the node representation similarity between connected nodes. The GAT method does not involve Laplacian matrix in the training process; instead, it trains the AGGREGARE mode between the central node and its neighbor nodes. Therefore, GAT can be applied to the task of changing the number or features of nodes or edges in some datasets, so it belongs to inductive learning. From the above analysis, we can see that GAT does not consider graph structure at all.

ADSF takes into account the node embedding generated by Random Walk in the range of two-hops neighbors for each node. However, neighborhood aggregation is only carried out on the first-order neighborhood.

Based on our discussion on prior approaches (GCN, GAT, and ADSF), we propose a design scheme guiding the development of GNNs methods. It can also better evaluate the advantages of our model ATPGNN in comparison with other methods.

TABLE 2 summarizes ATPGNN and some existing methods for the design scheme. We give the following explanations to all indexes of the design scheme:

<table>
<thead>
<tr>
<th></th>
<th>ATPGNN</th>
<th>ADSF</th>
<th>GAT</th>
<th>GCN</th>
</tr>
</thead>
<tbody>
<tr>
<td>node feature</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>inductive learning</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>local structure</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>high-order neighborhood</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>overall structure</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>flexibility</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

1 Only the two-hops neighbors are used in node embedding.
- **node feature.** In a graph, the node feature is often the most basic information, which represents the attributes of each node. The model should make full use of the node feature information.
- **inductive learning.** The trained model should be adapted to the changed test dataset. This requires that the nodes and edges in graph cannot be trained as a whole in the model.
- **local structure.** Local graph structure information is also an important part. It can reflect the node distribution details in the local neighborhood, so the model should be able to use this information.
- **high-order neighborhood.** There are also some nodes (i.e., nodes with similar structure or similar features to the central nodes) in the high-order neighborhood that are helpful to the aggregation of central nodes. The model should not be limited to the first-order neighborhood aggregation.
- **overall structure.** The overall structure of the graph is also available information for each node, which should be taken account in central node aggregation.
- **flexibility.** In order to adapt to different datasets or tasks, the model needs to have flexible components to fit these complex and changeable situations.

A model with excellent performance should meet the above design requirements as much as possible.

**IV. MODEL ARCHITECTURE OF ATPGNN**

In section III, we analyze that the existing models can only meet some indexes of the GNNs design scheme. This prompts us to design a new model on the basis of GNNs framework to meet all the indexes of the design scheme. In this section, we propose several definitions and variable components of the general framework of ATPGNN.

**A. BUILDING TOPOLOGY NEIGHBORHOOD**

Through the analysis in subsection III-F, we know that those remote nodes with similar topology structure as central node can improve the performance of the aggregation operating on central node. Making good use of these node information can satisfy the requirement of high-order neighborhood in the design scheme. So we should design a method to collect these remote nodes with similar topology structure. How do we find these remote nodes? First, we use node embedding methods to extract the abstract graph structure information into a concrete vector or matrix for each node in the graph. The goal of the node embedding methods is to find the low-dimensional vector representation of each node in a high-dimensional graph. In a graph \( G = (V, E, \mathcal{X}) \), the result of node embedding is the mapping \( t_i \) of each node \( v_i \).

\[
f: v_i \rightarrow t_i \in \mathbb{R}^{1 \times d} \quad \forall v_i \in V
\]

There are many methods of node embedding, such as Node2vec, DeepWalk, SDNE [31], GraphWave and so on. Because we want to find remote nodes with similar structure, we need to use a method to capture the global similarity in the graph, which takes the overall structure in the design scheme into account, rather than the method to capture the close relationships between nodes in local neighbors. There are many node embedding methods to extract the global network structure of nodes, such as GraphWave, HARP [32], etc. Using these methods, we can get the concrete representation of structure information of each node. We express this node embedding representation as vector \( \tilde{t}_i \) for node \( v_i \), which can be used to find the similar nodes which are far away from each other.

**Definition 1 (Topology Neighborhood):** For each central node \( v_i \), the topology neighborhood of \( v_i \) is a set \( \mathcal{T}_i \) of which each node embedding vector \( \tilde{t}_j (v_j \in \mathcal{T}_i) \) is similar with central node embedding vector \( \tilde{t}_i \) in a certain range.

Next, we quantify the similarity between different node embeddings. There are many quantifying methods, for example, the simplest method is to calculate the vector distance \( d_{ij} \) between central node embedding vector \( \tilde{t}_i \) and other node embedding vector \( \tilde{t}_j \) in the graph, such as Euclidean distance, cosine distance, etc. Comparing the distance \( d_{ij} \) with an introduced set threshold \( \varepsilon \), the set of nodes whose distance \( d_{ij} \) is less than the threshold \( \varepsilon \) is the topology neighborhood \( \mathcal{T}_i \) of the central node \( v_i \).

\[
\mathcal{T}_i = \{ v_j \mid v_j \in V, d(\tilde{t}_i, \tilde{t}_j) < \varepsilon \}
\]

In this article, we design a ingenious method to get the topology neighborhood of each node inspired by [26]. Specifically, we first get the first \( n \) nodes whose node embedding vectors are most similar to the central node embedding vector according to the average degree \( n \) of each node in the graph. These first \( n \) nodes constitute the initial topology neighborhood of each node. For example, in Cora dataset, the average degree of each node is \( n = 5 \), and then we find the first five nodes with the smallest distance from the central node to form the initial topology neighborhood of each node.

\[
\mathcal{T}_i = \{ v_j \mid v_j \in V, \text{TOP}_n \{ d(\tilde{t}_i, \tilde{t}_j) \} \}
\]

where \( \text{TOP}_n \{ \} \) finds the first \( n \) nodes whose node embedding vectors are most similar to the central node, \( \mathcal{T}_i \) is the initial unhandled topology neighborhood. From the link relationships of neighbor nodes in each topology neighborhood, we can get a new adjacency matrix \( A_T = [b_{ij}] \in \mathbb{R}^{N \times N} \), where \( b_{ij} = 1 \) represents node \( v_j \in \mathcal{T}_i \), otherwise \( b_{ij} = 0 \). Further, we can obtain a new high-order topology graph \( \mathcal{G}_T = (V, \mathcal{E}_T, \mathcal{X}_T) \) with the node set \( V = \{v_1, v_2, \ldots, v_N\} \) and the edge set \( \mathcal{E}_T = \{e_1, e_2, \ldots, e_T\} \) which is derived from \( A_T \).

Next, we apply the the weight \( \mu_{i,j}^{l-1} \) of the neural network proposed in [26] to filter the nodes \( v_j \in \mathcal{T}_i \) by multiplying the weight \( \mu_{i,j}^{l-1} \) such that we can obtain a updated \( \mathcal{T}_i \).

**B. AGGREGATION METHOD OF TOPOLOGY NEIGHBORHOOD**

Through the discussion of the previous section, we design a ingenious method to obtain the topology neighborhood of
each node. In this part, we will discuss how to aggregate these topology neighborhoods for each node in the high-order topology graph $G_T$.

**Definition 2 (Topology Neighborhood Aggregation):** For each node $v_i$ in the high-order topology graph $G_T$, all the representation vectors $\tilde{\vec{v}}_j$ ($v_j \in T_i$) of these nodes in the topology neighborhood $\tilde{T}_i$ of node $v_i$ are aggregated to update the representation vector $\bar{\vec{v}}_i$ of the node $v_i$ iteratively. In the $l$-th iteration ($\bar{\vec{v}}_i^l = \tilde{\vec{v}}_j^l$):

$$\bar{\vec{v}}_i^l = \text{AGGREGATE}^{(l)} \left( \{\tilde{\vec{v}}_j^{l-1} : v_j \in \tilde{T}_i \cup \{v_i\}\} \right) \quad (12)$$

There are many methods to aggregate the node representations in the topology neighborhood. The most primitive method is Message Passing. Message Passing algorithm is a model acting on graph, in which each node and each edge has its own vector. We initialize these vectors as input features of nodes and edges, and then each node repeatedly passes its current representation vector to its neighbor nodes, and aggregates the messages from its neighbors. After passing through a certain number of steps, the model outputs a stable representation for each node that can be used for downstream tasks.

$$\bar{\vec{m}}_i^l = \sum_{v_j \in \tilde{T}_i} M_{l-1} \left( \bar{\vec{v}}_i^{l-1}, \tilde{\vec{v}}_j^{l-1}, e_{ij} \right) \quad (13)$$

$$\bar{\vec{v}}_i^l = U_{l-1} \left( \bar{\vec{v}}_i^{l-1}, \bar{\vec{m}}_i^l \right) \quad (14)$$

where $M_{l-1}$ is the message function, $U_{l-1}$ is the node update function, and $e_{ij}$ is the feature of the edge between $v_i$ and $v_j$ (In this article, ignore the edge features, and if there is a link between $v_i$ and $v_j$, $e_{ij} = 1$, otherwise, $e_{ij} = 0$).

There are also some common aggregation methods, such as mean-pooling aggregation:

$$\bar{\vec{v}}_i^l = \text{mean} \left( \{\tilde{\vec{v}}_j^{l-1} : v_j \in \tilde{T}_i \cup \{v_i\}\} \right) \quad (15)$$

and max-pooling aggregation:

$$\bar{\vec{v}}_i^l = \text{max} \left( \{\tilde{\vec{v}}_j^{l-1} : v_j \in \tilde{T}_i \cup \{v_i\}\} \right) \quad (16)$$

In our approach, we use GNNs and variants to aggregate nodes in topology neighborhood. As neural networks, they can better fit the graph with complex pattern information and extract the information of remote nodes in the high-order topology graph. Many GNNs models have been proposed and several examples are given in TABLE 1.

Before aggregating, we multiply the node representation $\tilde{\vec{v}}_j^{l-1}$ in the topology neighborhood $\tilde{T}_i$ for central node $v_i$ by the weight $\mu_{i,j}^{l-1}$ mentioned in subsection IV-A.

$$\bar{\vec{v}}_j^{l-1} \leftarrow \mu_{i,j}^{l-1} \tilde{\vec{v}}_j^{l-1}, \quad v_j \in \tilde{T}_i \quad (17)$$

Then we use GNNs or their variants to aggregate the weighted $\tilde{\vec{v}}_j^{l-1}$ in $\tilde{T}_i$.

$$\bar{\vec{v}}_j^l = \text{AGGREGATE}^{(l)} \left( \{\tilde{\vec{v}}_j^{l-1}, \mu_{i,j}^{l-1} \tilde{\vec{v}}_j^{l-1} : v_j \in \tilde{T}_i\} \right) \quad (18)$$

$$\bar{\vec{v}}_i^l = \text{COMBINE}^{(l)} \left( \bar{\vec{v}}_j^l \right) \quad (19)$$

After aggregating nodes in topology neighborhood, the stable representation $\bar{\vec{v}}_i^l$ ($\bar{\vec{v}}_i^l = \tilde{\vec{v}}_j^l$ if there are $L$ iterations) is a feature representation for each node that contains remote nodes with similar global topology structure in the overall scope of the graph.

**C. METHODS OF NODE EMBEDDING IN NEIGHBORHOOD AGGREGATION**

We not only need to use node embedding method to get the topology neighborhood in subsection IV-A, but also use the node embedding vector $\bar{\vec{v}}_i$ obtained by node embedding method, which extracts the graph topology structure information for each node. How to choose and use node embedding method to get $\bar{\vec{v}}_i$ depends on the attributes of graph data we are going to process and what kind of graph structure representation we want to obtain. Here, we introduce several node embedding methods and analyze their performance.

- **DeepWalk** [15] can be the most famous work of node embedding which is an improved version based on Random Walk. The most important contribution of DeepWalk is that it links graph representation learning with Word2vec, a famous word embedding method in natural language processing. In DeepWalk, the authors transformed the network embedding problem to a word embedding one through the Random Walk algorithm. The extended methods from Random Walk also include Random Walk with Restart (RWR), Node2vec, Struc2vec, and so on.

- In **LINE** [33], the authors did not use Random Walk, but defined two kinds of similarity in graph: first-order neighborhood similarity and second-order one. In SDNE [31], the authors kept these two kinds of similarity in the process of graph representation learning through a specific deep learning method inspired by LINE.

- **GraphWave** [19] is an extensible unsupervised method, which does not need any prior knowledge, and learns the structure information of each node based on the diffusion of spectral wavelets centered on each node. Certainly, there are many other methods not introduced here. The application scenarios of downstream tasks largely determine the method of node embedding that we would adopt. If you are inclined to study the similarity of local neighborhood which reflects the local distribution of neighbors $v_j$ for each node $v_i$ in several-hops neighborhood, you can choose Node2vec, LINE, etc. This satisfies the requirement of **local structure** in the design scheme. If inclined to study global structure similarity which reflects the global structure information and high-order nodes information in graph, you can choose Struc2vec, GraphWave, etc. Moreover, if you need to consider the extra information between nodes and edges in graph, you can choose CANE [34], Trans-net [35], etc. Even you can choose several node embedding methods and combine the node embedding vectors.
V. MODEL ALGORITHM DESCRIPTION

Through the above analysis, we get several basic components for our model and will combine these components to form a basic framework in this section.

For each node \( v_j \) in the graph, we have three different information: the input feature of the node \( x_i \), the structure information representation \( \bar{r}_i \) obtained by node embedding method, and the stable representation \( \bar{s}_i \) containing the information of remote similar nodes obtained by aggregating node representation in the topology neighborhood \( \tilde{T} \). How to use these three different kinds of information for each node and aggregate them in graph? Inspired by attention mechanism, we can get three different attention coefficients for the three different information through calculating additive attention. Then we try to combine these three different attention coefficients based on GAT, so our model is able to make full use of these information. Recently, GAT can handle variable size input by combining GNNs with attention mechanism, and this characteristic of GAT satisfies the requirement of inductive learning in the design scheme.

We first calculate the attention coefficient \( e_{i,j}^l \) in the \( l \)-th layer of the model between the representations of nodes \( \bar{r}_i^l \) and \( \bar{r}_j^l \) (\( \bar{r}_i^l = \bar{s}_i^T, \bar{r}_j^l = \bar{s}_j^T \)).

\[
e_{i,j}^l = \delta \left( \left[ W^l \bar{r}_i^l || W^l \bar{r}_j^l \right] \right), \quad v_j \in N_i
\]  

(20)

In the above formula, \( W^l \) is the learnable matrix of the \( l \)-th layer, which increases the dimension of node features by linear mapping. \( || \) concatenates the transformed representations of nodes \( v_i \) and \( v_j \). Finally, \( \delta(\cdot) \) maps the concatenated high-dimensional features to a real number.

In subsection IV-B, we obtain the stable representation \( \bar{s}_i \) for each node \( v_i \) by aggregating the representations of remote similar nodes in the topology neighborhood \( \tilde{T}_i \). In the same way, we will get the attention coefficient \( f_{i,j}^l \) which contains the information of remote nodes with similar structure.

\[
f_{i,j}^l = \delta \left( \left[ W^l \bar{p}_i^l || W^l \bar{p}_j^l \right] \right), \quad v_j \in N_i
\]  

(21)

where \( \bar{p}_i^l \) is the representation containing remote nodes information with similar structure of the \( l \)-th layer for node \( v_i \) (\( \bar{p}_i^l = \bar{s}_i^T \)).

In subsection IV-C, for each node \( v_i \), we obtain the embedding representation \( \bar{r}_i \) of structural information by node embedding. Next, we use the attention mechanism to get the attention coefficient \( g_{i,j}^l \), which is the weight of the difference in the topological structures between the central node and its neighbors between the node embedding vectors \( \bar{r}_i \) and \( \bar{r}_j \). As in the previous method, we get the attention coefficient \( g_{i,j}^l \) in the neighborhood \( N_i \).

\[
g_{i,j}^l = \delta \left( \left[ W^l \bar{q}_i^l || W^l \bar{q}_j^l \right] \right), \quad v_j \in N_i
\]  

(22)

where \( \bar{q}_i^l \) is the representation of topological information of the \( l \)-th layer for node \( v_i \) (\( \bar{q}_i^l = \bar{r}_i^T \)). Besides using neural network to get the attention coefficient \( g_{i,j}^l \), we can also take the similarity between node embedding vectors \( \bar{r}_i \) and \( \bar{r}_j \) as the attention coefficient \( g_{i,j}^l \). For example, we can adopt the Jacard similarity to get the attention coefficient \( g_{i,j}^l \).

\[
g_{i,j}^l = \frac{|\bar{r}_i \cap \bar{r}_j|}{|\bar{r}_i \cup \bar{r}_j|} = \frac{|\bar{r}_i \cap \bar{r}_j|}{|\bar{r}_i| + |\bar{r}_j| - |\bar{r}_i \cap \bar{r}_j|}, \quad v_j \in N_i
\]  

(23)

In most cases, we use the Generalized Jaccard coefficient to calculate the similarity.

\[
g_{i,j}^l = \sum_{v_k \in (V_i \cup V_j)} \min \left( \frac{|\bar{r}_i \cap \bar{r}_j|}{|\bar{r}_i| + |\bar{r}_j| - |\bar{r}_i \cap \bar{r}_j|}, \frac{|\bar{r}_i \cap \bar{r}_j|}{|\bar{r}_i| + |\bar{r}_j| - |\bar{r}_i \cap \bar{r}_j|} \right) \quad v_j \in N_i
\]  

(24)

where \( V_i \) is the neighborhood range of node embedding method sampling for each node \( v_i \), which is the subgraph. If the range of sampling is the \( k \)-th-order neighborhood, the subgraph is composed of all the nodes and edges within the \( k \)-hops neighbors of \( v_i \), similarly, \( V_j \), \( \bar{r}_i \) is the \( p \)-th element of the vector \( \bar{r}_i \), similarly, \( \bar{r}_j \).

Different from the previous one, the node embedding generated by some methods can sometimes be a matrix. If the node embedding is in the form of matrix \( T_i \), there are many methods to calculate the matrix attention coefficient. The method of neural network is as follows:

\[
\bar{r}_i = \text{reshape}(T_i)
\]

(25)

\[
\bar{r}_j = \text{reshape}(T_j)
\]

(26)

\[
g_{i,j}^l = \delta \left( \left[ W^l \bar{q}_i^l || W^l \bar{q}_j^l \right] \right), \quad v_j \in N_i
\]  

(27)

to be specific, \( \text{reshape}(\cdot) \) convert the matrices \( T_i \) and \( T_j \) into the vectors \( \bar{r}_i \) and \( \bar{r}_j \), and then we use the method mentioned in equation (22).

We can also take the similarity between matrices \( T_i \) and \( T_j \) as the attention coefficient \( g_{i,j}^l \). For example, we can adopt the norm of the distance between \( T_i \) and \( T_j \) to get the attention coefficient \( g_{i,j}^l \).

\[
d_1(T_i, T_j) = \sum_{m=1}^{M} \sum_{n=1}^{N} |T_{i,m,n} - \bar{T}_{j,m,n}|
\]  

(28)

where \( T_{i,m,n} \) is the element of the \( m \)-th row and \( n \)-th column of the matrix \( T_i \), similarly, \( T_j \), and \( d_1 \) represents the 1 norm of the distance. We can also use 2 norm or infinite norm and other methods to get the attention coefficient \( g_{i,j}^l \).

Then, we normalize the three attention coefficients mentioned above.

\[
c_{i,j}^l = \frac{\exp \left( \text{LeakyReLU} \left( e_{i,j}^l \right) \right)}{\sum_{v_k \in N_i} \exp \left( \text{LeakyReLU} \left( e_{i,k}^l \right) \right)}
\]  

(29)

\[
\beta_{i,j}^l = \frac{\exp \left( \text{LeakyReLU} \left( f_{i,j}^l \right) \right)}{\sum_{v_k \in N_i} \exp \left( \text{LeakyReLU} \left( f_{i,k}^l \right) \right)}
\]  

(30)

where \( \text{LeakyReLU}(\cdot) \) is an activation function. For the attention coefficient \( g_{i,j}^l \), we have two cases. If the method of neural network is used to calculate the attention coefficient \( g_{i,j}^l \),
the normalization result of $g_{i,j}^l$ is the same as above:

$$\gamma_{i,j}^l = \frac{\exp \left( \text{LeakyReLU} \left( g_{i,j}^l \right) \right)}{\sum_{i \in N_i} \exp \left( \text{LeakyReLU} \left( g_{i,k}^l \right) \right)} \quad (31)$$

If we take the similarity between node embeddings as the attention coefficient $g_{i,j}^l$, we do not need an activation function, then the normalization result of $g_{i,j}^l$ is as follows:

$$\gamma_{i,j}^l = \frac{\exp \left( \left( g_{i,j}^l \right) \right)}{\sum_{i \in N_i} \exp \left( \left( g_{i,k}^l \right) \right)} \quad (32)$$

Next, we combine these three normalized attention coefficients to get the final attention coefficient. Here $b^l(\cdot)$, $c^l(\cdot)$ and $d^l(\cdot)$ are transfer functions.

$$A_{i,j}^l = \frac{b^l \left( \alpha_{i,j}^l \right) \alpha_{i,j}^l + c^l \left( \beta_{i,j}^l \right) \beta_{i,j}^l + d^l \left( \gamma_{i,j}^l \right) \gamma_{i,j}^l}{b^l \left( \alpha_{i,j}^l \right) + c^l \left( \beta_{i,j}^l \right) + d^l \left( \gamma_{i,j}^l \right)} \quad (33)$$

The final joint attention coefficient $A_{i,j}^l$ is used to weight and aggregate the node representations. The output of the $l$-th layer is:

$$\tilde{h}_i^l = \sigma \left( \sum_{j \in N_i} A_{i,j}^{l-1} W^{l-1} \tilde{h}_{j}^{l-1} \right) \quad (34)$$

where $\sigma$ is the activation function. Finally we add the multi-head operation to the output representation above:

$$\tilde{h}_i^l = \frac{1}{K} \sum_{k=1}^{K} A_{i,j}^{k(l-1)} W^{k(l-1)} \tilde{h}_{j}^{k(l-1)} \quad (35)$$

where $||$ is the concat operation, $A_{i,j}^{k(l-1)}$ is the $k$-th attention coefficient, there are $K$ attention mechanisms to consider, and $W^{k(l-1)}$ is the learnable matrix of the $l$-th layer of the $k$-th attention mechanism.

In summary, the above is the overall description of ATPGNN. The time complexity of ATPGNN is $O \left( K(NSS' + 3MS') + \text{aggregation} \right)$, where $K$ is the number of attention mechanisms, $N$ is the number of nodes in the graph, $M$ is the number of edges in the graph, $S$ is the dimension of input node feature, $S'$ is the dimension of output node representation, and aggregation is the time complexity of aggregation operation of topology neighborhood in subsection IV-B. Obviously, the time complexity is on the same order of magnitude as that of GAT.

We depict the general architecture in Fig. 1. First, the network embedding method mining global structure is used to find the remote similar nodes, which constitute a high-order topology graph. Second, aggregation operation is carried out on the topology neighborhood to obtain the convergent representation for each node. Third, the structural information of the node which contains the distribution details of each node’s neighbors in several-hops neighborhood is obtained by extracting graph structure embedding. Finally, we should also make use of the feature of each node. Then, the three features are calculated in the neighborhood to get the three attention coefficients for each node, and the three attention coefficients are fused to get the final attention coefficient, which is the key to weight nodes in every attention network layer. According to the attributes of graph datasets and downstream tasks, we can freely choose the three components in Fig. 1 to form a model suitable for specific scenarios and problems.
VI. ANALYSIS OF THE ADVANTAGES OF ATPGNN

GNNs do not have the ability to aggregate multi-hop neighbors. If we simply increase the number of layers of the model to expand the scope of each node’s receptive field, when considering the interaction between high-order neighbors, each node will fuse a lot of negative information unrelated to the node itself. Their representations will tend to be similar, due to the vanishing gradient problem. Our model overcomes the defect that GNNs cannot use the information of remote nodes because of the vanishing gradient problem. Instead of including all nodes in the graph directly into the aggregation operation, we selectively aggregate remote nodes. As shown in Fig. 2, when two remote nodes have similar structural information, we have reason to believe that they can provide positive information by each other to help the process of neighborhood aggregation. So we use the information of these nodes by constructing the high-order topology graph.

Because the aggregation operation of GNNs for each node is only limited in the first-order neighborhood, there is little difference between the local structure patterns of each node’s neighbors in GNNs when the model performs aggregation operation in the neighborhood. Because of this reason, it is difficult for the stable representation generated by GNNs to contain the difference of different neighbors’ structural information, even if the semantic difference is very large. Therefore, GNNs cannot distinguish non-isomorphic graphs due to the loss of the difference of the structure information by the aggregator of GNNs, and have some defects in identifying some common topology patterns. As shown in Fig. 3, our model uses the node embedding of each node as part of the attention mechanism to make full use of different topology patterns and successfully solves the above problem.

VII. EXPERIMENTS

In this section, we perform extensive node-label classification tasks on various graph-based benchmark datasets and evaluate our proposed model method.

A. EXPERIMENT SETUP

We validate our model in three benchmark graph datasets proposed by [36]. The characteristics of the datasets are shown in TABLE 3. These three datasets are all citation networks, in which nodes represent articles and edges indicate that one article is cited by another. The node feature is the bag-of-words representation of the article, and the node label is the academic topic of the article in the citation networks.

To demonstrate the advantages of our method, we compare our proposed model with several representative and state-of-the-art approaches, including two deep neural network based semi-supervised learning methods–Planetoid [37] and DeepWalk, five graph neural network methods–GCN, GAT, Disentangled Graph Convolutional Networks (DGCN) [38], Independence Promoted Graph Disentangled Networks (IPGDN) [39] and ADSF. Following the setup in [5] and [8], we use all unlabeled node features and the labels of 20 nodes per class for training. The performance of all the baseline methods are evaluated on 1,000 test nodes and verified on 500 validation nodes.

We adopt the same basic setting of network as GAT to set the hyper-parameters of our model. For the method of building topology neighborhood, we use the node embedding vector obtained by GraphWave to construct the neighborhood with remote nodes in this experiment. For the aggregation method of topology neighborhood, we adopt the basic settings of the graph neural network method used in our model. In this experiment, we use two aggregation methods of semi-supervised graph neural network, GCN and GAT in the following. For the graph structure information used in our model, we use two network embedding methods, GraphWave and Node2vec, to get the node embedding vector. Of course, the three variable components of our model are free to choose other methods and combine them.

We tune the hyper-parameters of both our model and our baselines using hyperopt [40]. Our hyper-parameter search space is as follows: the learning rate ~ loguniform$[e^{-8}, 1]$, the $l_2$ regularization term ~ loguniform$[e^{-10}, 1]$, dropout rate ~ loguniform$[e^{-8}, 1]$, and the dimension of the output vector of topology neighborhood aggregation $N \in \{2, 4, \ldots, 32\}$. We run hyperopt for 1000 trials for each setting. Finally, with the best hyper-parameters on the validation sets, we report the averaged performance of 100 runs on each semi-supervised citation network dataset.
We discuss the label filtering scheme and verify that it improves the proportion of positive information in the high-order topology graph. Fig. 4 shows the effect of label filter component on the performance of the model. Here, we report the accuracy of the semi-supervised classification of the four variants in the absence of the label filter component on three datasets—Cora, Citeseer and Pubmed. Compared with the model without label filter, it can be seen that the process of node selection improves the accuracy of node classification obviously. Therefore, increasing the weight

the aggregation method of topology neighborhood is GAT, the classification accuracy is higher than that when the aggregation method of topology neighborhood is GCN. When the graph structure information is generated by GraphWave, the classification accuracy is higher than that when the graph structure information is generated by Node2vec. This shows that GAT used to aggregate the node representations in the topology neighborhood can provide more useful information of remote similar nodes for aggregation of central nodes than GCN, and the node embedding generated by GraphWave can extract the structural information around nodes more accurately than Node2vec on Cora.

Similarly, for Citeseer dataset, we can find that the classification accuracy when the aggregation method of topology neighborhood is GCN is higher than that when the aggregation method of topology neighborhood is GAT. The classification accuracy when the graph structure information is generated by Node2vec is higher than that when the graph structure information is generated by GraphWave. For Pubmed dataset, we can find that the classification accuracy when the aggregation method of topology neighborhood is GCN is higher than that when the aggregation method of topology neighborhood is GAT. The classification accuracy when the graph structure information is generated by Node2vec is higher than that when the graph structure information is generated by GraphWave.


c) Pubmed

FIGURE 4. Label filter performance.


<table>
<thead>
<tr>
<th>Method</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep-Walk</td>
<td>67.2%</td>
<td>43.2%</td>
<td>65.3%</td>
</tr>
<tr>
<td>Planetoid</td>
<td>75.7%</td>
<td>64.7%</td>
<td>77.2%</td>
</tr>
<tr>
<td>GCN</td>
<td>81.5%</td>
<td>70.3%</td>
<td>79.0%</td>
</tr>
<tr>
<td>GAT</td>
<td>83.0%</td>
<td>72.5%</td>
<td>79.0%</td>
</tr>
<tr>
<td>DGCN</td>
<td>83.7%</td>
<td>73.1%</td>
<td>80.3%</td>
</tr>
<tr>
<td>IPGDN</td>
<td>84.1%</td>
<td>74.0%</td>
<td>81.2%</td>
</tr>
<tr>
<td>ADSF</td>
<td>85.4%</td>
<td>74.0%</td>
<td>81.2%</td>
</tr>
<tr>
<td>GAT-GraphWave-GraphWave</td>
<td>85.8%</td>
<td>74.8%</td>
<td>80.6%</td>
</tr>
<tr>
<td>GCN-GraphWave-GraphWave</td>
<td>84.9%</td>
<td>75.1%</td>
<td>81.4%</td>
</tr>
<tr>
<td>GAT-GraphWave-Node2vec</td>
<td>84.7%</td>
<td>74.5%</td>
<td>80.4%</td>
</tr>
<tr>
<td>GCN-GraphWave-Node2vec</td>
<td>84.6%</td>
<td>74.6%</td>
<td>80.6%</td>
</tr>
</tbody>
</table>

B. PERFORMANCE RESULTS OF SEMI-SUPERVISED CLASSIFICATION

The experimental results are shown in TABLE 4. In our experiment, we enumerate four different variants based on the general framework of our model. In TABLE 4, the first variant is GAT-GraphWave-GraphWave, in which the aggregation method of topology neighborhood is GAT, which is a semi-supervised graph neural network. The first GraphWave represents the method of building topology neighborhood and the second GraphWave represents that the graph structure information used in the model is the node embedding vector generated by GraphWave. Similarly, the other three variants are: GCN-GraphWave-GraphWave, GAT-GraphWave-Node2vec and GCN-GraphWave-Node2vec. Of course, according to specific dataset, different components can be used to produce different variants. In this experiment, we only list the above four different variants.

Our results are significantly better than the baseline. Several variants of our model perform better than some semi-supervised learning methods, such as Planetoid and DeepWalk, which shows the effectiveness and advantage of our model ATPGNN in graph based semi-supervised learning tasks. Several variants of our model perform better than graph neural network methods—GCN, GAT, DGCN, IPGDN and ADSF, that is to say, compared with other competing graph convolution models, the proposed topology neighborhood aggregation component and the full utilization of graph structure information in our model ATPGNN are more effective for graph data representation and learning.

Comparing the experimental results among the four variants of our model, for Cora dataset, we can observe that when the aggregation method of topology neighborhood is GAT, the classification accuracy is higher than that when the aggregation method of topology neighborhood is GCN. When the graph structure information is generated by GraphWave, the classification accuracy is higher than that when the graph structure information is generated by Node2vec. This shows that GAT used to aggregate the node representations in the topology neighborhood can provide more useful information of remote similar nodes for aggregation of central nodes than GCN, and the node embedding generated by GraphWave can extract the structural information around nodes more accurately than Node2vec on Cora.

Similarly, for Citeseer dataset, we can find that the classification accuracy when the aggregation method of topology neighborhood is GCN is higher than that when the aggregation method of topology neighborhood is GAT. The classification accuracy when the graph structure information is generated by Node2vec is higher than that when the graph structure information is generated by GraphWave. For Pubmed dataset, we can find that the classification accuracy when the aggregation method of topology neighborhood is GCN is higher than that when the aggregation method of topology neighborhood is GAT. The classification accuracy when the graph structure information is generated by Node2vec is higher than that when the graph structure information is generated by GraphWave.

C. MODEL ANALYSIS

In this part, we analyze the influence of some parameters and components of the model on the performance about the task of semi-supervised node classification.

1) LABEL FILTER PERFORMANCE

We discuss the label filtering scheme and verify that it improves the proportion of positive information in the high-order topology graph. Fig. 4 shows the effect of label filter component on the performance of the model. Here, we report the accuracy of the semi-supervised classification of the four variants in the absence of the label filter component on three datasets—Cora, Citeseer and Pubmed. Compared with the model without label filter, it can be seen that the process of node selection improves the accuracy of node classification obviously. Therefore, increasing the weight
of positive information can better optimize the topology neighborhood.

2) NUMBER OF LAYERS
The number of network layers is an important index to evaluate the stability of the model. Whether the model can maintain stable performance after increasing the acceptance domain of each node’s aggregation operation in the graph is very important for downstream tasks. In Fig. 5, we report the impact of the number of the model network layers on the performance of GCN, GAT and the four variants on three datasets—Cora, Citeseer and Pubmed. We extend the neighborhood of the graph processed by GCN and GAT with remote similar nodes, so that the neighborhood aggregation size of each node is consistent with our model. We can find that our model variants perform best when the number of network layers is two. While the GCN and GAT only perform best when the number of network layers is one. This is because GCN only averages neighbor nodes and aggregates them. GAT only weights node representations in the process of neighborhood aggregation by the difference between node features. Increasing the number of network layers will make the representations of nodes smooth for GCN and GAT. This shows that the extension of the attention mechanism of our model in graph topology information and high-order neighbors’ information can effectively overcome the problem of vanishing gradient in GCN and GAT.

3) SIZE OF TOPOLOGY NEIGHBORHOOD
Topology neighborhood is constructed by specific geometric relations. The graph data is mapped into a potential space by network embedding, and then in this potential space, each node takes the first \( n \) most similar nodes. The size \( n \) of topology neighborhood is a key parameter, which can neither be too small nor too large. We report the impact of changes in the size of topology neighborhood of the model on the performance of the four variants on three datasets—Cora, Citeseer and Pubmed. We conduct semi-supervised node classification experiments on different sizes of topology neighborhoods, and the results of the experiments are shown in Fig. 6. As the size of the topology neighborhood increases, the classification accuracy of the four variants shows a convex function, and reaches the maximum value when the size of the topology neighborhood is \( n = 1 \). This shows that the larger size of the topology neighborhood, the more negative information it contains. Of course, if the size of the topology neighborhood is too small, it cannot provide enough information to facilitate the aggregation operation of the central nodes. In our method, the optimal choice of topology neighborhood is \( n = 1 \), so our method can make full use of high-order neighborhood nodes when the size of topology neighborhood is equal to the average degree of nodes, and is equivalent to double the size of neighborhood for each node.

4) ABLATION STUDIES ON THE FEATURES GENERATED IN THE TOPOLOGY NEIGHBORHOOD AND NETWORK EMBEDDING
For our ATPGNN model, we use two kinds of information not used in GAT model, that is, the representation for each node containing the information of similar remote nodes generated by aggregating neighbor nodes in high-order topology graph, and the graph structure information around each node. We analyze the influence of the information of similar remote nodes or the structure information on the performance of the model. We perform ablation studies using the four variants on three datasets—Cora, Citeseer and Pubmed. The results of semi-supervised node classification are shown in Fig. 7. We find that the performance of our model is much better than that of GAT in most cases when the information of similar remote nodes or the structure information is removed. This shows that the two model components in ATPGNN can improve the performance of GAT independently in most cases. The reason why some variants are not as good as GAT is that when the available remote similar node information or the graph structure information is reduced, the model does not have enough positive information to resist the vanishing gradient effect caused by the increase of model network layers and neighborhood aggregation range of nodes. This is because these two factors will bring not only the relevant information for the central nodes, but also some noise in the process of node aggregation. In a few cases, we need to combine the information of similar remote nodes and the graph structure information mentioned above to improve the performance of the model. In most cases, the positive infor-
mation provided by either the information of similar remote nodes or the graph structure information can overcome the effect and improve performance.

5) PERFORMANCE OF THE MODEL WITHOUT NODE FEATURE
We make some interesting experiments, that is to remove the node feature and evaluate the performance of the model. We perform ablation studies using the four variants on three datasets—Cora, Citeseer and Pubmed. The results of semi-supervised node classification are shown in Fig. 8. When the node features are removed, the performance of the model is still much higher than that of GAT in most cases, which shows that the difference information between the nodes in neighborhood contributed by remote similar nodes and network embedding is more than that of node features.

D. STATISTICAL ANALYSIS OF ROC CURVE
We draw the ROC curve of classification results on the datasets—Cora, Citeseer and Pubmed, taking the model with the best classification accuracy among our four models as an example. For Cora dataset, the model with the best performance is GAT-GraphWave-GraphWave. For Citeseer dataset, the model with the best performance is GCN-GraphWave-GraphWave. For Pubmed dataset, the model with the best performance is GCN-GraphWave-GraphWave. The results of the experiments are shown in Fig. 12. For each dataset, we draw the ROC curve of the classification result of each class and the overall classification results. The ROC curve of the overall classification results can be divided into two types according to micro calculation method or macro one. We also calculate the AUC values of the classification result for each class and the whole.
FIGURE 9. Visualization embedding on Cora.

FIGURE 10. Visualization embedding on Citeseer.

FIGURE 11. Visualization embedding on Pubmed.

FIGURE 12. ROC curve of semi-supervised node classification.
E. VISUALIZATION ON REAL WORLD DATASETS

In order to study the performance of our model in the node feature representation, we finally visualize the real-world dataset to qualitatively evaluate our model. We get the output of the last layer of the the four variants on the datasets—Cora, Citeseer and Pubmed, and then map the representation vectors to two-dimensional space using t-SNE [41]. The final visualization results are shown in Fig. 9, Fig. 10 and Fig. 11. Each point indicates one author and its color indicates the research area. We can see that the representation learned by our four different models has high intra-class similarity and separates articles in different research area with distinct boundaries. The nodes with same labels show clear clusters, indicating that the classification performance of the four variants is good.

VIII. CONCLUSION

Many real-world data are presented in the form of graph, such as citation networks formed by citation relationships between articles, social networks composed of interpersonal relationships, knowledge maps, etc. GNNs can process these data and generate corresponding representations for entities in these networks. However, some existing GNNs models cannot make good use of graph topology information, and often ignore the positive information of high-order nodes for the central nodes to be aggregated. Due to these factors, the existing GNNs models cannot generate accurate node representations for downstream tasks. To solve these problems, we propose a semi-supervised general learning platform with three variable components. In particular, we study how to make full use of the topological information and high-order neighborhood nodes in GNNs. We create a high-order topology graph by mapping the graph data to a potential space by network embedding methods. Then the node aggregation operation is performed on the high-order topology graph. Next, we apply the abundant graph structure information around nodes to the aggregation operation in GNNs. Finally, the attention mechanism operation on graph is applied to the feature representation obtained by aggregating nodes in the topology neighborhood, the node embedding representation, and the node feature to update the representation of each node. The platform ATPGNN can support inductive learning and adapt to different tasks. By changing the variable components of our semi-supervised classification platform, we propose four typical models. We demonstrate the adaptability of our method to different tasks, and our model can use graph structure information more efficiently than existing methods to obtain better performance. We also propose a new design scheme to guide the development of GNNs in the future.

REFERENCES


KEHAO WANG (Member, IEEE) received the B.S. degree in electrical engineering and the M.S. degree in communication and information systems from the Wuhan University of Technology, Wuhan, China, in 2003 and 2006, respectively, and the Ph.D. degree from the Department of Computer Science, University of Paris-Sud XI, Orsay, France, in 2012. In 2013, he was a Post-Doctoral Researcher with The Hong Kong Polytechnic University. In 2013, he joined the School of Information Engineering, Wuhan University of Technology, where he is currently an Associate Professor. Since 2015, he has been a Visiting Scholar with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA, USA. His research interests are stochastic optimization, operation research, scheduling, wireless network communications, and embedded operating system.

HANTAO QIAN received the B.S. degree in electrical engineering from the Wuhan University of Technology, Wuhan, China, in 2019, where he is currently pursuing the M.S. degree with the Department of Information Engineering. His research interests include data hiding, machine learning, transfer reinforcement learning, and deep reinforcement learning.

KEZHONG LIU (Member, IEEE) received the B.S. and M.S. degrees in marine navigation from the Wuhan University of Technology (WUT), Wuhan, China, in 1998 and 2001, respectively, and the Ph.D. degree in communication and information engineering from the Huazhong University of Science and Technology, Wuhan, in 2006. He is currently a Professor with the School of Navigation, WUT. His active research interests include indoor localization technology and data mining for ship navigation.

MOZI CHEN (Member, IEEE) received the B.S. degree in electric engineering from the Hubei University of Science and Technology, China, in 2013, and the M.S. and Ph.D. degrees in navigation engineering from the Wuhan University of Technology (WUT), China, in 2016 and 2020, respectively. In 2021, he joined the School of Navigation, WTU, where he is currently an Associate Professor. His research interests focus on wireless sensor networks and indoor localization.

KAI ZHENG received the Ph.D. degree with the School of Geodesy and Geomatics, Wuhan University. He is currently Associate Researcher with the Wuhan University of Technology. His research interests are GNSS precise positioning techniques.
PAN ZHOU (Senior Member, IEEE) received the B.S. degree (Hons.) in the advanced class and the M.S. degree from the Department of Electronics and Information Engineering, Huazhong University of Science and Technology (HUST), Wuhan, China, in 2006 and 2008, respectively, and the Ph.D. degree from the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, USA, in 2011. He was a Senior Technical Member with Oracle, Inc., USA, from 2011 to 2013, and worked on Hadoop and distributed storage system for big data analytics at Oracle cloud platform. He is currently a Full Professor and a Ph.D. Advisor with the Hubei Engineering Research Center on Big Data Security, School of Cyber Science and Engineering, HUST, Wuhan, China. His current research interest includes security and privacy, big data analytics, machine learning, and information networks. He received the merit research award from HUST in his master study. He received the Rising Star in Science and Technology of HUST in 2017. He is currently an Associate Editor of the IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING.

DAPENG WU (Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2003. He is currently a Professor with the Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL, USA. His research interests are in the areas of networking, communications, signal processing, computer vision, machine learning, smart grid, data mining, and information and network security.