IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON ADVANCED DATA MINING METHODS FOR SOCIAL COMPUTING

Received August 6, 2019, accepted August 22, 2019, date of publication August 28, 2019, date of current version September 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2938039

hpGAT: High-Order Proximity Informed Graph Attention Network

ZHINING LIU[1], WEIYI LIU[2], PIN-YU CHEN[3], CHENYI ZHUANG[4], AND CHENGYUN SONG[5]

[1]School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China
[2]JD Intelligent Cities Research and JD Intelligent Cities Business Unit, Chengdu 610046, China
[3]IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA
[4]Ant Financial Services Group, Hangzhou 310012, China
[5]School of Computer Science and Engineering, Chongqing University of Technology, Chongqing 400054, China

Corresponding author: Chengyun Song (scyer123@163.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 41804112, and in part by the Scientific Research Foundation of Chongqing University of Technology.

ABSTRACT Graph neural networks (GNNs) have recently made remarkable breakthroughs in the paradigm of learning with graph-structured data. However, most existing GNNs limit the receptive field of the node on each layer to its connected (one-hop) neighbors, which disregards the fact that large receptive field has been proven to be a critical factor in state-of-the-art neural networks. In this paper, we propose a novel approach to appropriately define a variable receptive field for GNNs by incorporating high-order proximity information extracted from the hierarchical topological structure of the input graph. Specifically, multiscale groups obtained from trainable hierarchical semi-nonnegative matrix factorization are used for adjusting the weights when aggregating one-hop neighbors. Integrated with the graph attention mechanism on attributes of neighboring nodes, the learnable parameters within the process of aggregation are optimized in an end-to-end manner. Extensive experiments show that the proposed method (hpGAT) outperforms state-of-the-art methods and demonstrate the importance of exploiting high-order proximity in handling noisy information of local neighborhood.

INDEX TERMS Graph neural network, high-order proximity, network embedding.

I. INTRODUCTION

Graph neural networks have been successfully applied to handling non-Euclidean data such as graph-structured data (e.g. social networks, 3D point clouds, and biological networks) [1]. Unlike grid-structured data (e.g., images or audio waveforms), graph-structured data would require neural networks to support irregular inputs. Hence, directly applying a typical grid-structured based deep learning method to graph-structured data could be suboptimal.

To overcome such obstacle, graph convolution has been proposed [2]. By implementing a convolution operator on a graph (mostly, on the Laplacian matrix of a graph) in the spectral domain, one can successfully extract multiscale information from the irregular inputs. However, these spectral methods (e.g., [2]–[4]) may suffer from the computation complexity issues due to their inefficiency in conducting expensive eigendecomposition of the graph Laplacian matrix.

The associate editor coordinating the review of this article and approving it for publication was Shirui Pan.

To circumvent the computation complexity, recent works [5]–[8] discard the pervasive high-order information on a graph, and instead aggregate the information of neighbors to the node itself, which turns out to be a first-order approximation of the computationally-intensive spectral methods [5]. Notably, these graph neural networks usually limit the receptive field of the node on each layer to its one-hop neighbors and hence could be myopic for aggregation. On the other hand, a large receptive field [9] is known to be a critical factor in designing state-of-the-art neural networks. One conventional approach to increasing the receptive field is by stacking multiple layers, but we find that doing so will severely deteriorate the learning performance on the graph-structured data (see Fig.2 for details).

In this paper, we propose a novel approach to appropriately define a large receptive field for GNNs by incorporating high-order proximity information extracted from the input graph. We motivate our proposal using Fig.1. Scenario 1 elucidates a typical receptive field (with 1-hop neighbors) for graph convolution related algorithms. To obtain a dense

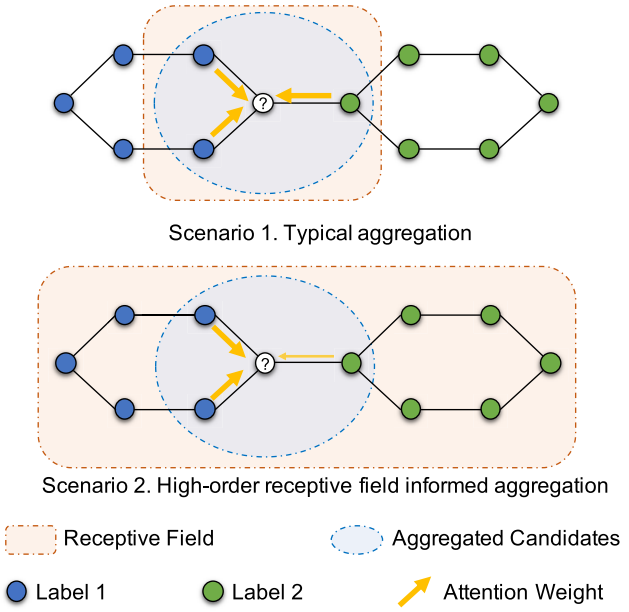


FIGURE 1. An illustration of high-order proximity information for graph convolution algorithms. Aggregated candidates indicate the nodes to be aggregated to the bridge node (marked in white color). The thickness of yellow arrows reflects the aggregation strength. The receptive field represents the range of the topological information the bridge node can perceive. Different color labels indicate different groups.

representation of the ''bridge node'' (marked in white color), its three neighbors are chosen for aggregation. However, in this case, an ideal learning algorithm should simultaneously emphasize the information from the blue group and suppress the influence from the green group for aggregation, otherwise the aggregated information might be too ''noisy'' if the weighted aggregation was falsely influenced by its neighbor in the green group. For example, if the three neighboring nodes share the same node attribute, there is no way to suppress the influence from the green group if calculating the weights only based on the node attribute.

To alleviate the problem of noisy aggregations, we propose to design an appropriate receptive field on the graph based on high-order proximity, which provides rich high-order topological information for a wide variety of graph analysis tasks [10]–[15]. Taking Scenario 2 as an example, with the increased receptive field, the bridge node can better perceive the topological structure embedded in the graph (without leveraging the label information) and adjust its local aggregation process accordingly. Therefore, in this case, high-order proximity information provides topology-aware aggregation and can suppress the noise from the green group. We will further demonstrate the benefit of incorporating high-order proximity information for aggregation on real-world datasets (see Fig.3 for details).

Another known issue in graph convolution related framework is that convolving all connected neighbors of a node without considering their topological roles does not comply with a basic but widely accepted hypothesis in graph analysis – different neighbors contribute differently to the node. In 2018, [7] proposed graph attention networks (GAT) such that each node would aggregate information from its one-hop neighbors with different attention coefficients. Nonetheless, despite its remarkable performance improvement and capability for both transductive and inductive learning on graph-structured data, the receptive field of GAT is still limited to one-hop neighbors, which again disregards the rich information from high-order proximity.

Targeting at solving the aforementioned problems, we propose **h**igh-order **p**roximity informed **g**raph **at**tention networks, **hpGAT**. There are several novel designs in hpGAT. First of all, inspired by a non-negative matrix factorization method (semi-NMF) that is trainable on neural network models [16], [17], we propose to design a group-aware and flexible receptive field for GNNs. Semi-NMF is an unsupervised method that encodes soft membership information for each node, which offers a structure-aware receptive field of arbitrary-order proximity. Second, with different receptive fields for topological group summarization at varying scales, we propose a new graph learning module consists of several graph attention layers that take high-order proximity to learn hidden representations of nodes. Finally, to integrate high-order topological information into the attention mechanism, hpGAT jointly optimizes non-negative matrix factorization and node classification. By training hpGAT in an end-to-end manner and adopting high-order proximity information via semi-NMF, the graph attention mechanism would learn to become more topology-aware and hence less myopic when aggregating neighboring nodes. In summary, the major contributions of this paper are summarized as follows:

1) We provide new insights and fine-grained analysis to study how the GAT model works on the task of node classification, and highlight the problem of noisy aggregation when the topological information is not fully utilized.
2) We propose hpGAT, an end-to-end trainable system which fuses high-order proximity information embedded in the multiscale representation of the input graph as well as node attributes weighted by the attention mechanism.
3) We conduct extensive experiments and analysis to demonstrate the effectiveness of hpGAT on several real-world datasets. hpGAT outperforms the state-of-the-art methods, and its superior performance is corroborated by observing its boosted performance in classifying nodes connecting to different groups.

II. RELATED WORKS

Deep neural networks have been successfully applied in a number of high impact domains, but most of them are specialized for handling grid-structured data, which do not align well with graph-structured data due to irregular inputs. How to generalize neural networks to work on arbitrarily structured graphs is still a challenging problem. Aiming at solving this, a surge of research interest has been devoted to

studying deep learning on graphs. Based on recent research finds, [18] proposed to divide the existing deep learning methods on the graph into three main categories: semi-supervised methods, unsupervised methods, and recent advancements. Specifically, semi-supervised methods include Graph Neural Networks [19] and Graph Convolutional Networks (GCNs) [3], [5], unsupervised methods are mainly composed of Graph Autoencoders [20]–[23] and recent advancements include Graph Recurrent Neural Networks [24]–[26] and Graph Reinforcement Learning [27]. For more details, we refer the reader to [18], [28], [29]. Among these methods, the GCNs have attracted a great deal of attention and serve as an important role in building up many other complex graph neural network models. Generally speaking, existing GCNs can be categorized into two types: spatial-domain based and spectral-domain based.

Spatial-domain based methods mainly focus on imitating the convolution operation of a conventional convolution neural network on grid-structured data (mostly based on the aggregation of the information of neighboring nodes) and extending the convolution operation to adapt to graphs. Through introducing a diffusion-convolution operation, [30] presented diffusion-convolutional neural networks, which outperforms probabilistic relational models and kernel-on-graph methods in the task of node classification. Reference [31] proposed a unified framework named Message Passing Neural Networks (MPNNs) for the graph convolution operation using a message passing function. Specifically, each node sends messages to its neighbors based on its current embedding representation and generates its new embedding representation based on messages received from immediate neighbors. Reference [6] introduced GraphSAGE, which generates embedding representations in an inductive way for each node by sampling a fixed number of nodes from its spatial neighborhood and aggregating their node features. Three aggregators (mean, LSTM and pooling) are optional, which are all integrated with trainable parameters. Furthermore, to enhance the scalability of GraphSAGE, [32] proposed PinSAGE with efficient random walks and graph convolutions. But these aggregators used in the two aforementioned do not explicitly filter information, which could easily introduce irrelevant information into the final embedding representations. To address this problem, [7] introduced the graph attention mechanism (GAT) to allow different weights for different neighbors of each node, which further improves the performance of node classification. Furthermore, [33] included another self attention pooling layer to generalize the graph representation from the various aspects of a matrix graph embedding, which leads to the dual attention graph convolutional networks. Recently, [34] proposed Graph Networks (GNs), which is a more general framework with three aggregation functions and three update functions. Other than graph topology and node attributes, GNs also include edge representations and the whole graph representation into consideration, which further improve the performance. In addition, there is also a surge of interest in the spatial-temporal graph modeling through integrating graph convolution with recurrent neural networks or convolution neural networks [35].

Spectral-domain based methods mainly take advantage of the spectral representations of graphs. By extending the operator of the convolution to the spectral domain based on the spectrum of the graph Laplacian, [3] defined a diagonal matrix as trainable parameters for each layer. However, the time complexity of this method is intensive due to expensive eigendecomposition of the graph Laplacian matrix, and these filters are non-spatially localized, which is not an efficient way for representation learning. Later, [2] proposed a novel parameterization by introducing a smoothing kernel for better capturing spatial locality. Reference [4] proposed to approximate the filters using the Chebyshev expansion that can be computed recursively from the graph Laplacian. With the help of this approximation, eigendecomposition is no longer required and the locality of filters can be ensured. Similarly, [36] proposed to use Cayley polynomials of the graph Laplacian to approximate localized filters on graphs, which offer better frequency localization than Chebyshev polynomials. Moreover, [5] proved that the first-order approximation of spectral convolutions on graphs is efficient enough to build representative convolutional filter functions by stacking multiple layers. With these simplifications, spectral-domain based methods have shown close connections to spatial-domain based methods for their similarity in aggregating information from neighboring nodes.

In summary, spatial-domain based methods and spectral-domain based methods both show promising results on various graph related analytics tasks. But spatial-domain based methods outperform spectral-domain based methods in terms of efficiency, generality and flexibility [29].

III. PRELIMINARIES AND MOTIVATION

A. GRAPH ATTENTION NETWORKS (GAT)

Given a graph $G = (V, E, X)$, where V and E represent the node set and edge set, respectively, and $X \in \mathbb{R}^{n\times c}$ (n is the number of nodes and c is the length of each feature vector) is the matrix representing node attributes. A common semi-supervised learning task is to infer the labels of a subset of nodes given G and known labels of the other nodes.

GAT [7] solves this problem by designing a graph attention layer (GAL). Let $P = \{\vec{p_1}, \vec{p_2}, \ldots, \vec{p_n}\} \in \mathbb{R}^{n\times d}$ (d is the dimension of the latent representation) denote the input to this layer, which are the node representations generated from the previous layer (the input of first layer is the node attributes X, and we omit the layer index for brevity).

GAT first applies a learnable linear transform on representations (parameterized by $\mathbf{W} \in \mathbb{R}^{d\times d'}$) of each two connected node pairs to compute the attention coefficients with the function $f : \mathbb{R}^{d'} \times \mathbb{R}^{d'} \to \mathbb{R}$:

$$e_{ij} = f(\mathbf{W}\vec{p_i}, \mathbf{W}\vec{p_j}) \tag{1}$$

The attention coefficients are then normalized across the neighbors of the targeted node i by using *softmax* function:

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{q \in \mathcal{N}_i} exp(e_{iq})} \tag{2}$$

where $\mathcal{N}_i$ is the set of neighboring nodes of node i, including node i itself.

With the attention coefficients, the final output of node i in the graph attention layer is the aggregated node representations from $\mathcal{N}_i$:

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

where $||$ denotes vector concatenation, σ is the *sigmoid* function and K is the number of heads (each head has independent transformation parameters). Technically speaking, we can view GAT as a trainable version of label propagation algorithm. The correlation between these two algorithms is discussed in the APPENDIX.

B. THE LIMITATION ON GAT

For each node, GAT leverages its 1-hop neighbors to perform weighted aggregation to obtain the hidden representation of the node itself, which implicitly indicates that for a typical GAT with two layers, the receptive field of a node is in fact limited to its 2-hop neighbors. As discussed above, increasing the range of receptive field helps a learning algorithm perceive richer topological information. Hence, for GAT, one naive way to increase the receptive field is by simply stacking more layers.

In this section, we first use real-world datasets to report the unsatisfactory performance of a typical node classification task by stacking more layers in GAT, then we give an explanation on the pessimistic outcomes we have observed. The experiment setup is the same as in the Experiments section except for that we use one head instead of eight for each graph attention layer due to the limited GPU memory.

Fig.2 shows the accuracy when stacking multiple layers. Here, different colors indicate different datasets, X-axis demonstrates the number of layers a GAT has, and the Y-axis is the accuracy. We conclude from the results that (i) On all datasets, when increasing the number of layers in GAT from 1 to 2, the performance increases accordingly, indicating that GAT gains more topological information from two layers than a single layer. (ii) However, by continuing adding more layers into GAT, the performance begins to decrease.

The outcome of the noticeable performance decay in all datasets indicates a limitation of GAT – stacking more layers does not yield the expected effect of the increased receptive field. For GAT, according to Eq.3, the core step in updating the representation $\vec{p}_i$ of a node i in a layer is achieved by weighted aggregation on the representations of all its neighbors $j \in \mathcal{N}_i$. This process can be regarded as a typical Laplacian smoothing process that implicitly increases the receptive field of a node.

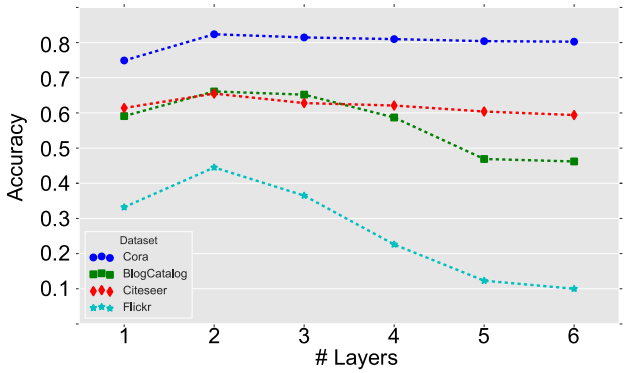


FIGURE 2. The accuracy of node classification task when stacking layers in GAT. Limited by the GPU memory, we use one head ($K = 1$) for each graph attention layer.

However, by repeatedly applying Laplacian smoothing many times (i.e. stacking more layers), the whole model will suffer from the over-smoothing problem. References [37], [38] have proved that by repeatedly applying Laplacian smoothing, the features of nodes within each connected component of the graph will converge to the same values. For GAT, as stacking layers is equivalent to applying Laplacian smoothing, it will also suffer from the over-smoothing problem on the learned node representations, leading to performance degradation.

C. ADVANTAGE IN HIGH-ORDER PROXIMITY

Recently, high-order proximity has shown to be effective in many graph learning tasks, especially in the research of network embedding. We refer readers to [39] for a comprehensive review. For example, [40] directly used the k-step transition probability matrix $A^k = \prod_k A$ as the high-order proximity matrix, where A is the normalized adjacency matrix representing the transition probability matrix of a single-step random walk. Reference [41] proposed a hierarchical feature aggregation model, where they initially aggregate features at different depth (a.k.a multiple-hop neighbors) of a node on each hierarchy.

Generally speaking, the ultimate goal of all these methods is to appropriately define the receptive field of a node on the graph, since a well-defined receptive field including high-ordered proximity allows a node to perceive richer and broader knowledge of the graph topology. However, naively using multiple-hop proximity or k-step transition matrix A^k as a way to increase the receptive field may obfuscate the essential topological information of a node, and thus may incur additional noises in the learning and analysis phases.

Take Fig.3 as an example. For node a, when naively aggregating its first-order neighbors (blue circle shadow), the node is myopic and can only perceive neighbors around itself. To increase the receptive field by using A^2, node a can grasp more information (gray rectangle shadow). However, this procedure may also include some irrelevant nodes and results in noisy or even uninformative aggregation. Consequently, it is crucial to properly define an appropriate receptive field.

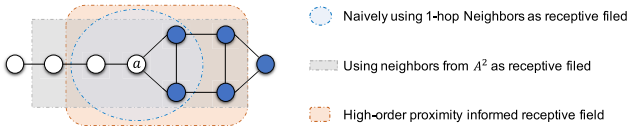


FIGURE 3. Different aggregation of node *a* can affect its perception of the network. A properly defined high-order proximity receptive field (yellow shadow) can provide *a* with more informative topological understanding than 1-hop neighbors (blue shadow), or an excessive receptive field covering the too many hops (gray shadow).

In this example, if the receptive field can perceive the topological information in advance, it is easy to find that a critical pattern that there exists a subgraph containing node a and the blue nodes. Based on this preferred receptive field (yellow rectangle shadow), node a can put more aggregation weights on its neighbors within the same subgraph, instead of aggregating other irrelevant nodes.

IV. METHODOLOGY

As discussed before, the limitation in GAT is its small receptive field (i.e., 1-hop neighbors). Moreover, simply stacking multiple layers under the framework of GAT leads to over-smoothing. To encourage the attention mechanism to be aware of the embedded high-order topological structures while maintaining its efficiency in local neighborhood aggregation, we propose a novel approach to overcome the limitation in GAT by introducing high-order proximity in the attention mechanism. Specifically, in addition to computing attention coefficients between every connected node pair using their node attributes, their pairwise topological relationships on high-order proximity are also included in the process of end-to-end learning.

In what follows, we first illustrate how to extract those pairwise relationships in the perspective of high-order proximity derived from the adjacency matrix, then we introduce a new graph attention layer based on high-order proximity. Finally, we introduce the proposed method, hpGAT.

A. EXTRACTING HIGH-ORDER PROXIMITY INFORMATION

Inspired by a non-negative matrix factorization method (semi-NMF) that is trainable on neural network models [16], [17], we propose to learn a set of low-dimensional representations that encode high-order proximity of the input graph, which can be learned from its adjacency matrix. To do this, we introduce a trainable factorization on the adjacency matrix A with two sets of matrices $\{Z_k\}_{k=1}^m$ and $\{H_k^+\}_{k=1}^m$ (m denotes the number of hierarchy of the matrix factorization, and the notation H_k^+ indicates that the matrix H_k contains only non-negative elements), which follow:

$$A \approx Z_1 Z_2 \dots Z_m H_m^+ \tag{4}$$

and

$$\begin{cases} H_{m-1}^+ \approx Z_m H_m^+ \\ \vdots \\ H_2^+ \approx Z_3 H_3^+ = Z_3 \cdots Z_m H_m^+ \\ H_1^+ \approx Z_2 H_2^+ = Z_2 \cdots Z_m H_m^+ \end{cases} \tag{5}$$

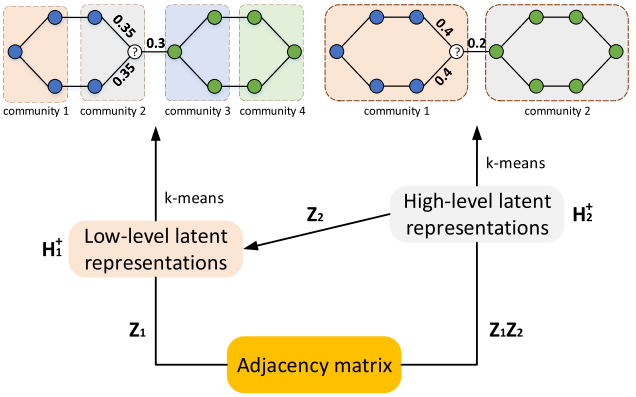


FIGURE 4. An illustration of how semi-NMF learns a hierarchy of latent representations that discover different levels of communities. Through factorizing $A \approx Z_1 Z_2 H_2^+ \approx Z_1 H_1^+$, two different levels of latent representations are generated, which are H_1^+ and H_2^+, and clustering on H_1^+ and H_2^+ leads to discover different hierarchies of communities. Furthermore, the normalized weights of S defined in Eq.8 over the three neighboring node of the bridge node (marked in white color) are also given over the corresponding edges. We can see that with the help of high-level latent representations, a much smaller weight is put on the link to the green group.

As stated in [16], [17], $\{H_k^+\}_{k=1}^m$ are restricted to be non-negative, whose entries are soft membership indicators for each node at different scales, and $\{Z_k\}_{k=1}^m$ are the corresponding base matrices that can be considered as group (cluster) centroids. Taking $m = 1$ as an example, we factorize A into two factors: $A \approx ZH^+$. If the orthogonal constraint is also imposed on H^+, such that $H^+(H^+)^T = I$, then every column vector would have only one positive element [42], which making semi-NMF equivalent to k-means with the same loss function:

$$loss = \sum_{i=1}^{n} \sum_{j=1}^{k} h_{ki} \|\vec{a}_i - \vec{z}_k\|^2 = \left\|A - ZH^+\right\|_F \tag{6}$$

where k is the number of clusters, h_{ki} is the element at the k^{th} row and i^{th} column of the matrix H^+, $\vec{a}_i$ and $\vec{z}_k$ are the i^{th} and k^{th} column of the matrix A and Z, respectively. $\|\cdot\|$ denotes the L2-norm of a vector and $\|\cdot\|_F$ is the Frobenius norm of a matrix.

Therefore, we can view the semi-NMF without the orthogonality constraint on the factor H^+ as a soft clustering method where H^+ is the soft membership indicator for each node, and Z denotes the corresponding cluster centroids. Moreover, $\{Z_k\}_{k=1}^m$ and $\{H_k^+\}_{k=1}^m$ are constructed following a hierarchical manner. By doing so, the representations $\{H_k^+\}_{k=1}^m$ encode high-order proximity on different hierarchies. Specifically, we show an example with $m = 2$ in Fig.4. With H_1^+ and H_2^+, we can discover different levels of communities in the graph through performing clustering.

B. hpGAL: GRAPH ATTENTION LAYERS WITH HIGH-ORDER PROXIMITY

With $\{H_k^+\}_{k=1}^m$, we calculate the attention coefficients e_{ij} by combining node attributes with a matrix $S \in \mathbb{R}^{n \times n}$

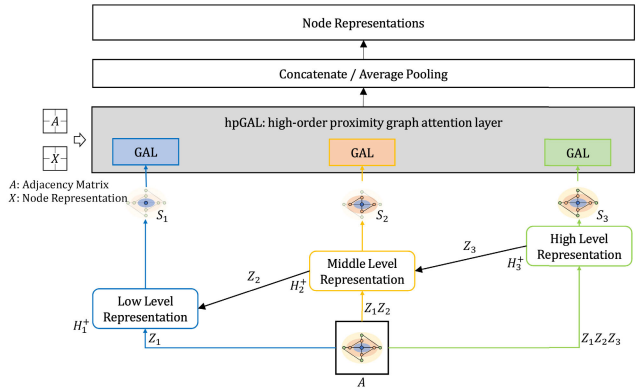


FIGURE 5. Illustration of a hpGAL with three ($m = 3$) levels of high-order proximities. Derived from the adjacency matrix A, three blocks (colored by blue, yellow and green) represent different scales of proximities, respectively, and the higher-level block is with larger receptive field. Different heights of these blocks represent the hierarchy among these blocks. $\{Z_k\}_{k=1}^m$ and H_m^+ are trainable matrix derived from the decomposition of the adjacency matrix which follow Eq.4 and Eq.5. According to Eq.8, the output of each block is fed into a separate GAL, along with the node attribute matrix X and the adjacency matrix A.

that denotes the pairwise relationships encoded high-order proximity:

$$e_{ij} = f(\mathbf{W}\vec{p}_i, \mathbf{W}\vec{p}_j) + S_{ij} \tag{7}$$

where $f = LeakyReLU(\vec{a}^T[\mathbf{W}\vec{p}_i||\mathbf{W}\vec{p}_j])$ is defined in GAT and S_{ij} denotes the (i, j) entry of the matrix $S \in \{S_1, S_2, \ldots, S_m\}$, which are calculated by

$$S_k \approx (H_k^+)^T H_k^+ \odot A \tag{8}$$

where $k = 1, \ldots, m$ and $\odot$ denotes the entry-wise (Hadamard) product of two matrices. Since $H_k^+ (k = 1, \ldots, m)$ represents soft membership indicators for each node at a specific scale [16], [17], S_{ij} can be viewed as the similarity between two nodes in the perspective of high-order proximity. Including S_{ij} in Eq.7 suggests that if two nodes are more similar in high-order proximity, larger weights should be imposed on each other when conducting the aggregation, which is illustrated in the case shown in Fig.4.

Fig.5 represents the overall architecture of a graph learning module with three-level high-order proximities (i.e., $m = 3$ and m is a hyperparameter that can be determined based on unsupervised hierarchical graph clustering methods, which will be discussed in details in next section). To be specific, this architecture contains a three-level hierarchical structure of the adjacency matrix A, where H_1^+, H_2^+ and H_3^+ captures the low, middle and high-level structures, respectively. For the k-th-level of the graph attention layer (GAL), we generate the output based on both S_k and node attributes X (or node latent representations generated by the previous layer) using Eq.3. Here, we restrict these three GALs to share the same parameters (i.e., $\mathbf{W}$ in Eq.7). Finally, the node representations are generated by average pooling or concatenating all outputs from the all GALs.

Algorithm 1 Building a hpGAT Model with 2 hpGALs

Input: Node attributes matrix X, Adjacency matrix A
Output: Predictions $\widetilde{Y}$, Reconstructed adjacency matrix A_r

1 Initializing an empty list $A_h = \{\varnothing\}$;
2 Initializing matrices $Z_1, Z_2, \ldots Z_m$ and H_m^+;
3 $H^+ \leftarrow ReLU(H_m^+)$;
4 $S \leftarrow (H^+)^T H^+ \odot A$;
5 $A_h \leftarrow \{A_h, S\}$;
6 **foreach** $Z \in \{Z_m, Z_{m-1}, \ldots, Z_2\}$ **do**
7 $\quad H^+ \leftarrow ReLU(ZH^+)$;
8 $\quad S \leftarrow (H^+)^T H^+ \odot A$;
9 $\quad A_h \leftarrow \{A_h, S\}$;
10 **end**
11 $A_r \leftarrow$ Eq.4;
12 $P \leftarrow ||_{a \in A_h} GAL(X, A, a)$ // concatenation;
13 $\widetilde{Y} \leftarrow \frac{1}{|A_h|} \Sigma_{a \in A_h} GAL(P, A, a)$ // averaging pooling;

C. hpGAT: HIGH-ORDER PROXIMITY INFORMED GAT

Throughout this paper, we build hpGAT with two hpGALs with multiple independent heads that learn how to attend the neighbors based on node representations and high-order proximity. For the first hpGAL, the input is the node attributes X and the outputs from each head would be concatenated and fed into the second hpGAL. An average pooling along with the *softmax* function is applied upon the outputs from the second GAL to obtain the predictions $\widetilde{Y}$. In order to efficiently take advantage of high-order proximity, we train hpGAT in an end-to-end manner by jointly optimizing the two tasks of matrix factorization and node classification. With the known node labels Y, the overall loss function is defined as:

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_s(Y, \widetilde{Y}) + \lambda \mathcal{L}_r, \\ \text{with} \quad \mathcal{L}_r &= \|A - A_r\|_F^2 = \left\|A - Z_1 Z_2 \cdots Z_m H_m^+\right\|_F^2 \end{aligned} \tag{9}$$

where $\mathcal{L}_s$ represents the supervised loss, A_r is the reconstructed adjacency matrix, $\|\cdot\|_F$ denotes the Frobenius norm, and λ is a hyperparameter that balances the loss of matrix reconstruction $\mathcal{L}_r$ and the supervised loss $\mathcal{L}_s$.

We summarize the algorithmic procedure of constructing hpGAT in Alg.1. Here, *ReLU* function ensures the non-negativity of $\{H_k^+\}_{k=1}^m$ (Line 3). With randomly initialized H_m^+ and $\{Z_k\}_{k=1}^m$, we obtain the similarity matrix S (Line 4) of the highest-order proximity of the graph. Then, we use a for-loop to recursively calculate S on different levels (Line 6 to 10) according to Eq.5. Finally, by using the "concatenation" operator, the first hpGAL produces the latent representations for all nodes (Line 12), and by using the "average pooling" operator, the second hpGAL outputs the prediction $\widetilde{Y}$ (Line 13).

To optimize the overall objective function in Eq.9, we choose the stochastic gradient descent optimizer to train our model. Note that our proposed algorithm does not involve any explicit matrix decomposition or expensive eigendecomposition. Comparing to GAT, it only requires a few additional

TABLE 1. Detailed information of the datasets.

Name	# Nodes	# Edges	# Attributes	# Labels
Cora	2708	10858	1433	7
Citeseer	3327	4676	3703	6
BlogCatalog	5196	171743	8189	6
Flickr	7575	239738	12047	9

learnable parameters (i.e., $\{Z_k\}_{k=1}^{m}$ and $\{H_k^+\}$). Therefore, our proposed hpGAT shares a similar complexity with GAT. We refer the readers to Section V-E for detailed complexity analysis between hpGAT and GAT.

V. EXPERIMENTS

In this section, we employ four real-world networks to validate the effectiveness of the proposed model on the task of node classification.

A. DATASETS AND EXPERIMENTAL SETUP

BlogCatalog and Flickr are two real-world social media datasets used in [43].[1] Cora[2] and Citeseer[3] are datasets based on citations between scientific papers. The details of these datasets are given in Table 1. For the experimental setup, we train all the algorithms on 20% of node labels in each dataset and use another 20% as the validation dataset for hyperparameter tuning. Then, we report the accuracy on the remaining 60% dataset. For hpGAT, we set eight attention heads for the first hpGAL and one attention head for the second hpGAL on all datasets, which is the same setup with GAT. In addition, a three-level hierarchies is adopted (i.e., $m = 3$) with the column size of $\{Z_k\}_{k=1}^{3}$ setting to 2,4,8, respectively. In addition, we have found that different values of λ have little influence on the overall performance (see Section V-D.2 for details), suggesting algorithmic stability. Hence, we use $\lambda = 1$ for all datasets. For all other baseline methods, we use the same parameter setup given in the original paper.

B. COMPARATIVE METHODS

The following are seven comparative methods (state-of-the-arts and baselines) to be compared with hpGAT.

- **DeepWalk** [44]: DeepWalk is an unsupervised network embedding method. We train a logistic regression model as the classifier based on the generated embeddings.
- **Chebyshev** [4]: Chebyshev approximates the filters by a Chebyshev expansion of the graph Laplacian to implement the convolution operation defined in the Fourier domain.
- **GCN** [5]: GCN simplifies the Chebyshev expansion with a first-order approximation of spectral convolutions.

[1] https://github.com/xhuang31/LANE
[2] https://linqs.soe.ucsc.edu/data
[3] http://csxstatic.ist.psu.edu/

TABLE 2. Node classification results.

	Cora	Citseer	BlogCatalog	Flickr
DeepWalk	0.672	0.432	0.650	0.421
Chebyshev	0.804	0.685	0.703	0.493
GCN	0.817	0.701	0.725	0.508
$GCN_{1,2}$	0.821	0.693	0.743	0.491
GAT	0.823	0.721	0.731	0.465
$GAT_{1,2}$	0.810	0.695	0.597	0.395
hpGCN	0.822	0.714	0.754	0.594
hpGAT	**0.831**	**0.730**	**0.768**	**0.603**

- **GAT** [7]: Beyond GCN, GAT introduces the attention mechanism to aggregate connected neighbors with different attention weights.
- **$GCN_{1,2}$** [28]: By using A^k instead of A as the input to the graph convolution, this method naively introduces high-order proximity into GCN. Here we test GCN with first-order and second-order graph convolution in the same manner as in [28].
- **$GAT_{1,2}$**: Similar to $GCN_{1,2}$, A and A^2 are inputs to GAL.
- **hpGCN (hpGAT w/o attention)**: Since GAT can be viewed as GCN with attention mechanism, here we also test the performance of high-ordered proximity informed GCN by removing the attention mechanism from hpGAT to demonstrate the effectiveness of high-order proximity.

C. NODE CLASSIFICATION AND ANALYSIS

1) PERFORMANCE COMPARISON

Table 2 reports the accuracy on node classification for all methods. In all datasets, our proposed hpGAT consistently attains the best performance by a large margin when compared to other methods, including the state-of-the-art methods such as GCN and GAT. The remarkable performance improvement in hpGAT is credited to the proposed high-order proximity informed attention mechanism. In addition, the results also corroborate the advantage of solely introducing high-order proximity in node classification, as hpGCN (i.e., hpGAT w/o attention) also enjoys a performance boost in accuracy.

By comparing GCN with $GCN_{1,2}$, we find that naively introducing high-order proximity through using A and A^2 together can sometimes obtain better results (observed on Cora and BlogCatalog), but it also could deteriorate the performance (observed on Citeseer and Flickr). One possible explanation is that when directly introducing k-hop neighbors into aggregation, it is more likely to involve noisy information and brings about a negative effect on classification. This phenomenon is also observed when comparing GAT with $GAT_{1,2}$. Without properly defined receptive fields (such as the use of hpGAT), $GAT_{1,2}$ results in a much worse performance than GAT due to noisy neighborhood aggregation.

2) ANALYSIS

We now dive into detailed ex post facto data analysis to provide deep insights on the performance improvement of hpGAT relative to GAT.

TABLE 3. First-order proximity label classification and purity analysis of hpGAT v.s. GAT.

Datasets	LR▽	(< 1, = 1) △	Performance Gain∗
Cora	0.47	(932, 1776)	(**+11**, +2)
Citeseer	0.36	(1967,1360)	(**+14**, +4)
BlogCatalog	0.20	(5185, 11)	(**+114**, +1)
Flickr	0.14	(7562, 13)	(**+625**, +2)

Notes: ▽: Linear classification results on test dataset. △: Purity Number ($< 1, = 1$) on all datasets. ∗: Purity Number ($< 1, = 1$) gain on test datasets.

As the adjacency matrix A contains only 1-hop neighbors for each node, to benchmark the relationships between connected neighbors of a node and their corresponding labels, we directly use the (rows of) adjacency matrix and the node labels in the training set to train a linear classifier (logistic regression) to predict the node labels. The performance of the classifier on the test datasets is reported in Table 3. We find that the performance of this baseline on Cora and Citeseer is much better than that on BlogCatalog and Flickr. It explains why hpGAT can only attain slight improvements over GAT on Cora and Citeseer since a small receptive field (first-order proximity) already contains sufficient information for predicting node labels on Cora and Citeseer.

Moreover, we propose a metric "purity" defined in Eq.10 to quantitatively analyze the distributions of the labels of neighbors of the node i:

$$purity = \frac{\sum_{j \in \mathcal{N}_i} 1_{l_i}(l_j)}{D_i} \tag{10}$$

where $\mathcal{N}_i$ is the 1-hop neighbors of node i, l_i denotes the label of the node i, 1_{l_i} is the indicator function (outputs 1 when the input equals l_i and 0 otherwise) and D_i is the cardinality of $\mathcal{N}_i$. Therefore, purity $= 1$ indicates all neighbors of a node share the same label as itself, and purity < 1 indicates the neighbors of a node have different labels from itself. Table 3 reports the purity statistics of all nodes (we call it purity number) on the four datasets. For Cora and Citeseer datasets, the purity numbers between < 1 and $= 1$ are more balanced, and for BlogCatalog and Flickr datasets, the purity numbers are heavily imbalanced. Consequently, for Cora and Citeseer, it would be easier for a graph attention mechanism to learn proper weights on the neighboring nodes because the majority of nodes have the same label as their neighbors. On the other hand, for BlogCatalog and Flickr almost every node has purity < 1 and hence learning proper attention weights is more challenging.

To validate our hypothesis that high-order proximity information can suppress noisy aggregation in GAT and hence improve the learning performance, we report the gains in purity number by comparing hpGAT with GAT in Table 3. We find that in both cases (purity < 1 or $= 1$) hpGAT has a better classification performance than GAT on all datasets. More importantly, we find that no matter the dataset is imbalanced in purity distribution or not, hpGAT is more effective in predicting the node labels with purity < 1 than GAT. The results show that the major performance gain in hpGAT actually comes from its ability to correctly predict the labels of "noisy" nodes, which proves the effectiveness and importance of high-order proximity in node classification.

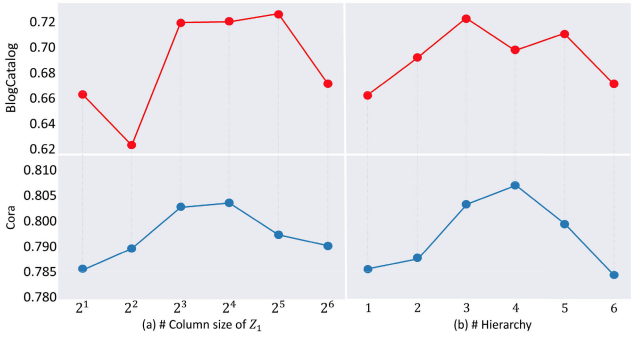


FIGURE 6. Parameter analysis with one attention head on (a) number of column size of Z, and (b) number of hierarchy.

D. PARAMETER ANALYSIS ON HIGH-ORDER PROXIMITY

In this section, we mainly analyze how the number of column size (the column size of $\{Z_k\}_{k=1}^m$), the number of hierarchy m and the weight λ affect the performance of hpGAT.

1) PARAMETER ANALYSIS ON THE COLUMN SIZE OF $\{Z_K\}_{k=1}^m$ AND m

Fig.6 reports the accuracy for hpGAT on Cora and BlogCatalog datasets with respect to different parameter settings. The vertical axis of each plot shows the resulting accuracy. To analyze the number of column size in $\{Z_k\}_{k=1}^m$, we report the performance trend of hpGAT with varying column size from 2 to 64 under a single hierarchy reconstruction (left of Fig.6). In addition, to analyze the number of hierarchies within a hpGAT, we report the performance trend of hpGAT with different hierarchies ranging from 1 to 6 (right of Fig.6), where the size of related columns is 2, 2-4, 2-4-8, 2-4-8-16, 2-4-8-16-32 and 2-4-8-16-32-64, respectively.

These results show the sensitivity of these parameters on the performance of hpGAT. Regarding the number of column size, we find that on both datasets, moderate-sized setting gives the best performance, and the stability can be expected as the "sweet spot" is quite flat. These findings are also well aligned with the intuition that a small number of column size may limit the representation power while a large number of column size may be redundant. Similarly, regarding the number of hierarchy, either setting too many or too few layers may harm the performance, which can be explained by the fact that either extremely rough or fine-grained hierarchical topological information is not ideal for learning appropriate attention weights. It is worth noting that there are many grounded methods (e.g. the Louvain method [45], [46]) in the traditional graph analysis field to automatically determine the number of hierarchy of a network in an unsupervised manner, which can be used as an informative prior for setting

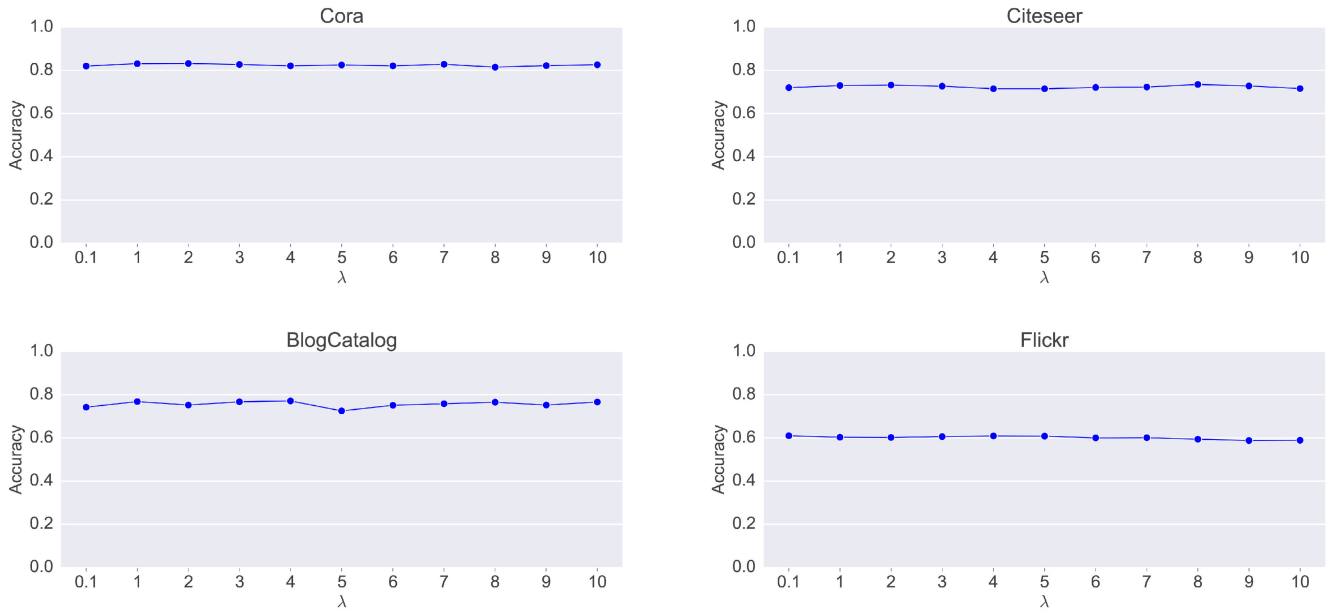


FIGURE 7. Experiments on the sensitivity of λ by changing its value from 0.1 to 10.

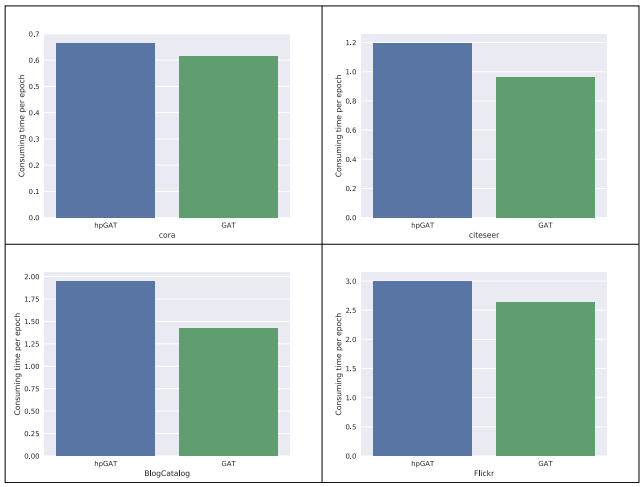


FIGURE 8. Experiments on average runtime for training an epoch for hpGAT and GAT on all four datasets, respectively.

hyperparameters in hpGAT. In fact, we find that for Cora and BlogCatalog, the number of hierarchy given by the Louvain algorithm is 4 and 2, respectively, which gives a near-optimal performance according to Fig.6.

2) PARAMETER ANALYSIS ON λ

Here, to investigate the influence of λ in Eq.9, we vary λ from 0.1 to 10. Fig.7 reports the resulting performances on the node classification task for all four datasets. For each subgraph, the X-axis represents the value of λ, and Y-axis represents the accuracy.

Inspecting the results from all four datasets, we can easily make a conclusion that the change of λ has a limited effect on the overall performance within the tested range. In other words, it means that the hyperparameter λ is not sensitive to the overall algorithm. In particular, if we set $\lambda = 0$, hpGAT reduces to the case of GAT.

E. EMPIRICAL COMPLEXITY ANALYSIS

We conduct our experiments on a typical machine with Ubuntu 16.10 system, 64G 2133MHz RAM, Intel E5-2630V4 CPU, and the Titan X GPU with 12G memory. All experiments are implemented using keras 2.2.0 and tensorflow-gpu 1.8.0.

Fig.8 shows the average runtime (in seconds) for each epoch when training a hpGAT and a GAT (both with two hpGALs/GALs). Here, one hpGAL contains three GALs with shared parameters. For each subgraph, X-axis represents all four datasets, and Y-axis represents the training time.

It is clear to see that for each dataset, the average training time for each epoch for hpGAT and GAT is similar, which empirically demonstrates that hpGAT and GAT share a similar computation complexity.

VI. CONCLUSION

In this paper, we propose a novel high-order proximity informed graph attention network (hpGAT). By introducing high-order proximity extracted from the learnable factorization of the adjacency matrix into the attention mechanism, hpGAT is more effective in aggregating information from neighboring nodes. Extensive experiments and detailed data analysis on several real-world networks demonstrate the superior performance of hpGAT on node classification.

APPENDIX
RELATION TO LABEL PROPAGATION ALGORITHM

It is well known that the node attributes help to describe the particular information of a node itself. The graph topology,

on the other hand, helps to *propagate information* between two connected nodes with respect to the weight on the edge. The original graph attention network (GAT) is able to learn representations of nodes in a graph, taking both topology and node attributes into account. Particularly, GAT advocates weighted aggregation of **one-hop neighbors** of a node for helping information propagation. This process is related to the label propagation algorithm (LPA), a well-studied framework for studying the information propagation upon topology.

Analog to GAT, LPA also leverages neighborhood information of a node to label the node itself. Alg.2 demonstrates a standard process to propagate labels on a graph. Here, $Y^{(t)}$ denotes the labels (at iteration t), $PY^{(t)}$ works as an aggregation operation over neighbors for each node (irrespective of whether the graph includes self-connections for every node or not).

Algorithm 2 Label Propagation Algorithm (LPA)

Input: Graph $G(V, E)$, (optional) Initial labels Y_L
Output: Predicted labels $\hat{Y}$

Compute diagnose matrix $D_{ii} = \sum_{j \in V} A_{ij}$;
Compute transition matrix $P = D^{-1}A$;
Initialize $Y^{(0)} = Y_L$;
repeat
  $Y^{(t+1)} \leftarrow PY^{(t)}$;
  $Y_L^{(t+1)} \leftarrow Y_L^{(t)}$;
until $Y^{(t)}$ *converges*;
return $\hat{Y} \leftarrow Y^{(t)}$;

By regarding $Y_i^{(t)}$ as the representation of node i at iteration t, we can replace the transition matrix P in Alg.2 with a neural network layer-like differentiable function with trainable parameters, as shown in Eq.11.

$$Y_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} \underbrace{\mathcal{F}(Y_i^{(l)}, Y_j^{(l)})}_{P_{ij}} Y_j^{(l)}\right) \tag{11}$$

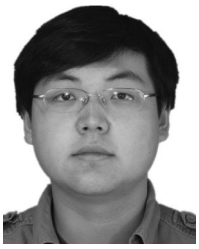
where $Y_{i \in V}$ is the attribute of node i, $\mathcal{N}_i$ is the neighbor set of node i including itself, c_{ij} is an appropriately chosen normalization constant for the edge between node i and j. Further, $Y_i^{(l+1)}$ is transformed to a vector of activations of node i from the l^{th} neural network layer. Please note that Eq.11 is actually the layer function $h_i' \leftarrow \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{ij} h_j\right)$ in GAT. Here, α_{ij} is the aggregation weight from node j to node i. Hence, we can make a (loosely speaking) conclusion that, GAT model can be interpreted as a differentiable and parameterized generalization of the label propagation algorithm.

REFERENCES

[1] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, Jul. 2017.
[2] M. Henaff, J. Bruna, and Y. LeCun, "Deep convolutional networks on graph-structured data," 2015, *arXiv:1506.05163*. [Online]. Available: https://arxiv.org/abs/1506.05163
[3] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," 2013, *arXiv:1312.6203*. [Online]. Available: https://arxiv.org/abs/1312.6203
[4] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 3844–3852.
[5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*. [Online]. Available: https://arxiv.org/abs/1609.02907
[6] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1024–1034.
[7] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–12. [Online]. Available: https://openreview.net/forum?id=rJXMpikCZ
[8] J. Chen, T. Ma, and C. Xiao, "FastGCN: Fast learning with graph convolutional networks via importance sampling," 2018, *arXiv:1801.10247*. [Online]. Available: https://arxiv.org/abs/1801.10247
[9] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2017.
[10] S. Fortunato, "Community detection in graphs," *Phys. Rep.*, vol. 486, nos. 3–5, pp. 75–174, 2010.
[11] P.-Y. Chen, S. Choudhury, and A. O. Hero, "Multi-centrality graph spectral decompositions and their application to cyber intrusion detection," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 4553–4557.
[12] P.-Y. Chen and L. Wu, "Revisiting spectral graph clustering with generative community models," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2017, pp. 51–60.
[13] E. Abbe, "Community detection and stochastic block models: Recent developments," *J. Mach. Learn. Res.*, vol. 18, no. 177, pp. 1–86, Jan. 2018.
[14] P.-Y. Chen and A. O. Hero, "Multilayer spectral graph clustering via convex layer aggregation: Theory and algorithms," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 3, pp. 553–567, Sep. 2017.
[15] V. S. Dave, B. Zhang, P.-Y. Chen, and M. A. Hasan, "Neural-brane: Neural Bayesian personalized ranking for attributed network embedding," 2018, *arXiv:1804.08774*. [Online]. Available: https://arxiv.org/abs/1804.08774
[16] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. W. Schuller, "A deep semi-NMF model for learning hidden representations," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1692–1700.
[17] G. Trigeorgis, K. Bousmalis, S. Zafeiriou, and B. W. Schuller, "A deep matrix factorization method for learning attribute representations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 3, pp. 417–429, Mar. 2017.
[18] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," 2018, *arXiv:1812.04202*. [Online]. Available: https://arxiv.org/abs/1812.04202
[19] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Trans. Neural Netw.*, vol. 20, no. 1, pp. 61–80, Jan. 2009.
[20] T. N. Kipf and M. Welling, "Variational graph auto-encoders," 2016, *arXiv:1611.07308*. [Online]. Available: https://arxiv.org/abs/1611.07308
[21] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 1225–1234.
[22] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang, "MGAE: Marginalized graph autoencoder for graph clustering," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 889–898.
[23] S. Pan, R. Hu, S.-F. Fung, G. Long, J. Jiang, and C. Zhang, "Learning graph embedding with adversarial training methods," 2019, *arXiv:1901.01250*. [Online]. Available: https://arxiv.org/abs/1901.01250
[24] F. Monti, M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3697–3707.
[25] J. You, R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec, "GraphRNN: Generating realistic graphs with deep auto-regressive models," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 5694–5703.
[26] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, "Structured sequence modeling with graph convolutional recurrent networks," in *Proc. Int. Conf. Neural Inf. Process.* Cham, Switzerland: Springer, 2018, pp. 362–373.

[27] N. De Cao and T. Kipf, "MolGAN: An implicit generative model for small molecular graphs," 2018, *arXiv:1805.11973*. [Online]. Available: https://arxiv.org/abs/1805.11973
[28] Z. Zhou and X. Li, "Graph convolution: A high-order and adaptive approach," 2017, *arXiv:1706.09916*. [Online]. Available: https://arxiv.org/abs/1706.09916
[29] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," 2019, *arXiv:1901.00596*. [Online]. Available: https://arxiv.org/abs/1901.00596
[30] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1993–2001.
[31] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 1263–1272.
[32] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for Web-scale recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 974–983.
[33] F. Chen, S. Pan, J. Jiang, H. Huo, and G. Long, "DAGCN: Dual attention graph convolutional networks," 2019, *arXiv:1904.02278*. [Online]. Available: https://arxiv.org/abs/1904.02278
[34] P. W. Battaglia *et al.*, "Relational inductive biases, deep learning, and graph networks," 2018, *arXiv:1806.01261*. [Online]. Available: https://arxiv.org/abs/1806.01261
[35] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," 2019, *arXiv:1906.00121*. [Online]. Available: https://arxiv.org/abs/1906.00121
[36] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "CayleyNets: Graph convolutional neural networks with complex rational spectral filters," 2017, *arXiv:1705.07664*. [Online]. Available: https://arxiv.org/abs/1705.07664
[37] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," 2018, *arXiv:1801.07606*. [Online]. Available: https://arxiv.org/abs/1801.07606
[38] P.-Y. Chen, B. Zhang, and M. Al Hasan, "Incremental eigenpair computation for graph Laplacian matrices: Theory and applications," *Social Netw. Anal. Mining*, vol. 8, no. 1, p. 4, 2018.
[39] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowl.-Based Syst.*, vol. 151, pp. 78–94, Jul. 2018.
[40] C. Yang, M. Sun, Z. Liu, and C. Tu, "Fast network embedding enhancement via high order proximity approximation," in *Proc. 26th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2017, pp. 3894–3900.
[41] U. Soni, M. Bhambhani, and M. M. Khapra, "Network embedding using hierarchical feature aggregation," in *Proc. ICLR Workshop*, 2018. [Online]. Available: https://openreview.net/forum?id=rkhNTYJwf
[42] C. Ding, T. Li, and M. I. Jordan, "Convex and semi-nonnegative matrix factorizations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 1, pp. 45–55, Jan. 2010.
[43] X. Huang, J. Li, and X. Hu, "Label informed attributed network embedding," in *Proc. 10th ACM Int. Conf. Web Search Data Mining*, 2017, pp. 731–739.
[44] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 701–710.
[45] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech., Theory Exp.*, vol. 2008, no. 10, 2008, Art. no. P10008.
[46] M. E. J. Newman, "Modularity and community structure in networks," *Proc. Nat. Acad. Sci. USA*, vol. 103, no. 23, pp. 8577–8582, 2006.

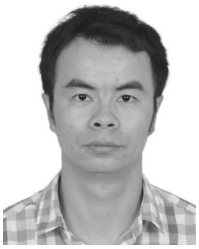
ZHINING LIU received the bachelor's degree in communication engineering from the University of Electronic Science and Technology of China (UESTC), in 2014, where he is currently pursuing the Doctoral degree in information and communication engineering. His current research interests include deep learning on network embedding and seismic image processing.



WEIYI LIU received the Ph.D. degree in communication and information systems from the University of Electronic Science and Technology of China (UESTC), in 2019. He is currently a Principle Data Scientist with JD Intelligent Cities Business Unit, China. His current interests include multilayer network embedding, community detection algorithms, and social network analysis.

PIN-YU CHEN received the B.S. degree in electrical engineering and computer science (Undergraduate Honors Program) from National Chiao Tung University, Taiwan, in 2009, the M.S. degree in communication engineering from National Taiwan University, Taiwan, in 2011, the Ph.D. degree in electrical engineering and computer science, and the M.A. degree in statistics from the University of Michigan, Ann Arbor, MI, USA, in 2016. He is currently a Research Staff Member with the AI Foundations Learning Group, IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA. His recent research is on adversarial machine learning and robustness analysis of neural networks; moreover, his research interests include graph and network data analytics and their applications to data mining, machine learning, signal processing, and cyber security. He is also a member of the Tau Beta Pi Honor Society and the Phi Kappa Phi Honor Society. He was a recipient of the Chia-Lun Lo Fellowship from the University of Michigan Ann Arbor, the NIPS 2017 Best Reviewer Award, and the IEEE GLOBECOM 2010 GOLD Best Paper Award and several travel grants, including the IEEE ICASSP 2014 (NSF), the IEEE ICASSP 2015 (SPS), the IEEE Security and Privacy Symposium, the NSF Graph Signal Processing Workshop 2016, and the ACM KDD 2016.

CHENYI ZHUANG received the B.S. degree in SE from Nanjing University, in 2011, and the M.S. and Ph.D. degrees in informatics from Kyoto University, in 2014 and 2017, respectively. He joined Ant Financial, as an algorithm Expert, in June 2019. Prior to that, he was a Researcher with the Artificial Intelligence Research Center (AIRC), AIST, in Japan. From 2015 to 2018, he was also serving as a Young Scientist in Japan Society for the Promotion of Science (JSPS). His current research primarily involves structured data mining, machine learning, and urban computing.



CHENGYUN SONG was born in Gansu, China, in 1988. He received the M.S. degree in control theory and control engineering from the Lanzhou University of Technology, in 2010, and the Ph.D. degree in information and communication engineering from the University of Electronic Science and Technology of China, in 2017. Since 2018, he has been a Lecturer with the School of Computer Science and Engineering of Chongqing University of Technology, China. His research interests mainly include signal processing, data mining, and pattern recognition.