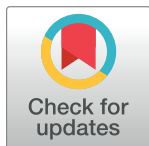


RESEARCH ARTICLE

Path-enhanced graph convolutional networks for node classification without features

Qingju Jiao^{1*}, Peige Zhao², Hanjin Zhang³, Yahong Han⁴, Guoying Liu⁵

1 School of Computer and Information Engineering, Anyang Normal University, and Key Laboratory of Oracle Bone Inscriptions Information Processing, Ministry of Education of China, Anyang, Henan, China, **2** School of Computer and Artificial Intelligence, Zhengzhou University, Zhengzhou, Henan, China, **3** School of Software and Internet of Things Engineering, Jiangxi University of Finance and Economics, Nanchang, Jiangxi, China, **4** College of Intelligence and Computing, Tianjin University, Tianjin, China, **5** School of Software Engineering, Anyang Normal University, and Key Laboratory of Oracle Bone Inscriptions Information Processing, Ministry of Education of China, Anyang, Henan, China

* qjjiao@aynu.edu.cn

Abstract

Most current graph neural networks (GNNs) are designed from the view of methodology and rarely consider the inherent characters of graph. Although the inherent characters may impact the performance of GNNs, very few methods are proposed to resolve the issue. In this work, we mainly focus on improving the performance of graph convolutional networks (GCNs) on the graphs without node features. In order to resolve the issue, we propose a method called *t-hop*GCN to describe *t-hop* neighbors by the shortest path between two nodes, then the adjacency matrix of *t-hop* neighbors as features to perform node classification. Experimental results show that *t-hop*GCN can significantly improve the performance of node classification in the graphs without node features. More importantly, adding the adjacency matrix of *t-hop* neighbors can improve the performance of existing popular GNNs on node classification.

OPEN ACCESS

Citation: Jiao Q, Zhao P, Zhang H, Han Y, Liu G (2023) Path-enhanced graph convolutional networks for node classification without features. PLoS ONE 18(6): e0287001. <https://doi.org/10.1371/journal.pone.0287001>

Editor: Toqir Rana, The University of Lahore, PAKISTAN

Received: January 7, 2023

Accepted: May 29, 2023

Published: June 9, 2023

Copyright: © 2023 Jiao et al. This is an open access article distributed under the terms of the [Creative Commons Attribution License](https://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Data Availability Statement: All relevant data are within the paper and its [Supporting Information](#) files.

Funding: Henan Provincial Colleges and Universities Youth Key Teacher Training Plan (No. 2021GGJS129) and the National Natural Science Foundation of China (No. 61806007).

Competing interests: The authors have declared that no competing interests exist.

1. Introduction

Deep learning models have been successfully applied to different fields, such as computer vision [1], natural language processing [2] and false data injection attack [3]. However, these models do not handle the graph data which can easily describe many real complex systems in biology, physics, sociology and computer science. Graph Neural Networks (GNNs) that combine the paradigm of deep learning can deal with the tasks on graph data [4]. Graph convolutional networks (GCNs) [5] are typical and successful models of GNNs and undergo rapid development over the past few years.

A lot of deuterogenic GCNs are proposed to improve the performance and apply to different fields. Pei et al propose a geometric aggregation scheme (termed Geom-GCN) [6] to improve the performance of GCN. Geom-GCN can overcome the losses of discriminative structures and long-range dependencies by aggregating the structural neighborhoods in latent space. As with Geom-GCN, the method WGCN proposed by Zhao et al embeds graphs into a

latent space, and gains geometrical relationships of nodes [7]. To improve the classification accuracy of GCN, Wang et al propose an adaptive multi-channel graph convolutional network (AM-GCN) [8]. AM-GCN employs attention mechanism to learn adaptive weights of the embeddings from nodes features and topological structures. Yang et al propose a factorizable graph convolutional network (FactorGCN) [9] to produce disentangled node features which is used for graph and node classification. FactorGCN disentangles inputted graph into several factorized graphs which correspond to several latents, and aggregates nodes in each latent to produce new features. Using a feature similarity preserving aggregation which can fuse graph structure and node features, SimP-GCN [10] is proposed to improve the performance of GCN. Furthermore, SimP-GCN also can acquire the feature similarity and dissimilarity relations between nodes by self-supervised learning.

The GCN proposed by Kipf et al [5] only uses two convolutional layers, but shallow layers may not capture deeper topology structure and the information of high-order neighbors [11]. But deep GCNs suffer from over smoothing and over fitting. By removing certain edges which makes the connections between nodes more sparse and generates more diversity into the graph, DropEdge [12] proposed by Rong et al can alleviate both over smoothing and over fitting issues in deep GCN. Chen et al employ initial residual and identity mapping to design a deep GCNII model [11]. The model relieves the over smoothing problem on semi and fully supervised tasks. Feng et al propose a graph random neural network (GRAND) [13] to alleviate the over smoothing issue. GRAND first augments graph by a random propagation strategy and then optimizes prediction consistency by consistency regularization. Chen et al propose a residual network structure to resolve over smoothing problem for user-item interaction data [14]. Bo et al propose a frequency adaptation graph convolutional network (FAGCN) which adaptively fuses low-frequency and high-frequency signals to alleviate the over smoothing problem [15]. Yang et al propose multilayer graph convolutional networks with dropout (DGCs) to perform feature augmentation and relieve over fitting problem by performing non-linearity removal and weight matrix merging between graph conventional layers [16].

Existing graph neural networks (GNNs) may suffer from high time complexity and high demand of memory. Wang et al propose a binary graph convolutional network (Bi-GCN) to handle the issue by binarizing network parameters and node features [17]. By considering the random features in speeding up the training, Huang et al propose a graph convolutional network with random weights (GCN-RW) [18], which employs random filters to revising convolutional layer and regularized least squares loss to adjust learning objective. Graph sampling is a classic and effective model to resolve time and memory challenges. Therefore, some sampling-based GCNs are proposed, such as Cluster-GCN [19] and fastGCN [20]. Although graph topology sampling is an effective method to reduce the memory and computational cost in training GCNs, the study of the relationship between them is rare from the view of theory. Therefore, Li et al describe the impact of generalization performance and sample complexity from graph structures and topology sampling [21].

The GCNs mentioned above are designed from the view of methodology. In fact, the intrinsic characters of graph data (such as incompleteness, noise and dynamic) impact the performance of GCNs. In order to overcome the incompleteness and missing, Taguchi et al use gaussian mixture model to represent missing data and calculate the expected activation of neurons, which enable GCN to resolve the issue mentioned above [22]. Gan et al propose a dynamic graph convolutional network to obtain high-quality data from original graph by fusing multiple local and global graphs, and then perform GCN in a low-dimensional space [23]. Pareja et al propose evolving graph convolutional networks for dynamic graphs (EvolveGCN) [24]. EvolveGCN uses recurrent neural network (RNN) to describe dynamic graphs by evolving network parameters.

Most node classification methods using GNNs work well by aggregating adjacent node features iteratively [25, 26]. However, a large number of graphs do not contain node features. For example, a classical graph without node features is a molecular graph in which nodes and edges represent atoms and chemical bonds respectively [27, 28]. In the social field, the graphs in the REDDIT data [29, 30] also do not include node features. The nodes in these graphs are users and the edges represent the mutual relationship of comments. Unfortunately, current GNNs cannot obtain excellent performance on the graphs without node features [31]. In this study, we focus on the node classification using GCNs without node features. To improve the performance of GCNs without node features, it is necessary to extract more information of adjacent nodes through the graph topology, such as 2-hop neighbors or farther hop neighbors. In fact, the message between adjacent nodes is passed along edge paths [12]. To resolve the issue mentioned above, we introduce *t*-hop neighbors [32], which are generated by edge paths as feature matrix of GCNs, to capture more adjacent information. Different from GCNs, the input feature matrix of the proposed method (named *t*-hopGCN) is a *t*-hop adjacency matrix instead of an identity matrix. Experimental results show that the proposed method *t*-hopGCN can significantly improve the performance of node classification in the graph without node features. The main contributions of this study can be summarized as follows. First, we extract a new feature matrix from graph structure by *t*-hop neighbors introduced in this work. The new feature matrix can provide a universal guideline to extract node features from graph information including neighborhoods and path. Second, a novel approach *t*-hopGCN for node classification is proposed, and *t*-hopGCN outperforms other GNNs by a large margin. And finally, the performance of 12 GNNs or variants are improved obviously on the graphs without node features by adding the *t*-hop feature matrix, indicating that our research can be used as a general skill to improve the performance.

The rest of this work is organized as follows. In section 2, we introduce the principles of GCN and the proposed *t*-hopGCN in detail. Section 3 provides the comparative experimental results on six graphs data. Then, the performance of different GNNs on node classification by adding *t*-hop matrix features is investigated in section 4. Section 5 discusses the selection of a parameter in *t*-hopGCN. Finally, Section 6 concludes this work and provides some directions for future works.

2. Methods

2.1. Graph convolutional networks

Here, we first introduce some basic concepts of a graph. A graph G with n nodes and m edges is described as $G = (V, E)$, V and E are the sets of nodes and edges in the graph, respectively. If each node v_i has d features, the graph can also be represented as $G = (V, E, \mathbf{X})$, where $\mathbf{X} = [x_1, x_2, \dots, x_i, \dots, x_n]^T \in R^{n \times d}$ is the feature matrix for n nodes, and $x_i \in R^d$ is the d dimensional feature vector of node v_i . To calculate conveniently, a graph is usually written in the form of adjacency matrix \mathbf{A} . If there is an edge between node v_i and node v_j , then $\mathbf{A}(i, j)$ is the weight of the edge; otherwise $\mathbf{A}(i, j) = 0$.

GCN is a classic model of GNNs and describes node features by aggregating the features from its neighbors. A main content of GCNs is a layer-wise propagation rule for neural network models described as Eq (1) [5]:

$$\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{D}}^{-1/2} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-1/2} \mathbf{H}^{(l)} \mathbf{W}^{(l)}) \tag{1}$$

$\mathbf{H}^{(l)}$ is the matrix of activations in the l^{th} layer; $\mathbf{H}^{(0)} = \mathbf{X}$, $\sigma(\cdot)$ denotes an activation function, such as the $ReLU(\cdot) = \max(0, \cdot)$; $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ (\mathbf{I} is an identity matrix) and $\tilde{\mathbf{D}}$ is a degree

matrix, $\tilde{\mathbf{D}}_{ii} = \sum_{j=1}^n \tilde{\mathbf{A}}_{ij}$; $\mathbf{W}^{(l)} \in R^{d \times f}$ with d dimensional feature vector and f filters is a trainable weight matrix in layer l .

The graph convolutional network is applied for semi-supervised classification by a two-layer graph convolutional networks with a softmax (Eq (2)) classifier on the output features.

$$Z = \text{softmax}(\hat{\mathbf{A}}\text{ReLU}(\hat{\mathbf{A}}\mathbf{X}\mathbf{W}^{(0)})\mathbf{W}^{(1)}) \tag{2}$$

where $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}$, $\text{softmax}(x_i) = \frac{1}{\sum_j \exp(x_j)}$, the weights $\mathbf{W}^{(0)}$ and $\mathbf{W}^{(1)}$ are trained using gradient descent.

The loss function is defined as the cross-entropy loss over all labeled nodes (Eq (3)):

$$L = -\sum_{l \in y_L} \sum_{f=1}^F Y_{lf} \ln Z_{lf} \tag{3}$$

where y_L is the set of node indices with labels, F is the dimension of the output features and is equal to the number of classes. $Y \in R^{y_L \times F}$ is a label indicator matrix.

2.2. Our method *t*-hopGCN

We first introduce the *t*-hop (or *t*-order) neighbors (N_i^t) of a node v_i by edges path like the node’s local neighborhood defined by Hamilton [33]. For a node v_i , N_i^t is the set of nodes whose shortest path (d_{sp}) to node v_i is less than or equal to t (see Eq (4)).

Fig 1 illustrates the *t*-hop neighbors. For example, the 1-hop neighbors of node 1 are nodes 2 and 3. Then, we define the adjacency matrix of *t*-hop neighbors on graph (\mathbf{M}^{t-hop}). In order to enhance the role of path in node feature, the element of \mathbf{M}^{t-hop} is set the shortest path between two nodes (see Eq (5), Fig 2). Fig 2A is a graph and its adjacency matrices. Fig 2B illustrates the 0-hop neighbors for each node and its adjacency matrix \mathbf{M}^{0-hop} . In fact, the adjacency matrix for 0-hop neighbors and adjacency matrix (\mathbf{M}^{1-hop}) for 1-hop neighbors (Fig 2C) are the identity matrix and adjacency matrix of the graph, respectively. Fig 2D and 2E represent the adjacency matrix of 2-hop neighbors and adjacency matrix of 3-hop neighbors, respectively.

$$N_i^t = \begin{cases} \{v_j | v_j \in V, i \neq j \text{ and } d_{sp}(i, j) \leq t\} & t > 0 \\ \{v_i\} & t = 0 \end{cases} \tag{4}$$

$$\mathbf{M}_{ij}^{t-hop} = \begin{cases} d_{sp}(i, j) & i \neq j \\ 0 & i = j \end{cases} \tag{5}$$

To generate a new feature matrix \mathbf{Y} from \mathbf{X} by using one layer GCN, we can rewrite the Eq (6) by setting $\hat{\mathbf{A}} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2} = \tilde{\mathbf{D}}^{-1/2}(\mathbf{A} + \mathbf{I})\tilde{\mathbf{D}}^{-1/2}$, and the Eq (6) can be represented by Eq (7).

$$\mathbf{Y} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{X} \tag{6}$$

$$\mathbf{Y} = \hat{\mathbf{A}}\mathbf{X} \tag{7}$$

where $\hat{\mathbf{A}}$ is a convolutional matrix. Clearly, the graph convolution is the key to the huge performance gain because GCN mixes the features of a vertex and its nearby neighbors [26]. In fact, the role of matrices $\tilde{\mathbf{D}}$ and \mathbf{I} are to normalize the adjacency matrix \mathbf{A} . Therefore, we can

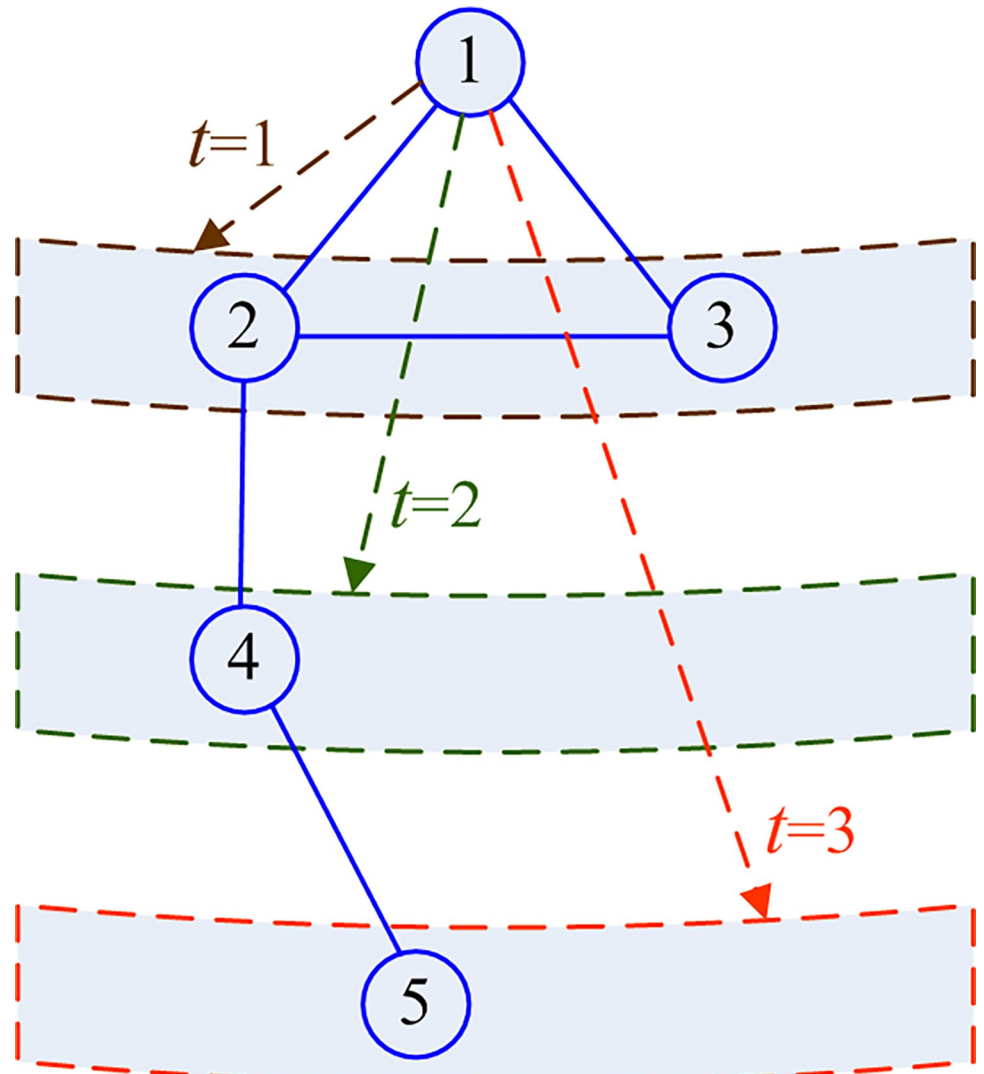


Fig 1. An example of *t*-hop neighbors. The 1-hop neighbors of node 1 are the nodes 2 and 3, $N_1^2 = \{2, 3, 4\}$ and $N_1^3 = \{2, 3, 4, 5\}$.

<https://doi.org/10.1371/journal.pone.0287001.g001>

simplify the Eq (7) by replacing \hat{A} with A (see Eq (8)).

$$Y = AX \tag{8}$$

From Eq (8), we can obtain the value of Y_{ij} (see Eq (9)) that aggregate the sum of the neighbor's features.

$$Y_{ij} = \sum_h^{N_i} A_{ih} \times X_{hj} \tag{9}$$

where N_i is the neighbors of node v_i . If the nodes in the graph do not have features, the feature matrix in GCN is set as identity matrix (Eq (10)).

$$Y = AI \tag{10}$$

The Eq (10) can be rewrite by an adjacency matrix of 0-hop M^{0-hop} (see Eq (11)), and the

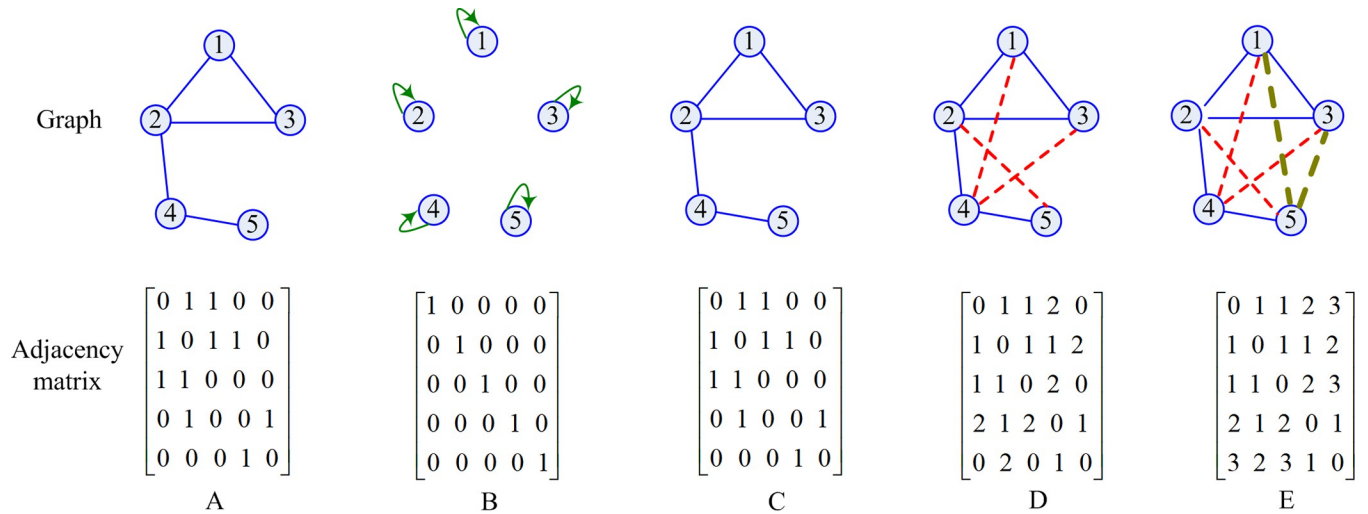


Fig 2. The t -hop neighbors and adjacency matrices.

<https://doi.org/10.1371/journal.pone.0287001.g002>

element can be rewrite by Eq (12).

$$\mathbf{Y} = \mathbf{A}\mathbf{M}^{0-hop} \tag{11}$$

$$\mathbf{Y}_{ij} = \sum_l^{N_{ij}^0} \mathbf{A}_{il} \times \mathbf{M}_{lj}^{0-hop} \tag{12}$$

In Eq (12), N_{ij}^0 represent the intersection of N_i^0 and N_j^0 , N_i^0 and N_j^0 are 0-hop neighbors of nodes v_i and v_j , respectively.

From Eq (11) and Eq (12), we can see that GCNs only capture self-feature when nodes do not have features. Thus, GCN cannot perform well on the graphs without nodes features. In this work, we use adjacency matrix \mathbf{M}^{t-hop} as feature matrix for GCNs because the high hop of \mathbf{M}^{t-hop} can capture more information on neighbors. Eq (13) shows the element \mathbf{Y}_{ij} in our method. The feed forward propagation in our method is described as Eq (14).

$$\mathbf{Y}_{ij} = \sum_l^{N_{ij}^t} \mathbf{A}_{il} \times \mathbf{M}_{lj}^{t-hop} \tag{13}$$

$$\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{D}}^{-1/2} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-1/2} (\mathbf{M}^{t-hop})^{(l)} \mathbf{W}^{(l)}) \tag{14}$$

3. Results

In order to verify the effectiveness of our method t -hopGCN, 12 methods including GCN [5], FastGCN [20], GAT [34], SGC [35], ClusterGCN [19], DAGNN [25], APPNP [36], SSGC [37], GraphMLP [38], RobustGCN [39], LATGCN [40] and MedianGCN [41] are tested on six widely used datasets (see S1 Datasets). These six datasets are Cora, Citeseer and Pubmed [42], Karate [43], Dolphins [44] and Polbook (<http://www-personal.umich.edu/~mejn/netdata/>, Books about US politics). The former three datasets are citation networks in which each node has label and features. The later three datasets are graphs with strong community structure and the nodes in these graphs have no labels and features. In the study, we treat the nodes in the same community with the same class. For Citeseer graph, we only select the nodes with labels and features, as a results, the Citeseer graph includes 3312 nodes. Table 1 illustrates some characteristics of six graphs. Note that, if the graph is disconnected, the diameter is the

Table 1. Some characteristics of six graphs.

Graphs	Nodes	Edges	Classes	Diameter	Connectivity
Cora	2708	5429	7	19	No
Citeseer	3312	4732	6	28	No
Pubmed	19717	44338	3	18	Yes
Karate	34	78	2	5	Yes
Dolphins	62	159	2	8	Yes
Polbook	105	441	3	7	Yes

<https://doi.org/10.1371/journal.pone.0287001.t001>

maximum of diameters of all connected components, and the average path length is the mean of the average path lengths of all connected components.

First, the feature matrices (see [S2 Datasets](#)) are set as identity matrices for 12 baseline methods and the adjacency matrix of t -hop neighbors for t -hopGCN. The parameters in the 12 methods are default in GraphGallery [45], which is an easy-to-use platform for fast benchmarking and easy development of graph neural networks. Then, for Cora, Citeseer and Pubmed graphs, the order of the nodes in the feature matrix is the same as the order of the nodes in the original data, and we evaluate t -hopGCN and 12 methods with 5% of the training size and 10% of the test size, respectively. For other three small graphs, the order of the nodes in the feature matrices are rearranged by their classes, where we rank the nodes alternately using the labels of classes. Since the sizes of the three graphs are small, we evaluate t -hopGCN and 12 methods with 20% of the training size and 20% of the test size, respectively. The accuracy and (weighted) F-score [46] of t -hopGCN and 12 methods are shown in Tables 2 and 3.

For the Cora, Citeseer and Pubmed without community structure, our method t -hopGCN outperforms other methods significantly on accuracy (see [Table 2](#)) and F-score (see [Table 3](#)). Using accuracy, t -hopGCN improves over GCN by 20.37% on Cora, and improves over the worst LATGCN by 22.6% and the best GAT by 2.6%, respectively. On Citeseer, t -hopGCN improves over RobustGCN with the worst performance by 32.32% and GraphMLP with the best performance by 19.94%, respectively. For Pubmed, the two relative increases are 35.35% and 24.24%, respectively. Likewise, t -hopGCN shows promising results of F-score compared with other methods (see [Table 3](#)). For instance, t -hopGCN gives 15.16%, 32.06% and 37.77% relative improvements over the worst methods (LATGCN, RobustGCN and MedianGCN) on

Table 2. The accuracy of different methods.

Methods	Cora	Citeseer	Pubmed	Karate	Dolphins	Polbook
t -hopGCN	0.3630	0.4350	0.7713	0.5714	0.7692	0.4545
GCN	0.1593	0.1722	0.5289	0.4286	0.3846	0.5455
FastGCN	0.2333	0.1390	0.5005	0.5714	0.6154	0.4545
GAT	0.3370	0.1208	0.4894	0.5714	0.3846	0.5000
SGC	0.3000	0.1178	0.5066	0.5714	0.5385	0.4545
ClusterGCN	0.3037	0.1752	0.5132	0.8571	0.5385	0.5000
DAGNN	0.2444	0.1601	0.5051	0.5714	0.3846	0.5000
APPNP	0.3148	0.1329	0.5030	0.5714	0.7692	0.5455
SSGC	0.2926	0.1329	0.5259	0.5714	0.3846	0.4545
GraphMLP	0.1630	0.2356	0.4178	0.8571	0.6154	0.4545
RobustGCN	0.2815	0.1118	0.4772	0.8571	0.3077	0.5000
LATGCN	0.1370	0.1420	0.5254	0.4286	0.3846	0.2727
MedianGCN	0.1556	0.1269	0.4214	0.4286	0.7692	0.4545

<https://doi.org/10.1371/journal.pone.0287001.t002>

Table 3. The F-score of different methods.

Methods	Cora	Citeseer	Pubmed	Karate	Dolphins	Polbook
<i>t-hop</i> GCN	0.2875	0.3994	0.7642	0.5143	0.6689	0.3961
GCN	0.1723	0.1586	0.5042	0.2571	0.4308	0.5076
FastGCN	0.1671	0.1182	0.4014	0.5143	0.5861	0.4545
GAT	0.1718	0.0900	0.4075	0.5143	0.4308	0.3772
SGC	0.1850	0.0904	0.4652	0.5143	0.5549	0.2841
ClusterGCN	0.2005	0.1595	0.4880	0.8571	0.5705	0.4842
DAGNN	0.2287	0.1219	0.4926	0.5143	0.4308	0.3636
APNP	0.1834	0.1155	0.4395	0.5143	0.6689	0.5455
SSGC	0.2354	0.1142	0.4804	0.5143	0.4066	0.4545
GraphMLP	0.1741	0.2393	0.4436	0.8571	0.6584	0.4569
RobustGCN	0.1737	0.0788	0.4136	0.7143	0.2582	0.5573
LATGCN	0.1359	0.1234	0.5044	0.2571	0.4359	0.3000
MedianGCN	0.1639	0.0912	0.3865	0.2571	0.7516	0.2841

<https://doi.org/10.1371/journal.pone.0287001.t003>

Cora, Citeseer and Pubmed respectively. The three relative improvements over the best methods (SSGC, GraphMLP and LATGCN) are 5.21%, 16.01% and 25.98%, respectively.

On Dolphins data, *t-hop*GCN achieves the highest accuracy of 76.92%. Using the F-score, the performance *t-hop*GCN is 66.89% slightly lower than MedianGCN with the best performance. For accuracy and F-score, *t-hop*GCN does not perform better than other methods on Karate and Polbook. The potential reason is that the three graphs have strong community structure [47], but *t-hop*GCN cannot capture it well.

4. *t-hop* features improve different GNNs

In this section, we investigate if adding *t-hop* features can improve the performance of popular GNNs on node classification. Here, 12 original methods are compared by adding *t-hop* matrix features. Fig 3 and S1 Fig (see Supporting information) show the accuracy and F-score

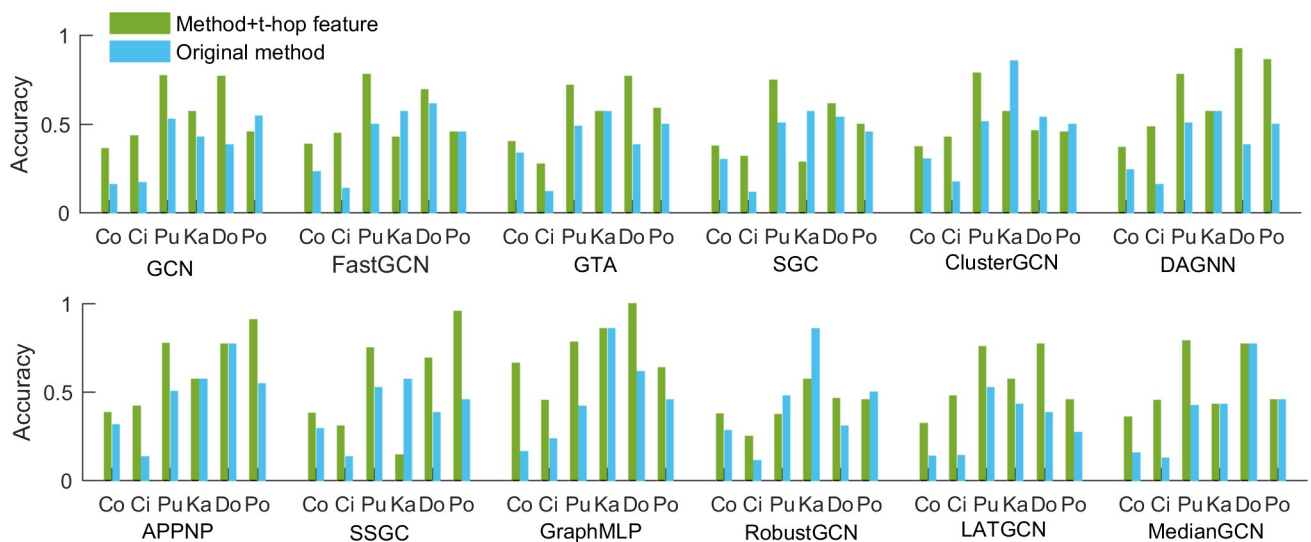


Fig 3. Accuracy comparison between original methods and modified methods with *t-hop* features, Co, Ci, Pu, Ka, Do and Po represent Cora, Citeseer and Pubmed, Karate, Dolphins and Polbook.

<https://doi.org/10.1371/journal.pone.0287001.g003>

comparison between original methods and the modified methods with *t-hop* feature, and Fig 4 and S2 Fig (see Supporting information) show the accuracy and F-score improvement or decrease by adding *t-hop* features, respectively. From the four figures, we can see that the accuracies and F-scores of 11 methods (except for RobustGCN on Pubmed) are improved remarkably by adding *t-hop* features on Cora, Citeseer and Pubmed. The highest improvements in accuracy and F-score are GraphMLP on Cora data, and the relative increase reaches 50% and 48.84% (see Fig 4 and S2 Fig) respectively. The smallest improvement in accuracy and F-score are GAT on Cora with 6.3% and SGC on Cora with 6.63% respectively.

On karate data with strong community structure, the accuracies of two methods including GCN and LATGCN are improved by 14.28%. The performance of five methods (GAT, DAGNN, APPNP, GraphMLP and MedianGCN) remains the same by adding *t-hop* features, and the rest of five methods yield worse performance after adding *t-hop* features. On the Dolphins data, the accuracies of nine methods are improved, and the best improvement is DAGNN by 53.85%. Unfortunately, the accuracy of ClusterGCN decreases by 7.7% after adding *t-hop* features. For Polbook data, the accuracies of seven methods (GAT, SGC, DAGNN, APPNP, SSGC, GraphMLP and LATGCN) are improved after adding *t-hop* features with a large margin, where the maximal improvement is 50%. However, other three methods perform worse after adding *t-hop* features. Likewise, the performance on F-score has not been improved effectively by adding *t-hop* features on Karate (see S2 Fig), and the F-scores of only three methods are improved. For Dolphins and Polbook, the number of methods whose F-scores are improved significantly by adding *t-hop* features are nine and six, respectively.

Furthermore, we analyze the average accuracies and F-scores from the points of graph data and methods. Fig 5A and S3A Fig (see Supporting information) show the improvement of the average accuracies and F-scores of different methods on six graphs, and Fig 5B and S3B Fig (see Supporting information) show the improvement of the average accuracies and F-scores on different graphs on 12 methods respectively. As shown in the Fig 5A and S3A Fig, we can see that the average accuracy and F-score on Cora, Citeseer, Pubmed, Dolphins and Polbook are improved significantly by adding *t-hop* features, while the average accuracy and F-score on

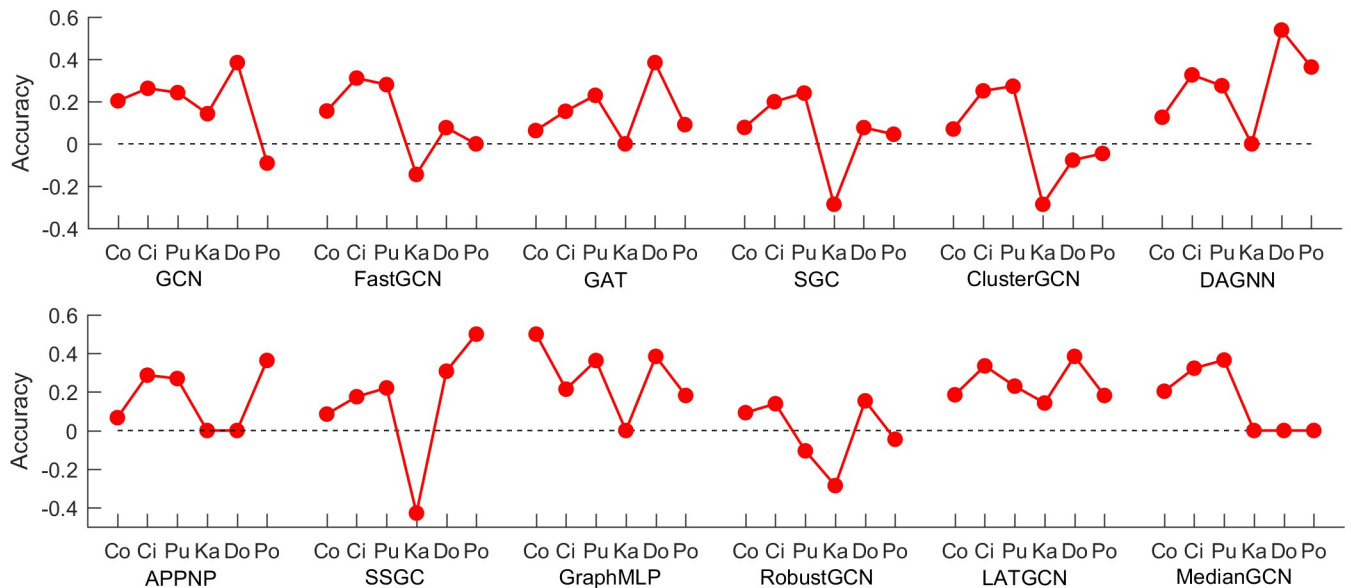


Fig 4. The accuracy improvement or decrease by adding *t-hop* features for 12 methods.

<https://doi.org/10.1371/journal.pone.0287001.g004>

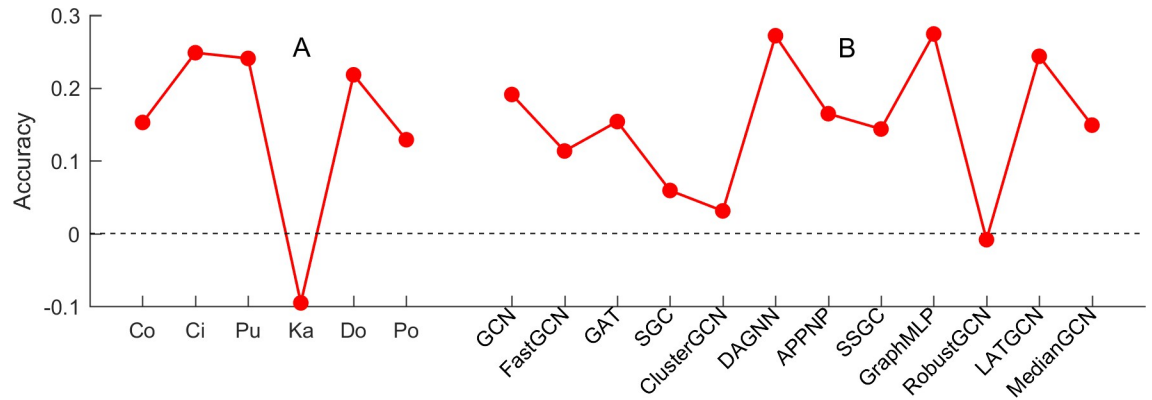


Fig 5. The improvement of average accuracy on six graphs and 12 methods after adding *t*-hop features.

<https://doi.org/10.1371/journal.pone.0287001.g005>

Karate decreases sharply. The potential reason is that Karate data has strong community structure which contains *t*-hop information. As shown in Fig 5B, we can see that the average accuracies of 11 methods are improved significantly. The best improvement is GraphMLP with a relative increase of 27.4%. Only the method RobustGCN with *t*-hop features achieve worse average accuracy by 0.85%. When we use F-score to measure these results, the average F-scores of all 12 methods are improved and the highest improvement achieves 25.49% (see S3B Fig). These results suggest that our study will open a new idea to research node classification in graphs without features.

5. Selection of the parameter *t* in *t*-hopGCN

The parameter *t* plays a vital role in *t*-hopGCN. Here, we investigate the relationship between the parameter *t* and the accuracy and F-score of *t*-hopGCN. If the graph diameter greater than or equal to 10, the parameter *t* is set from 1 to 10 (see Eq (15)). Otherwise, the parameter *t* is set from 1 to the diameter. Fig 6 and S4 Fig (see Supporting information) show the changes of accuracy and F-score of *t*-hopGCN on six graphs by increasing the parameter *t*. Overall, the accuracy and F-score of *t*-hopGCN decrease as the parameter *t* grows. More specifically, *t*-

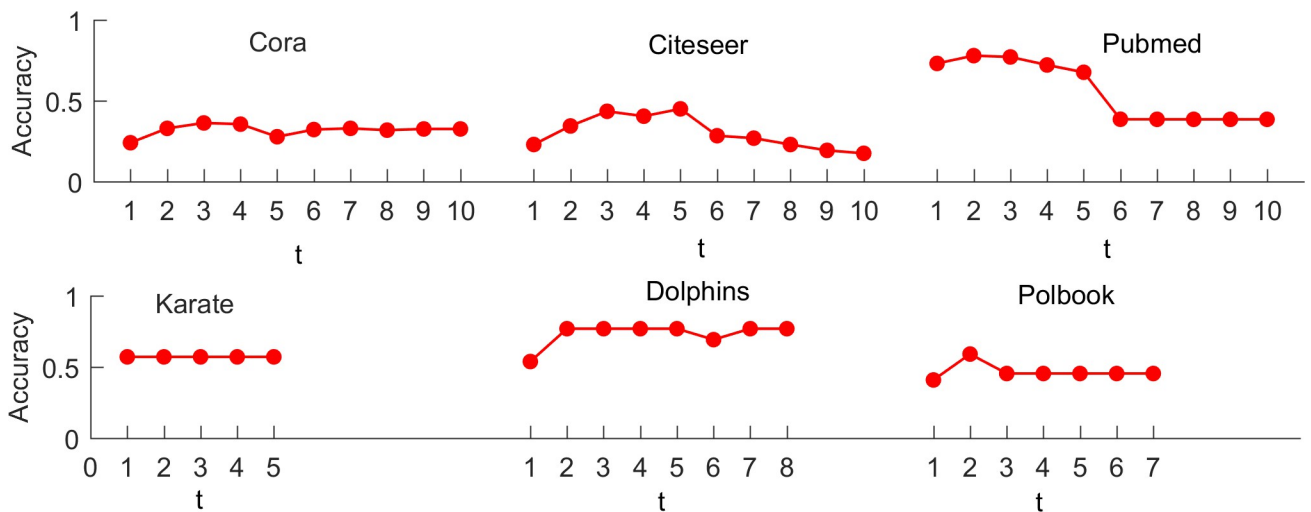


Fig 6. The relationship between the parameter *t* and the accuracy of *t*-hopGCN on six graphs.

<https://doi.org/10.1371/journal.pone.0287001.g006>

*hop*GCN achieves the best performance on accuracy and F-score when $t = 3$ on Cora and Dolphins. Moreover, *t-hop*GCN with $t = 3$ gets the highest value of F-score on Citeseer. *t-hop*GCN achieves the best performance (accuracy and F-score) when $t = 2$ on Pubmed and Polbook. Although the highest value of the accuracy with 77.94% and F-score with 77.62% are appears when the parameter t is set as 2 for Pubmed, the *t-hop*GCN with $t = 3$ achieves a close accuracy of 77.13% and F-score of 76.42% respectively. Similarly, although the parameter t is set 3, Citeseer with 43.5% accuracy does not achieve the highest value, the difference between the accuracy with $t = 3$ and the best accuracy with $t = 5$ is only 1.52%. For Karate, with the increase in the parameter t , the values of the accuracy and F-score remain the same, probably due to strong community structure. In summary, it is reasonable to set the parameter t to 3 for *t-hop*GCN, and a smaller parameter t can also reduce the computation complexity.

$$t = \begin{cases} [1, 2, \dots, 10] & \text{diameter} \geq 10 \\ [1, 2, \dots, \text{diameter}] & \text{diameter} < 10 \end{cases} \quad (15)$$

6. Conclusion

In order to improve the performance of node classification using GCN without node features, we propose a new method named *t-hop*GCN with *t-hop* adjacency matrix as node features. Experimental results show that *t-hop*GCN can significantly improve the performance of node classification on six graphs without node features. For example, on Cora, Citeseer and Pubmed, *t-hop*GCN gains the best accuracy and F-score comparing with other 12 methods, and the best improvements are 35.35% and 37.77%. More importantly, the performance (accuracy and F-score) of 12 GNN methods are improved remarkably by adding *t-hop* features. The highest improvements in accuracy and F-score are GraphMLP on Cora data, and the relative increase reaches 50% and 48.84% respectively. Furthermore, the average accuracies of 11 GNN methods and the average F-scores of 12 GNN methods on six graphs are improved significantly. Thus, the skill for extracting node features from graph structure can be applied to improve the performance of GNNs. It is expected that our research will provide a universal guideline to explore GNNs on the graph without node features for broader potential applications. In future work, we plan to extend the three aspects as following. First, insight into the principle of *t-hop*GCN will be investigated. Second, the relationship between the performance of *t-hop*GCN (or other GNNs methods) and graph structure is still unknown, resulting in poor performance on Karate data by adding *t-hop* features. Third, a pressing problem is to reduce the dimension of the *t-hop* feature matrix that becomes very large and sparse with the increasing size of the graph.

Supporting information

S1 Datasets. Six original graph data.

(ZIP)

S2 Datasets. Input data for different GNNs.

(ZIP)

S1 Fig. F-score comparison between original methods and modified methods with *t-hop* features. Co, Ci, Pu, Ka, Do and Po represent Cora, Citeseer and Pubmed, Karate, Dolphins and Polbook.

(TIF)

S2 Fig. The F-score improvement or decrease by adding *t-hop* features for 12 methods.

(TIF)

S3 Fig. The improvement of average F-score on six graphs and 12 methods after adding *t*-hop features.

(TIF)

S4 Fig. The relationship between the parameter *t* and the F-score of *t*-hopGCN on six graphs.

(TIF)

Acknowledgments

We are grateful to Xiaoyong Pan and Zhan Zhang for their advice on this study.

Author Contributions

Data curation: Peige Zhao.

Formal analysis: Qingju Jiao, Peige Zhao, Hanjin Zhang.

Methodology: Qingju Jiao, Hanjin Zhang, Yahong Han.

Validation: Guoying Liu.

Writing – original draft: Yahong Han.

Writing – review & editing: Qingju Jiao, Guoying Liu.

References

1. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. *NeurIPS 2012: Proceedings of the 25th International Conference on Neural Information Processing Systems*; 2012 Dec 3–6; Nevada, America. New York: Curran Associates Inc.; 2012. p. 1095–105.
2. Lauriola I, Lavelli A, Aielli F. An introduction to Deep Learning in Natural Language Processing: Models, techniques, and tools. *Neurocomputing*. 2022; 470: 443–56.
3. Nur Fathin Najwa Binti Mustaffa S, Farhan M. Detection of False Data Injection Attack using Machine Learning approach. *Mesopotamian J Cybersecurity*. 2022; 2022: 38–46.
4. Zhou J, Cui G, Hu S, Zhang Z, Yang C, Liu Z, et al. Graph neural networks: A review of methods and applications. *AI Open*. 2020; 57–81.
5. Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. *ICLR 2017: International Conference on Learning Representations*; 2017 Apr 24–26; Toulon, France. OpenReview.net; 2017. p. 1–14.
6. Pei H, Wei B, Chang KC, Lei Y, Yang B. Geom-GCN: Geometric Graph Convolutional Network. *ICLR 2020: International Conference on Learning Representations*; 2020 Apr 26–30; Addis Ababa, Ethiopia. OpenReview.net; 2020. p. 1–12.
7. Zhao Y, Qi J, Liu Q, Zhang R. WGCN: Graph Convolutional Networks with Weighted Structural Features. *SIGIR'21: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*; 2021 Jul 11–15; Virtual Event, Canada. New York: Association for Computing Machinery (ACM); 2021. p. 624–33.
8. Wang X, Zhu M, Bo D, Cui P, Shi C, Pei J. 2020. AM-GCN: Adaptive Multi-channel Graph Convolutional Networks. *KDD'20: Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*; 2020 Aug 23–27; Virtual Event, America. New York: Association for Computing Machinery (ACM); 2020. p. 1243–53.
9. Yang Y, Feng Z, Song M, Wang X. Factorizable Graph Convolutional Networks. *NeurIPS 2020: Proceedings of the 34th Conference on Neural Information Processing Systems*; 2020 Dec 6–12; Vancouver, Canada. New York: Curran Associates Inc.; 2020. p. 1–11.
10. Jin W, Derr T, Wang Y, Ma Y, Liu Z, Tang J. Node Similarity Preserving Graph Convolutional Networks. *WSDM'21: The ACM International Conference on Web Search and Data Mining*; 2021 Mar 8–12; Jerusalem, Israel. New York: Association for Computing Machinery (ACM); 2021. p. 148–156.

11. Chen M, Wei Z, Huang Z, Ding B, Li Y. Simple and Deep Graph Convolutional Networks. *ICML'20: Proceedings of the 37th International Conference on Machine Learning*; 2020 Jul 13–18; Online. New York: PMLR; 2020. p. 1725–35.
12. Rong Y, Huang W, Xu T, Huang J. Dropedge: towards deep graph convolutional networks on node classification. *ICLR 2020: International Conference on Learning Representations*; 2020 Apr 26–30; Addis Ababa, Ethiopia. *OpenReview.net*; 2020. p. 1–17.
13. Feng W, Zhang J, Dong Y, Han Y, Luan H, Xu Q, et al. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. *NeurIPS 2020: Proceedings of the 34th Conference on Neural Information Processing Systems*; 2020 Dec 6–12; Vancouver, Canada. New York: Curran Associates Inc.; 2020. p. 1–12.
14. Chen L, Wu L, Hong R, Zhang K, Wang M. Revisiting Graph Based Collaborative Filtering: A Linear Residual Graph Convolutional Network Approach. *AAAI-20: The Thirty-Fourth AAAI Conference on Artificial Intelligence*; 2020 Feb 7–12; New York, America. Menlo Park: Association for the Advancement of Artificial Intelligence (AAAI); 2020. p. 27–34.
15. Bo D, Wang X, Shi C, Shen H. Beyond Low-frequency Information in Graph Convolutional Networks. *AAAI-21: The Thirty-Fifth AAAI Conference on Artificial Intelligence*; 2021 Feb 2–9; Online. Menlo Park: Association for the Advancement of Artificial Intelligence (AAAI); 2021. p. 3950–57.
16. Yang F, Zhang H, Tao S. Simplified multilayer graph convolutional networks with dropout. *Appl Intell.* 2022; 52: 4776–91.
17. Wang J, Wang Y, Yang Z, Yang L, Guo Y. Bi-GCN: Binary Graph Convolutional Network. *CVPR: 2021 Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*; 2021 Jun 20–25; Online. New Jersey: Institute of Electrical and Electronics Engineers (IEEE); 2021. p. 1561–70.
18. Huang C, Li M, Cao F, Fujita H, Li Z, Wu X. Are Graph Convolutional Networks With Random Weights Feasible? *IEEE Trans Pattern Anal Mach Intell.* 2022; 1–18.
19. Chiang W, Liu X, Si S. Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks. *KDD'19: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*; 2019 Aug 4–8; Alaska, America. New York: Association for Computing Machinery (ACM); 2019. p. 257–66.
20. Chen J, Ma T, Xiao C. Fastgcn: fast learning with graph convolutional networks via importance sampling. *ICLR 2018: International Conference on Learning Representations*; 2018 Apr 30–May 3; British Columbia, Canada. *OpenReview.net*; 2018. p. 1–15.
21. Li H, Wang M, Liu S, Chen P, Xiong J. Generalization Guarantee of Training Graph Convolutional Networks with Graph Topology Sampling. *ICML'22: Proceedings of the 39th International Conference on Machine Learning*; 2022 Jul 17–23; Maryland, America. New York: PMLR; 2022. p. 13014–51.
22. Taguchi H, Liu X, Murata T. Graph convolutional networks for graphs containing missing features. *Future Gener Comp Sy.* 2021; 117: 155–68.
23. Gan J, Hu R, Mo Y, Kang Z, Peng L, Zhu Y, et al. Multigraph Fusion for Dynamic Graph Convolutional Network. *IEEE Trans Neural Netw Learn Syst.* 2022; 1–12.
24. Pareja A, Domeniconi G, Chen J, Ma T, Suzumura T, Kanezashi H, et al. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. *AAAI-20: The Thirty-Fourth AAAI Conference on Artificial Intelligence*; 2020 Feb 7–12; New York, America. Menlo Park: Association for the Advancement of Artificial Intelligence (AAAI); 2020. p. 5363–70.
25. Liu M, Gao H, Ji S. Towards Deeper Graph Neural Networks. *KDD'20: Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*; 2020 Aug 23–27; Virtual Event, America. New York: Association for Computing Machinery (ACM); 2020. p. 338–48.
26. Li Q, Han Z, Wu X. Deeper insights into graph convolutional networks for semi-supervised learning. *AAAI-18: The Thirty-Second AAAI Conference on Artificial Intelligence*; 2018 Feb 2–7; Louisiana, America. Menlo Park: Association for the Advancement of Artificial Intelligence (AAAI); 2018. p. 3538–45.
27. Ramakrishnan R, Dral P O, Rupp M, Anatole von Lilienfeld O. Quantum chemistry structures and properties of 134 kilo molecules. *Sci Data* 1. 2014; 140022. <https://doi.org/10.1038/sdata.2014.22> PMID: 25977779
28. Ruddigkeit L, van Deursen R, Blum LC, Reymond J-L. Enumeration of 166 Billion Organic Small Molecules in the Chemical Universe Database GDB-17. *J Chem Inf Model.* 2012; 52 (11): 2864–75. <https://doi.org/10.1021/ci300415d> PMID: 23088335
29. Morris C, Kriege NM, Bause F, Kersting K, Mutzel P, Neumann M. TUDataset: A collection of benchmark datasets for learning with graphs. *ICML Workshop on Graph Representation Learning and Beyond.* 2020.

30. Cui H, Lu Z, Li P, Yang C. On Positional and Structural Node Features for Graph Neural Networks on Non-attributed Graphs. *CIKM '22: Proceedings of the 31st ACM International Conference on Information and Knowledge Management*; 2022 Oct 17–21; Atlanta, America. New York: Association for Computing Machinery (ACM); 2022. p. 3898–902.
31. Chen X, Chen S, Yao J, Zheng H, Zhang Y, Tsang IW. Learning on Attribute-Missing Graphs. *IEEE Trans Pattern Anal Mach Intell.* 2020; 44(2): 740–57.
32. You J, Ying R, Leskovec J. Position-aware Graph Neural Networks. *ICML'19: Proceedings of the 36th International Conference on Machine Learning*; 2019 Jun 9–15; California, America. New York: PMLR; 2019. p. 7134–43.
33. Hamilton WL, Ying R, Leskovec J. Inductive representation learning on large graphs. *NeurIPS 2017: Proceedings of the 31st Conference on Neural Information Processing Systems*; 2017 Dec 4–9; California, America. New York: Curran Associates Inc.; 2017. p. 1–11.
34. Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y. Graph Attention Networks. *ICLR 2018: International Conference on Learning Representations*; 2018 Apr 30–May 3; British Columbia, Canada. *OpenReview.net*; 2018. p. 1–12.
35. Wu F, Souza A, Zhang T, Fifty C, Yu T, Weinberger K. Simplifying Graph Convolutional Networks. *ICML'19: Proceedings of the 36th International Conference on Machine Learning*; 2019 Jun 9–15; California, America. New York: PMLR; 2019. p. 6861–71.
36. Gasteiger J, Bojchevski A, Günnemann S. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. *ICLR 2019: International Conference on Learning Representations*; 2019 May 6–9; New Orleans, America. *OpenReview.net*; 2019. p. 1–15.
37. Zhu H, Koniusz P. Simple Spectral Graph Convolution. *ICLR 2021: International Conference on Learning Representations*; 2021 May 3–7; Online. *OpenReview.net*; 2021. p. 1–15.
38. Hu Y, You H, Wang Z, Wang Z, Zhou E, Gao Y. Graph-MLP: Node Classification without Message Passing in Graph. *arXiv:2106.04051*, [Preprint]. 2021. Available from: <https://arxiv.53yu.com/pdf/2106.04051.pdf>.
39. Zhu D, Zhang Z, Cui P, Zhu W. Robust Graph Convolutional Networks Against Adversarial Attacks. *KDD '19: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*; 2019 Aug 4–8; Alaska, America. New York: Association for Computing Machinery (ACM); 2019. p. 1399–407.
40. Jin H, Zhang X. Latent Adversarial Training of Graph Convolution Networks. *ICML Workshop on Learning and Reasoning with Graph Structured Representations*; 2019. Available from: https://www.cs.uic.edu/~hjin/files/icml_ws_latgcn.pdf.
41. Chen L, Li J, Peng Q, Liu Y, Zheng Z, Yang C. Understanding Structural Vulnerability in Graph Convolutional Networks. *IJCAI-21: The 30th International Joint Conferences on Artificial Intelligence*; 2021 Aug 19–27; Montreal, Canada. Massachusetts: Morgan Kaufmann Publishers; 2021. p. 2249–55.
42. Sen P, Namata G, Bilgic M, Getoor L, Galligher B, Eliassi-Rad T. Collective classification in network data. *AI magazine*, 2008; 29(3): 93.
43. Zachary WW. An information flow model for conflict and fission in small groups. *J Anthropol Res.* 1977; 33(4): 452–73.
44. Lusseau D, Schneider K, Boisseau OJ, Haase P, Slooten E, Dawson SM. The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations. *Behav Ecol Sociobiol.* 2003; 54: 396–405.
45. Li J, Xu K, Chen L, Zheng Z, Liu X. GraphGallery: A Platform for Fast Benchmarking and Easy Development of Graph Neural Networks Based Intelligent Software. *ICSE-Companion: 2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings*; 2021 May 25–28; Madrid, Espana. New Jersey: Institute of Electrical and Electronics Engineers (IEEE); 2021. p. 13–16.
46. Sokolova M, Japkowicz N, Szpakowicz S. Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation. *19th Australian Joint Conference on Artificial Intelligence (AI 2006: Advances in Artificial Intelligence)*; 2006 Dece 4–8; Hobart, Australia, Berlin: Springer; 2006. p. 1015–21.
47. Newman MEJ. Communities, modules and large-scale structure in networks. *Nat Phys* 2012; 8: 25–31.