Federation University

Federation University ResearchOnline

https://researchonline.federation.edu.au

Copyright Notice

This is the peer-reviewed version of the following article:

Xia, F., Wang, L., Tang, T., Chen, X., Kong, X., Oatley, G., & King, I. (2023). CenGCN: Centralized Convolutional Networks with Vertex Imbalance for Scale-Free Graphs. IEEE Transactions on Knowledge and Data Engineering, 1–1.

Available online: https://doi.org/10.1109/TKDE.2022.3149888

Copyright © 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

See this record in Federation ResearchOnline at:
http://researchonline.federation.edu.au/vital/access/HandleResolver/1959.17/191922

CRICOS 00103D RTO 4909

CenGCN: Centralized Convolutional Networks with Vertex Imbalance for Scale-Free Graphs

Feng Xia, *Senior Member, IEEE,* Lei Wang, Tao Tang, Xin Chen, Xiangjie Kong, *Senior Member, IEEE,* Giles Oatley, and Irwin King, *Fellow, IEEE*

Abstract—Graph Convolutional Networks (GCNs) have achieved impressive performance in a wide variety of areas, attracting considerable attention. The core step of GCNs is the information-passing framework that considers all information from neighbors to the central vertex to be equally important. Such equal importance, however, is inadequate for scale-free networks, where hub vertices propagate more dominant information due to vertex imbalance. In this paper, we propose a novel centrality-based framework named CenGCN to address the inequality of information. This framework first quantifies the similarity between hub vertices and their neighbors by label propagation with hub vertices. Based on this similarity and centrality indices, the framework transforms the graph by increasing or decreasing the weights of edges connecting hub vertices and adding self-connections to vertices. In each non-output layer of the GCN, this framework uses a hub attention mechanism to assign new weights to connected non-hub vertices based on their common information with hub vertices. We present two variants CenGCN_D and CenGCN_E, based on degree centrality and eigenvector centrality, respectively. We also conduct comprehensive experiments, including vertex classification, link prediction, vertex clustering, and network visualization. The results demonstrate that the two variants significantly outperform state-of-the-art baselines.

Index Terms—Graph Convolutional Networks, Vertex Centrality, Network Analysis, Graph Learning, Representation Learning

arXiv:2202.07826v1 [cs.LG] 16 Feb 2022

1 Introduction

The graph, as an abstract data type, can represent the complex relationships between objects in many real-world networks. Representative networks include social networks [1], biological networks [2], and academic networks [3]. Numerous studies [4], [5], [6], [7] demonstrate the possibilities of extracting rich information from graph-structured data, thereby realizing many practical applications, including vertex classification and link prediction. However, how to extract useful information from these data remains a challenging issue and is thus worthy of exploration in depth.

Recently, extensive studies [8], [9], [10], [11], [12] have shown that Graph Convolutional Networks (GCNs) are powerful tools for handling graph-structured data and for a wide spectrum of graph-based applications, from recommender systems [13], [14] to knowledge graphs [15], [16]. Existing GCNs adopt an information-passing framework [17], where each vertex aggregates information from its immediate neighbors and itself, and considers information from different vertices equally important. However, such equal importance is counterintuitive when different neighbors pass information with different influence to the central vertex. For instance, a person could connect to both friends and work colleagues in social networks where vertices denote persons. When recognizing one's workplace, the information from colleagues is more relevant than that from friends. In this instance, we need to weight more highly the influence of colleagues. Equally aggregating information, however, fails to capture this differentiation between vertices.

Many complex networks in the real world, such as the Internet and social networks, are scale-free networks [18]. We find in these networks an inequality of information from different vertices because of vertex imbalance. The scale-free property, one of the fundamental macroscopic structures of networks, dictates that the vertex degrees follow a power-law distribution: the probability distribution decreases as the vertex degree increases, with a long tail tending to zero. Therefore, significant vertex imbalance appears in a scale-free network, and only a few vertices are of high degree and regarded as hub vertices or simply hubs. The majority of vertices linking to a high-degree vertex are, however, of low degree, and not highly connected. In scale-free networks, a vertex with a high degree is usually the hub of a community, and the information it contains is more influential than that from vertices with low degrees. For instance, in social media, a celebrity that has a great number of followers can spread more news than a less prestigious person. In this case, if we wish to tap into a users' interests, it would be helpful to consider links to celebrities. Additionally, when modeling the diffusion, more weight should be given to edges connecting hub vertices [19].

For scale-free networks, it is possible to implement new GCNs that capture this differentiation in information passing between vertices. Since vertex centrality can be considered a useful measure of the importance of individual

This work was partially supported by the National Natural Science Foundation of China (62072409) and Zhejiang Provincial Natural Science Foundation (LR21F020003).

- *F. Xia, T. Tang, and G. Oatley are with School of Engineering, IT and Physical Sciences, Federation University Australia, Ballarat, VIC 3353, Australia.*
- *L. Wang and X. Chen are with School of Software, Dalian University of Technology, Dalian 116620, China.*
- *X. Kong is with College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China.*
- *I. King is with Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.*

Corresponding author: Xiangjie Kong; email: xjkong@ieee.org.

vertices [20], we propose a centrality-based Graph Convolution Network. In this paper, we consider various centrality measurements, rather than solely focusing on vertex degree. The degree of a vertex is a centrality measurement. The proposed GCNs allow hub vertices, selected by centrality indices, to pass more information. Thus, each vertex receives more information from the hub vertices.

When designing centrality-based GCNs, we need to address the issue that a vertex with a higher centrality index may be linked to similar vertices and possibly to dissimilar vertices due to its high prestige and popularity [21]. For example, in academia, some distinguished scholars can collaborate with others from outside their research laboratories. In social networks, a celebrity receives a great number of followers, but most of them may have totally different backgrounds.

The relationship between two connected but dissimilar vertices needs to be weakened. When aggregating information, neighboring information with similar features is aggregated to reinforce the correct features to facilitate downstream tasks such as classification, while the opposite effect is achieved if dissimilar information is aggregated. Therefore, different weights are required. To address the issue detailed above, we consider the similarity between vertices in the underlying network structure. We use random walk computation in order to calculate this similarity [22], [23], [24], [25]. Subsequently, a label propagation algorithm is applied over hub vertices to quantify the similarity between vertices and their hub neighbors. Through these quantified similarities and vertex centrality indices, we propose a graph transformation method to increase or decrease the weights of edges connecting hub vertices and to add self-connections to vertices. In the transformed graph, the influence from hub vertices to their similar neighbors is strengthened, and the influence to dissimilar neighbors is weakened.

In the transformed graph, vertices are influenced by their hub neighbors. It is possible that some neighbors of a vertex are non-hub and these neighbors will also have an effect. Therefore, we propose a hub attention mechanism that passes information between non-hub vertices that share common hubs. This attention mechanism is faster and has fewer parameters to be learned than the previous graph attention mechanism [26].

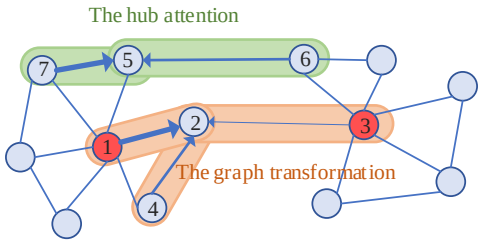


Fig. 1. An illustrative example, where v_1 and v_3 are hub vertices, and vertex v_2 is highly similar to hub vertex v_1, but not to hub vertex v_3. The graph transformation is proposed to handle hub neighbors, and the hub attention is proposed to handle non-hub neighbors.

Fig. 1 presents an example to illustrate how the graph transformation and the hub attention work. Two hub vertices, v_1 and v_3, are marked by red. In the network structure, v_2 are similar with v_1 but dissimilar with v_3. When the information of v_1, v_3 and v_4 runs into v_2, the graph transformation gives a higher weight to the edge e_{12}, a lower weight to the edge e_{23}, and the original weight to the edge e_{24}. v_5 shares the same hub with v_7 but no same hub with v_6. When the information of v_7 and v_6 runs into v_5, the hub attention mechanism gives a higher weight to the edge v_{57} and a lower weight to the edge v_{56}. Note that self-connections are omitted for clarification of how to handle neighbors.

We name our overall framework as CenGCN. In this paper, we present two variants of CenGCN, CenGCN_D and CenGCN_E, based on degree centrality and eigenvector centrality, respectively. To evaluate their performances, we conducted four experiments, vertex classification, link prediction, vertex clustering, and network visualization, on five datasets. These experiments show that the two variants outperform state-of-the-art baselines, even by 70.1% on vertex classification. Though the scale-free property is based on vertex degree, the finding that CenGCN_E, based on eigenvector centrality, achieves excellent performance inspires us to utilise more centrality measurements for GCNs and additional graph-based methods. Further, experiments also show that CenGCN_D and CenGCN_E exhibit a greater performance over GCNs as the network becomes deeper. Thus, an observation derived from this study is that we should explore vertex imbalance and unequal information by vertex centrality on deeper GCNs.

The contributions of this paper can be summarized as follows:

- We propose a framework named CenGCN for scale-free networks. This framework effectively addresses the unequal importance of information from different vertices.
- We propose using label propagation to quantify the similarity between hub vertices and their neighbors, a graph transformation method that captures the influence of hub vertices by vertex centrality, and a hub attention mechanism that assigns new weights to non-hub neighbors by the same hubs.
- We present two variants of CenGCN, namely CenGCN_D and CenGCN_E. Extensive experiments show that these two variants outperform state-of-the-art baselines as well as deeper GCNs.

The remainder of this paper is structured as follows. In Section 2, we give an overview of related work. In Section 3, we introduce preliminaries related to this study and then introduce the CenGCN framework in Section 4. We present extensive experiments to verify the efficacy of this framework in Section 5. Finally, we conclude this paper in Section 6.

2 Related Work

In this section, we review related work about GCNs and scale-free networks.

2.1 Graph Convolutional Networks

Shuman et al. introduce a convolution operation on graph-structured data using the Fourier basis for signal processing

in the paper [27]. Based on this study [27], Bruna et al. [28] define a convolution for graphs from the spectral domain using the graph Laplacian. Theoretical analysis shows that this definition of the convolution operation on graphs can mimic certain geometric properties of Convolutional Neural Networks (CNNs) [29]. A significant limitation of this convolution is that the decomposition of the Laplacian matrix is not scalable to large-scale graphs. To solve the efficiency problem, Defferrard et al. [10] propose a K-localized spectral filter represented by K-order polynomials in the Laplacian. Through the Chebyshev expansion, we can recursively and fast compute the localized filter. With K set to 1, Kipf et al. [8] consider only the first-order neighbors and define a layer-wise propagation rule. The form of this propagation rule is a first-order approximation of localized spectral filters defined in the paper [10]. The first-order GCNs yield a fascinating performance, and many methods have been proposed to improve it.

Graph Attention Networks (GATs) [26] observe that the contributions from neighbors to the central vertex are unequal and adopt attention mechanisms to learn the relative weights between two connected vertices. Furthermore, GATs employ multi-head attention to stabilize the learning process of self-attention. Hierarchical Graph Convolutional Networks (H-GCN) [30] address the failure of GCNs to obtain adequate global information. They repeatedly aggregate structurally similar nodes to hypernodes and then refine the coarsened graph to the original to restore the representation for each node in order to increase the receptive field of each vertex. GCNs are designed for semi-supervised learning, and to extend to unsupervised learning DGI [11] presents a general approach for learning vertex representations within graph-structured data by maximizing mutual information between patch representations and corresponding high-level summaries of graphs.

2.2 Scale-Free Networks

The scale-free property describes how vertex degrees follow a power-law distribution in some networks, such as the Internet [18]. The study presented in [31] reviews some of the empirical evidence for the existence of power-law forms and the theories proposed to explain them. Clauset et al. [32] present a principled statistical framework for discerning and quantifying power-law behavior in empirical data. To define precisely the scale-free graphs, the study [33] provides one possible measure of the extent to which a graph is scale-free. Considering the scale-free property of real-world networks, Jo et al. [34] propose a single-machine based graph engine equipped with the hierarchical indicator and the block-based workload allocation.

Despite many studies about the scale-free property, existing GCNs-based methods have not yet considered it. Recently, Feng et al. [21] proposed a principle for scale-free property preserving network embedding algorithms. Feng et al's study has three significant differences to our study: (i) we believe that those neighbors of a high-degree vertex contain both similar vertices and dissimilar vertices with it; Feng et al's study only assumes a high-degree vertex is dissimilar to its neighbors; (ii) we either reward or punish hub vertices with high centrality indices; Feng et al's study punishes vertices with high degrees; (iii) we learn vector representations using GCNs, or non-linear deep models; Feng et al's study uses spectral cluster and random walk, both of which are linear.

3 Preliminaries

In this section, we introduce the preliminaries related to this study, including the definitions of graphs and GCNs.

3.1 Graph

We consider a graph $G = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ is the vertex set containing n vertices, and $E = \{e_{ij}\}_{1 \le i,j \le n}$ is the edge set. If an undirected edge e_{ij} exists between v_i and v_j, $e_{ij} \in E$. We define the adjacency matrix of G as $A \in \mathbb{R}^{n \times n}$, where $A_{ij} = 1$ if $e_{ij} \in E$, and $A_{ij} = 0$ otherwise. We use D to denote the degree diagonal matrix with $D_{ii} = \sum_j A_{ij}$. For the considered graph, we define a feature matrix $X \in \mathbb{R}^{n \times m}$, where the i_{th} row X_i is v_i's features, and m is the number of the features. Each vertex v_i has a neighbor set N_i. If $A_{ij} = 1$, $v_j \in N_i$.

A graph has scale-free property if its vertex degrees follow a power-law distribution. In this type of graph, only a few vertices are of high degree and called hub vertices or hubs. The majority of vertices connected to a high-degree vertex are of low degree and are not likely to be connected to each other. Formally, the probability density function of the vertex degree D_{ii} has the following form:

$$P_{D_{ii}}(d) = Cd^{-\alpha}, \alpha > 1, d > d_{min} > 0, \tag{1}$$

where α is the exponent parameter, and C is the normalization term. The power-law form only applies to vertices with degrees greater than a certain minimal value d_{min} [32].

In graph theory, centrality has been extensively studied. A vertex with a higher centrality index usually is more influential and has greater prestige. To measure vertices' centrality indices, a number of methods have been put forward [35]. A well-known measure is degree centrality [36], which regards D_{ii} as the index of v_i's centrality. Another popular measure is eigenvector centrality [37]. It is defined as the principal eigenvector of the adjacency matrix defining the network. Other centrality measurements include closeness centrality, betweenness, information centrality, flow betweenness and others [38].

3.2 Graph Convolutional Networks (GCNs)

The convolution operation on graph G is defined in the Fourier domain:

$$y = g_\theta(L)x = g_\theta(U\Lambda U^T)x = Ug_\theta(\Lambda)U^T x, \tag{2}$$

where $L = I_n - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is the normalized Laplacian. I_n is the identity matrix. Λ and U are the diagnonal matrix of eigenvalues and the matrix of eigenvectors of L, respectively. x is the input signal, and y is the filtered signal. $g_\theta(\Lambda)$ is the parameterized filter defined by [10] as a K^{th} order polynomial:

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k, \tag{3}$$

where the parameter θ_k is the polynomial coefficient. To circumvent the multiplication with Fourier basis U that has $O(n^2)$ operations, Defferrard et al. [10] adopt the Chebyshev polynomial $T_k(x)$ of order k computed by the recurrence relation $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0 = 1$ and $T_1 = x$. Thus, the filter is parameterized as:

$$g_\theta(\Lambda) = \sum_{k=1}^{K-1} \theta_k T_k(\hat{\Lambda}), \tag{4}$$

where $\hat{\Lambda} = 2\Lambda/\lambda_{max} - I_n$ is a diagonal matrix of scaled eigenvalues, and λ_{max} is the maxmium eigenvalue of L. The filtering operation can then be written as $y = g_\theta(L)x = \sum_{k=0}^{K-1} \theta_k T_k(\hat{L})x$, where $\hat{L} = 2L/\lambda_{max} - I_n$ is the scaled Laplacian.

Further, let $\lambda_{max} = 2$ and $K = 1$, we can reach the GCNs [8] defined by a layer-wise convolutional operation with the following layer-wise propagation rule:

$$H^{k+1} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^k W^k), \tag{5}$$

Here, $\tilde{A} = A + I$ is the adjacency matrix of the undirected graph G with added self-connections. I is the identity matrix. $\tilde{D}$ is the degree diagonal matrix affiliated to $\tilde{A}$. W^k is the learnable weight in k_{th} layer. $\sigma(\cdot)$ is an activation function, such as $ReLU(\cdot)$. H^k is the input in the k_{th} layer. We set $H^0 = X$. From Eq. (5), we can see that GCNs can be understood as special cases of a simple differentiable information-passing framework [17], i.e., aggregating information from neighbors and itself. One alternative propagation rule often used [39] is defined as

$$H^{k+1} = \sigma(\tilde{D}^{-1}\tilde{A}H^k W^k). \tag{6}$$

The above rule can be obtained if $L = I_n - D^{-1}A$.

4 The Method

In this section, we first introduce the motivation of this study and then elaborate on the technical details of our proposed framework named CenGCN.

4.1 Motivation

From Eq. (5) and Eq. (6), it can be seen that GCNs leverage the immediate adjacency matrix to averagely aggregate information from neighbors and selves, with the belief that all information from different sources is equally important. GCNs only consider inter-node connections when aggregating neighbor information, not vertex types and vertex information. In the real world, some networks have a scale-free property, such as social networks. In these networks, a vertex is more likely to be attracted by hub vertices with high centrality values than by ordinary vertices with low centrality values. Thus the information from hub vertices is more dominant. Existing GCNs, however, have not yet exploited such an important property. In this paper, we study how to define a generalized and transformed adjacency matrix that captures the influence of hub vertices on their neighbors, and how to use the transformed adjacency matrix to improve performance of GCNs.

Recently, Yan et al. [40], [41] have proposed a transformed adjacency matrix defined as:

$$\tilde{A} = D(T - I) + BAB, \tag{7}$$

where B, a biased diagonal matrix with each entry greater than zero, changes weights of all edges. T, a diagonal matrix where each entry $T_{ii} \geqslant 1$, adds a self-connection to each vertex. Because of various B and T, the transformed adjacency matrix can support a wide variety of centrality indices and communities and is beneficial to capture underlying network characteristics. In Eq. (7), the weights of self-connections are limited to multiples of their degrees. To generalize significantly the transformed adjacency matrix, we redefine it as:

$$\tilde{A} = T + BAB, \tag{8}$$

where both T_{ii} and B_{ii} are greater than or equal to one. If we set $T = I$ and $B = I$, $\tilde{A}$ in Eq. (7) is equal to counterparts in Eq. (5) and (6). Because of the generality and flexibility of T and B, we can define manifold $\tilde{A}$, whereby GCNs employ multiple network characteristics. In this paper, we study how to incorporate vertex centrality into T and B, which is particularly important for scale-free networks.

A trivial solution is setting T_{ii} and B_{ii} to the centrality index of v_i. However, one non-negligible issue is that hub vertices with high centrality are likely to attract dissimilar vertices, due to their high attractiveness. For instance, we may follow some persons merely because of their reputation on social media. Such dissimilarity will be strengthened if we use centrality indices to weight edges. To weaken the influence of hub vertices on dissimilar neighbors while transforming the adjacency matrix, we consider the underlying network structure. Numerous studies have indicated that the network structure implies the similarity between vertices [22], [23], [42]. A vertex pays more attention to hubs that show higher similarity with it in network structure. Therefore, instead of relying on the T and B matrices to transform the adjacency matrix, we design three functions to transform the adjacency matrix by combining the vertex centrality indices and the similarity between vertices, as compared to Eq. (8). Finally, we define the transformed adjacency matrix as a combination of three functions:

$$\tilde{A}_{ij} = \begin{cases} f_C(v_i) & \text{if } i = j, \\ f_B(A_{ij}, f_C(v_i), f_C(v_j), f_S(v_i, v_j)) & \text{if } i \neq j, \end{cases} \tag{9}$$

where $f_C : V \to \mathbb{R}$ tells us vertices' centrality indices, and $f_S : V \times V \to \mathbb{R}$ returns the similarity of two vertices in network structure. $f_B : \mathbb{R}^4 \to \mathbb{R}$ calculates a new weight for each pair of connected vertices. This definition of transforming adjacency matrix has two advantages:

- Incorporating vertex centrality indices, vertices pay more attention to similar neighbors with higher centrality indices.
- Considering the underlying network structure, the influence of hub vertices on their dissimilar neighbors is reduced.

It is insufficient for GCNs to directly use the transformed adjacency matrix. This is because the transformation process ignores the influence of non-hub neighbors that are beneficial for the central vertex. We propose a hub attention

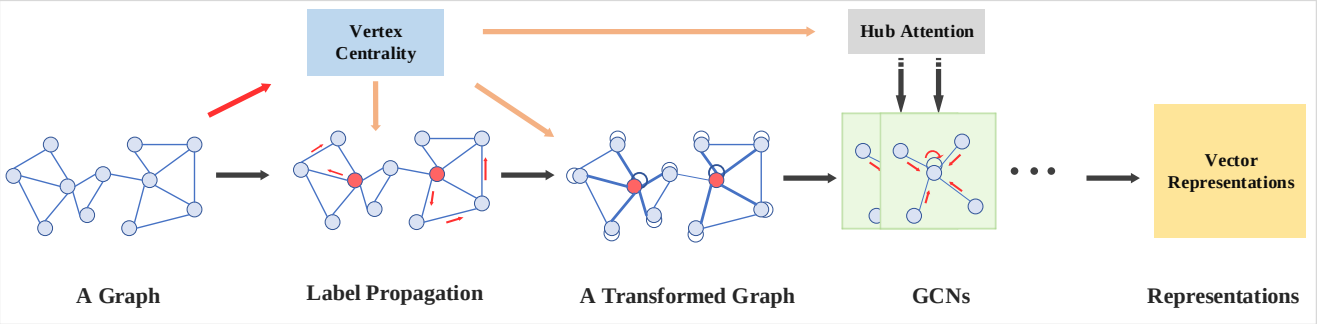


Fig. 2. The framework of CenGCN consists of four steps: computing vertex centrality indices; label propagation using hub vertices, highlighted by red in the above feature; obtaining a transformed graph with self-connections; using multi-layer GCNs with hub attention to generate vector representations. Note that the dots represent repeated layers of GCNs.

mechanism, by which more information passes through edges whose two endpoints share many common hubs.

Next, we elaborate on how to define the three functions f_C, f_B, and f_S and how to inject the hub attention into the layers of GCNs. Firstly, we describe the overall framework of CenGCN.

4.2 Overall Framework

The overall framework of CenGCN is shown in Fig. 2. Given a graph with scale-free property, we first compute the vertex centrality indices. Any vertex centrality measurement can be used here. Based on the computed vertex centrality, we can identify hub vertices and highlight them in red. After labeling these hub vertices, we propose a label propagation method to quantify the similarity between hubs and their neighbors. The proposed method is based on random walk that can reveal the similarity between vertices in network structure. Using the quantified similarities and centrality indices, we transform the given graph to a new graph by increasing or decreasing the weight of each edge. Then self-connections are added to the new graph to force nodes to concentrate on their own characteristics. In the shown figure, the weight of each edge in the transformed graph is drawn proportionally as the thickness of the corresponding line. Then, the transformed graph is used to build multi-layer GCNs. Meanwhile, the hub attention mechanism is injected into each layer to enhance CenGCN. The outputs of GCNs are vector representations of all vertices. These representations can be used for subsequent tasks, such as vertex classification.

Next, we detail each step of this framework.

4.3 Vertex Centrality

There exists various centrality measurements. In this study, we use degree centrality [36] and eigenvector centrality [37], but other centrality measurements are also applicable.

For degree centrality, we define $c_i = D_{ii}$. For eigenvector centrality, we first obtain λ_{max}, the maximum absolute eigenvalue of the adjacency matrix. After that, we compute the eigenvector $\vec{v}$ that corresponds to eigenvalue λ_{max}, according to Eq. (10):

$$\lambda_{max}\vec{v} = A\vec{v}, \tag{10}$$

where $\vec{v}_i$ is the centrality index of v_i. To ensure every value in $\vec{v}$ is greater than or equal to one, we define $\vec{v}'$=$abs(\vec{v})/min(abs(\vec{v}))$, where $abs(\vec{v})$ denotes changing each element in $\vec{v}$ to its absolute value, and $min(abs(\vec{v}))$ represents the minimum absolute value in $\vec{v}$. Finally, we define $c_i = \vec{v}'_i$.

After obtaining vertex centrality indices denoted by c, we define those vertices with very high centrality as hub vertices. To be specific, vertices whose centrality indices are in the top $r\%$ $(0 < r < 100)$ are hubs. We denote the set of hub vertices by N_h and use r to denote the proportion of hub vertices.

The function $f_C(v_i)$ is defined as:

$$f_C(v_i) = \begin{cases} c_i & \text{if } v_i \in N_h, \\ 1 & \text{else.} \end{cases} \tag{11}$$

We only consider the influence of hub vertices. Therefore, f_C is defined to maintain only the centrality indices of the hub vertices.

4.4 Label Propagation

To capture the similarity between hub vertices and their neighbors in the underlying network structure, we propose a label propagation method based on random walk. It is well established that random walk shows the similarity between vertices in the network structure [25]. As a result, it is widely used for community detection [43], [44] and recommendations [45], [46]. If two vertices are similar, there is a high probability to move from one vertex to the other vertex within a small number of hops. This proposed label propagation outputs the probabilities from hub vertices to their neighbors. These probabilities reflect their similarities in the network structure.

We first give each hub vertex a unique label and store labels in the matrix $L \in \mathbb{R}^{|N|\times|N_h|}$, where all elements are zero, but $L_{i,N_h\text{-}index(i)} = 1$ if $v_i \in N_h$. N_h-$index(i)$ denotes the index of hub vertex i in the set of hub points $|N_h|$. We define $P = D^{-1}A$ as a probability transfer matrix, with P_{ij} representing the probability of hopping immediately from v_i to v_j.

After one propagation through Eq. (12), vertices obtain labels from hubs connected with them. Repeated propagations transmit labels of hubs to more vertices, the proportion

of labels decreasing as the distance to hubs increases. After t (t is set to 5 in this paper) propagations (Eq. (13)), we denote the final label matrix by L^t. The i_{th} row L_i^t represent the probabilities of moving from v_i to all hub vertices within 5 hops. The label score L_{ij}^t is the specific probability of moving from v_i to v_j, revealing how similar vertex v_i and hub vertex v_j.

$$L^1 = PL, \tag{12}$$

$$L^t = PL^{t-1}, \qquad t > 1. \tag{13}$$

Given a vertex v_i, a hub vertex v_j, and score L_{ij}^t, we cannot immediately decide whether v_i and v_j are accidentally connected or dissimilar, since L_{ij}^t is greater than zero. If e_{ij} connecting vertex v_i and hub vertex v_j is an accidental edge that is the dissimilar situation between vertex, the corresponding value L_{ij}^t should be very small, but we need to decide the extent of the smallness. Here, we assume that a vertex should have a stronger relationship with hub vertices connected with it than with hub vertices not connected with it when no accidental links appear. Given a vertex v_i and its hub neighbors N_i^h, we reward these connected hubs whose label scores are among the top $|N_i^h|$ scores of L_i^t and punish those linking hubs whose label scores are outside the top $|N_i^h|$ scores. Formally, we sort L_i^t by decreasing order and define $Rank_i(j)$ as a function returning the rank of L_{ij}^t in the sorted order. The $fs(v_i, v_j)$ is defined as:

$$f_S(v_i, v_j) = min(f'_S(v_i, v_j), f'_S(v_j, v_i)), \tag{14}$$

where

$$f'_S(v_i, v_j) = \begin{cases} 1 & \text{if } v_j \notin N_h, \\ 1 & \text{if } v_j \in N_h \text{ and } Rank_i(j) \leqslant |N_h^i|, \\ -1 & \text{if } v_j \in N_h \text{ and } Rank_i(j) > |N_h^i|. \end{cases} \tag{15}$$

$min(\cdot, \cdot)$ returns the minimum value, rendering $f_S(v_i, v_j)$ a symmetric function, i.e., $f_S(v_i, v_j) = f_S(v_j, v_i)$. If $f_S(v_i, v_j) = -1$, v_i and v_i are dissimilar, even though they link to each other. It is noted that if and only if $v_i \in N_h$ or $v_j \in N_h$, $f_S(v_i, v_j)$ has a chance of equaling to -1. The reasons are: (i) hub vertices are more likely to link to dissimilar neighbors than ordinary vertices; (ii) A low centrality index is unable to add large weights to neighbors.

4.5 Graph Transformation

Next, we define $f_B(A_{ij}, f_C(v_i), f_C(v_j), f_S(v_i, v_j))$ as:

$$f_B(A_{ij}, f_C(v_i), f_C(v_j), f_S(v_i, v_j)) = \begin{cases} A_{ij} * f_C(v_i)^p * f_C(v_j)^p & \text{if } f_S(v_i, v_j) = 1, \\ A_{ij} * f_C(v_i)^q * f_C(v_j)^q & \text{if } f_S(v_i, v_j) = -1, \end{cases} \tag{16}$$

where we use two hyper-parameters p and q ($p > 0, q < 0$) to control the influence extent of vertex centrality indices. If $f_S(v_i, v_j) = 1$, centrality indices are used to weight A_{ij}. Otherwise, we reduce the weight of A_{ij} using centrality indices.

After defining f_C, f_S and f_B, we can obtain the transformed adjacency matrix $\tilde{A}$. The transformed matrix not only incorporates vertex centrality indices, but also considers the underlying structure. We summarize this process of graph transformation in Algorithm 1.

Algorithm 1 Graph Transformation

Input: A graph $G = (V, E)$, the adjacency matrix A, hub rate r, propagation number T, a centrality measurement, and hyper-parameters p and q.
Output: A transformed adjacency matrix $\tilde{A}$.

```
Compute centrality index c_i for v_i ∈ V
Obtain hub vertices N^h whose centrality indices are in top r%
Define f_C according to Eq. (11)
Define label matrix L and probability matrix P
L^1 = PL
for t = 2 to T do
    L^t = PL^{t-1}
end for
Define f_S according to Eq. (14)
Define f_B according to Eq. (16)
Obtain Ã according to Eq. (9)
```

4.6 Hub Attention

We define the convolution operation of GCNs at k_{th} layer as

$$H^k = \sigma(\tilde{D}^{-1}\tilde{A}H^{k-1}W^{k-1}), \tag{17}$$

where σ an activation functions, set to $tanh$ in this study. For vertex v_i, this operation also can be written as:

$$H_i^k = \sigma(\frac{1}{\tilde{D}_{ii}} \sum_{v_j \in N_i \cup \{v_i\}} \tilde{A}_{ij} H_j^{k-1} W^{k-1}). \tag{18}$$

From the above equation, we can see that if v_i is connected to a hub vertex with an extremely high centrality index, the information flowing into v_i is almost totally from this hub. The information from non-hub neighbors plays an important role in the decision of the central vertex, such as deciding which class it belongs to. We propose a hub attention mechanism which assigns new weights to non-hub neighbors by consideration of common information from hub vertices. After the convolution of the transformed graph, non-hubs with many shared hub vertices will have similar features between them, and the attention mechanism will assign large weights between vertices with similar features. Therefore, a large weight is assigned to two connected non-hub vertices that share significant hub information. We define $\tilde{N}_i^h = N_i - N_i^h$ as the set of non-hub neighbors of v_i. At k_{th} layer, the weight between v_i and v_j is defined as:

$$a_{ij} = \frac{exp(H_i^k \cdot H_j^k)}{\sum_{v_l \in \tilde{N}_i^h \cup \{v_i\}} exp(H_i^k \cdot H_l^k)}, \tag{19}$$

where $\cdot$ represents the dot product of two vectors. Based on the hub attention, a new convolution is defined as:

$$\tilde{H}_i^k = \sigma(\sum_{v_j \in \tilde{N}_i^h \cup \{v_i\}} a_{ij} H_j^k). \tag{20}$$

The resulting $\tilde{H}_i^k$ is concatenated with H_i^k to enhance GCNs. Finally, H_i^k is computed anew as $H_i^k = H_i^k || \tilde{H}_i^k$, where $||$ represents the concatenation of two vectors. The hub attention mechanism is significantly different with GATs [26]. The differences include:

1) GATs learn individual representations. The representations learned by the hub attention is used as vital complements to GCNs.
2) GATs define many matrices to learn attention weights and require multi-head attention to maintain stability. The hub attention does not need these and thus it is faster to compute attention weights.
3) GATs learn attention weights using features of the previous layer. The hub attention learns them using features of the current layer.

4.7 Optimization

Suppose that CenGCN uses K-layer GCNs, the final vector representation is $Z = H^K$. The learned Z can be used for several network-based tasks. To train CenGCN, we consider both semi-supervised learning and unsupervised learning in case there is no supervision information available.

Semi-supervised Learning. Let Y denote ground-truth vertex class and $\mathcal{Y}_L$ denotes the set of node indices that have class information. We use cross-entropy as the loss function:

$$\mathcal{L} = -\sum_{l \in \mathcal{Y}_L} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf}, \tag{21}$$

where $Y_{lf} = 1$ indicates that v_l belongs to class f, while $Y_{lf} = 0$ indicates otherwise.

Unsupervised Learning. For unsupervised learning, the loss function is defined by reconstructing edges in the original graph:

$$\mathcal{L} = ||(sigmoid(ZZ') - A) \otimes \hat{A}||_F^2, \tag{22}$$

where Z' is the transpose of Z. $||\cdot||_F^2$ is the squared Frobenius norm. $\otimes$ represents element-wise matrix multiplication. In the adjacency matrix A, zero elements outnumber non-zero elements, particularly for sparse networks. As a result, the unsupervised learning is prone to reconstruct zero elements of A. In the above equation, we thus define a matrix $\hat{A}$ to attach higher weights to non-zero elements of A. Specifically, $\hat{A}_{ij} = \rho$ ($\rho > 1$) if $A_{ij} = 1$, else $\hat{A}_{ij} = 1$. Here, we set ρ to 100.

The final loss function is defined as:

$$\mathcal{L}_{loss} = \mathcal{L} + \alpha \mathcal{L}_{reg}, \tag{23}$$

where $\mathcal{L}_{reg}$ is the regularization loss of all learned weights, defined as $\sum_{k=0}^{K-1} ||W^k||_F^2$. α is the hyper-parameter set to 5×10^{-4}. To minimize the loss $\mathcal{L}_{loss}$ and update the parameters of CenGCN, we employ $Adam$ [47], [48] and $Dropout$ [49], [50] with $keep\_pro = 0.5$.

We summarize the overall framework of CenGCN in Algorithm 2.

4.8 Computational Complexity

The first step of the CenGCN framework is calculating centrality indices. The time complexity at this step is $\Theta(n)$ when degree centrality is used and $\Theta(n^2)$ when eigenvector centrality is used. The next step is label propagation and can be finished in $\Theta(Tn^2)$ since here we only calculate T dot productions of a vector and a matrix. At last, the time complexity in GCN layers is $\Theta(Ln^2)$, where L here is the number of layers. Overall, the computational complexity of the CenGCN framework is $\Theta((T + L)n^2)$.

Algorithm 2 The Framework of CenGCN

Input: A graph $G = (V, E)$, the adjacency matrix A, the feature matrix X, hub rate r, propagation number T, a centrality measurement, number of layers K, hyper-parameters p, q and α, learning rate θ, class information $\mathcal{Y}_L$, and a convergence condition.
Output: Well trained CenGCN.

```
Obtain the transformed adjacency matrix Ã and N^h using Algorithm 1
Initialize all weight parameters {W^k}_{0≤k≤K−1}
while The convergence condition is not satisfied: do
    H^0=X
    for k = 1 to K do
        Compute H^k according to (17)
        Compute H̃^k according to (20)
        H^k = H^k||H̃^k
    end for
    Z = H^K
    Compute L_reg = Σ_{k=0}^{K−1} ||W^k||_F^2
    if class information is available: then
        Compute L according to (21)
    else
        Compute L according to (22)
    end if
    Compute L_loss according to (23)
    Minimize L_loss by Adam with learning rate θ
end while
```

5 Experiments

In this section, we compare our proposed framework CenGCN with several baselines by running four experiments: vertex classification, link prediction, vertex cluster, and network visualization. The results of parameter sensitivities are presented at the end of the section. We have implemented the CenGCN in Python 3.6 with Tensorflow1.15.

5.1 Datasets

We use five datasets, which are introduced in [51], [21] and [52]. Their statistics are summarized in Table 1.

- Facebook: The data was collected from survey participants using the Facebook app. Vertices represent users, and edges represent friendship.
- Twitter: The data was crawled from public sources. Vertices indicate users, and edges denote following relationships.
- Gplus: The data was collected from Google+. Vertices indicate users, and edges denote following relationships.
- Youtube: The data was collected from a video-sharing website that includes a social network. Users are denoted by vertices, and edges denote friendship.
- LiveJournal: The data was collected from a free online blogging community, where users declare friendship with each other. Vertices represent users, and edges represent friendship.

The above datasets are scale-free networks. Most of the connections are concentrated in a few centers. In the semi-supervised learning task (vertex classification), for all five datasets we use vertex classes as labels.

TABLE 1
Statistics of datasets. '—' means no data available.

datasets	Vertices	Edges	Features	Classes
Facebook	3944	87870	1385	8
Twitter	1533	38323	10353	8
Gplus	5331	351726	1988	5
Youtube	4684	19443	—	8
LiveJournal	3009	44599	—	4

5.2 Comparison Algorithms

Based on two different centrality measurements, we define two variants of CenGCN:

- CenGCN_D: Utilizes degree centrality within the overall proposed framework.
- CenGCN_E: Utilizes eigenvector centrality within the overall proposed framework.

In addition, we define the following variants from CenGCN_D and CenGCN_E as complements to demonstrate the efficacy and necessity of each part of CenGCN.

- CenGCN_TD and CenGCN_TE: Uses only the transformed adjacency matrix, without the hub attention mechanism.
- CenGCN_AD, CenGCN_AE: Uses only the hub attention mechanism, without the transformed adjacency matrix.
- CenGCN_WD, CenGCN_WE: Uses the centrality indices of hub vertices to increase edge weights by setting p to q.
- CenGCN_ID, CenGCN_IE: Uses the centrality indices of hub vertices to decrease edge weights by setting p to q.

To verify the efficiency of CenGCN, we conduct experiments against the following baselines:

- GCN_Cheby [10]: Uses fast localized convolutional filters on graphs using Chebyshev expansion.
- GCNs [8]: Uses a layer-wise convolutional operation that encodes both local graph structure and vertex features.
- GATs [26]: Leverages masked self-attentional layers to specify different weights to different vertices in a neighborhood.
- DGI [11]: Learns vertex representations in an unsupervised manner, by relying on maximizing mutual information between patch representations and corresponding high-level summaries of graphs.
- H-GCN [30]: Repeatedly aggregates structurally similar vertices to hyper-vertices and then refines the coarsened graph to the original to restore the representation for each vertice.
- DPSW [21]: Punishes the proximity between high-degree vertices using scale-free property preserving network embedding algorithm. DPSW represents the best model drawing upon DP-Spectral and DP-Walker.

5.3 Experimental Setup

We consider the four different network tasks:

- Vertex classification: This is a semi-supervised learning task. Classes of vertices are the ground truth. 10% vertices with class information are used as training examples, and 10% vertices with class information are validation examples. The remaining vertices are test examples. The learning rate θ is set to 0.01, and the iteration number is set to 1000. The best parameters on validation examples are saved and then used for test examples. Accuracy is used as the evaluation metric.
- Link prediction: This is an unsupervised learning task. We first randomly hide 50% edges as positive examples and randomly select 50% non-existent edges as negative examples. The remaining graph is used to train. According to the paper [23], the Hadamard operator of two vertices is a good representation for their edge. Thus, we construct edge representations by this operator. Logistic regression is used for binary classification. The learning rate θ is set to 0.01. We stop the training when the loss $\mathcal{L}_{loss}$ remains stable or the iteration number is over 150. AUC (Area Under the Curve) is used as the evaluation metric.
- Vertex clustering: This is an unsupervised learning task. The representation Z serves as the input features of K-means, a clustering method. Classes of vertices are the ground truth. Normalized Mutual Information (NMI) [53] is used as the evaluation metric. The learning rate θ is set to 0.001, a smaller rate 0.00001 on Twitter. We stop the training when the loss $\mathcal{L}_{loss}$ remains stable or the iteration number is over 150.
- Network visualization: The representation Z obtained in vertex clustering is used here for network visualization. We feed Z into the standard t-SNE tool [54] to lay out the network and mask vertices of the same class with the same color. The network is visualized in a 2-dimensional space.

In semi-supervised learning, we employ a two-layer GCN with a 16-unit hidden layer for all variants of CenGCN. In unsupervised learning, we employ a two-layer GCN with a 512-unit hidden layer and a 128-unit output layer for all variants of CenGCN. The settings and sensitivities of parameters p, q, and r, as well as the number of layers, are presented in *Parameter Sensitivity*. The parameters of the baselines are set in accordance with the original papers.

5.4 Vertex classification

The task of vertex classification is discovering classes of those vertices that have no class information. We first verify the efficacies of CenGCN and baselines through this task in this experiment. Table 2 shows the accuracies of CenGCN's variants and baselines on vertex classification. The best performance is boldfaced. From the table, we can see that on the five networks, CenGCN_D always achieves the best performance and CenGCN_E outperforms all baselines. These results demonstrate the significant superiority of CenGCN and the necessity to incorporate vertex centrality indices into GCNs. Besides, the following findings are also striking:

TABLE 2
The accuracy of vertex classification. The best performance is boldfaced.

Algorithm	Facebook	Twitter	Gplus	Youtube	LiveJournal
GCN_Cheby	0.915	0.972	0.787	0.812	0.810
GCNs	0.914	0.954	0.716	0.889	0.901
GATs	0.970	0.967	0.732	0.827	0.892
DGI	0.936	0.954	0.771	0.227	0.592
H-GCN	0.982	0.943	0.914	0.915	0.888
DPSW	0.892	0.789	0.922	0.892	0.872
CenGCN_D	**0.992**	**0.987**	**0.949**	**0.920**	**0.912**
CenGCN_TD	0.970	0.982	0.943	0.914	0.897
CenGCN_AD	0.970	0.969	0.933	0.915	0.910
CenGCN_WD	0.832	0.965	0.941	0.904	0.903
CenGCN_ID	0.888	0.967	0.861	0.893	0.893
CenGCN_E	**0.992**	**0.987**	0.936	0.919	0.903
CenGCN_TE	0.912	0.905	0.717	0.873	0.894
CenGCN_AE	0.916	0.930	0.742	0.866	0.900
CenGCN_WE	0.932	0.973	0.717	0.892	0.895
CenGCN_IE	0.912	0.971	0.870	0.902	0.877

TABLE 3
The AUC score of link prediction. The best performance is boldfaced.

Algorithm	Facebook	Twitter	Gplus	Youtube	LiveJournal
GCN_Cheby	0.672	0.842	0.725	0.676	0.759
GCNs	0.809	0.729	0.711	0.578	0.711
GATs	0.633	0.852	0.558	0.685	0.757
DGI	0.723	0.862	0.678	0.613	0.621
H-GCN	0.708	0.564	0.601	0.656	0.739
DPSW	0.767	0.581	0.797	0.714	0.753
CenGCN_D	**0.892**	**0.873**	**0.801**	**0.731**	0.848
CenGCN_TD	0.854	0.857	0.787	0.718	0.850
CenGCN_AD	0.885	0.855	0.775	0.713	0.837
CenGCN_WD	0.884	0.850	0.796	0.728	0.831
CenGCN_ID	0.882	0.847	0.699	0.713	0.828
CenGCN_E	0.891	0.871	0.769	0.727	**0.853**
CenGCN_TE	0.887	0.858	0.753	0.715	0.850
CenGCN_AE	0.868	0.856	0.746	0.756	0.841
CenGCN_WE	0.808	0.861	0.742	0.698	0.848
CenGCN_IE	0.840	0.856	0.776	0.681	0.842

- On two networks, Facebook and Twitter, CenGCN_D and CenGCN_W have the same performance. But on the other three networks, CenGCN_D outperforms CenGCN_W. Overall, CenGCN_D performs better CenGCN_E, owing to the scale-free property based on vertex degrees. The finding that CenGCN_E outperforms all baselines gives us motivation to explore more centrality measurements.
- This table shows that CenGCN_D and CenGCN_E consistently outperform DPSW on the five networks, though they are proposed for scale-free networks. The difference in performance can likely be attributable to the fact that a hub vertex can link to both similar and dissimilar vertices, while DPSW assumes that a vertex with a higher degree is more dissimilar to its neighbors.
- Compared with standard GCNs, CenGCN_D and CenGCN_E achieve great performance. On Gplus, CenGCN_D achieves an improvement of 23.6%. Compared with state-of-the-art GCN-based variants, CenGCN_D outperforms GATs by 22.0% on Gplus, outperforms DGI by 70.1% on Youtube, and outperforms H-GCN by 4.5% on Twitter. These significant improvements indicate the necessity for GCNs to utilise vertex centrality.
- CenGCN_D outperforms the other four variants of CenGCN that use degree centrality; CenGCN_E outperforms the other four variants of CenGCN that use eigenvector centrality. These results indicate that the transformed graph needs to be combined with the hub attention mechanism, and we need to consider both the increase and decrease of edge weights.

5.5 Link Prediction

Link prediction aims at predicting whether two vertices that are not connected are potentially connected. In this experiment, we concentrate on the link prediction task and compare the performance of CenGCN and baselines. Table 3 shows their AUC scores on link prediction. We report the best performance by boldface. From these AUC scores in the table, we can see that CenGCN performs extremely well. Among the five maximum scores, CenGCN_D achieves four and CenGCN_E achieves one. These results suggest the vertex centrality is a powerful indicator of link prediction. More noticeable findings are summarized as follows:

- As a whole, CenGCN_D performs better than CenGCN_E. On Livejournal, CenGCN_E outperforms CenGCN_D only by 0.3%. We also notice that the AUC gap between them is small, except for Gplus where CenGCN_D outperforms CenGCN_E by up to 5.2%. Although the vertex degree is an intuitive centrality measurement for scale-free networks, the results of eigenvector centrality compare very favourably to these of degree centrality.
- CenGCN_D outperforms all baselines, with the largest improvement of 9.0% on Facebook. CenGCN_E outperforms all baselines in the vast majority of cases, with the largest improvement of 8.9% on Facebook. It is an unanticipated finding that DPSW performs better CenGCN_E on Gplus. But on the other four networks, CenGCN_E performs better than DBSW, particularly on Twitter.
- In most cases, CenGCN_D is the best among all variants with degree centrality, and CenGCN_E is the best among all variants with eigenvector centrality. Contrary to expectations, CenGCN_IE outperforms CenGCN_E on Gplus and CenGCN_TD outperforms CenGCN_D on Livejournal. Overall, CenGCN_D or CenGCN_E performs best only when all designed parts are used.

5.6 Vertex clustering

Vertex clustering is a typical unsupervised learning task and is used to find which vertices form a group. In this experiment, we compare the performance of CenGCN and baselines through this task. We show the NMI in Table 4, where the best performance is reported by boldface. Table 4 shows that on the five networks, CenGCN_D achieves the best performance and CenGCN_W performs better than all

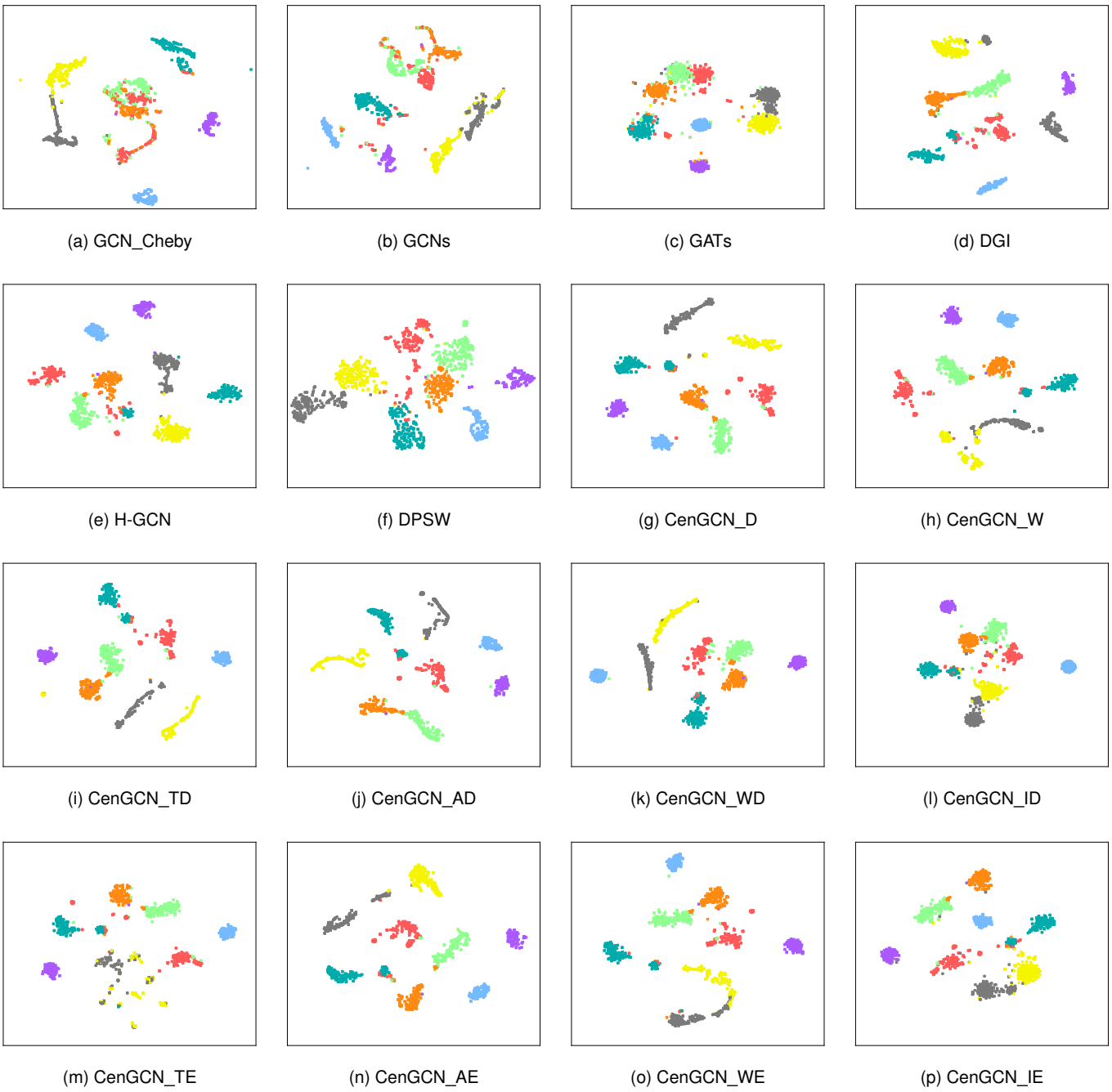


Fig. 3. Network Visualization using t-sne. Each color represents one class.

baselines, further suggesting the importance to utilise vertex centrality for GCNs. Other noticeable observations from this experiment are summarized as follows:

- CenGCN_D always outperforms CenGCN_E on the five networks. On Gplus, the NMI gap between them is 22.2%. These results may be explained by the fact that the scale-free property is defined based on vertex degrees. When no class information is provided, network structures play a major role in training CenGCN.
- From the table, we can see that the two variants of CenGCN, CenGCN_D and CenGCN_E, outperform all baselines. CenGCN_D achieves significant improvement on Facebook and Gplus. On Facebook, CenGCN_D achieves improvement of at least 20.2%; On Gplus, it achieves improvement of at least 27.7%. Besides, CenGCN_E achieves improvement of at least 10.8% on Twitter.
- It still can be seen that CenGCN_D performs best among variants with degree centrality and CenGCN_E performs best among variants with eigenvector centrality. The result further indicates the necessity of each part of CenGCN. It is surprising to find that only using the transformed graph or the hub attention is sufficient to achieve significant performance when we use degree centrality.

TABLE 4
The NMI of vertex clustering. The best performance is boldfaced.

Algorithm	Facebook	Twitter	Gplus	Youtube	LiveJournal
GCN_Cheby	0.540	0.729	0.389	0.379	0.636
GCNs	0.561	0.656	0.288	0.300	0.657
GATs	0.577	0.756	0.440	0.552	0.685
DGI	0.639	0.799	0.341	0.056	0.143
H-GCN	0.683	0.714	0.317	0.424	0.686
DPSW	0.342	0.506	0.211	0.642	0.512
CenGCN_D	**0.885**	**0.911**	**0.717**	**0.700**	**0.783**
CenGCN_TD	0.860	0.881	0.613	0.668	0.767
CenGCN_AD	0.838	0.909	0.620	0.657	0.764
CenGCN_WD	0.784	0.808	0.326	0.598	0.738
CenGCN_ID	0.581	0.751	0.417	0.207	0.685
CenGCN_E	0.752	0.907	0.495	0.670	0.761
CenGCN_TE	0.672	0.757	0.441	0.619	0.774
CenGCN_AE	0.732	0.905	0.468	0.667	0.739
CenGCN_WE	0.709	0.875	0.387	0.654	0.750
CenGCN_IE	0.738	0.881	0.240	0.623	0.721

5.7 Network Visualization

Network Visualization helps us explore the network structure in a low-dimensional space. In this experiment, we visualize the Twitter network using the learned vector representations. Fig. 3 shows the visualized network on a 2-dimensional space, where each color represents one class. We summarize observable findings as follows:

- GCN_Cheby strongly confuses Red, Orange, and Light Green, and cannot develop boundaries to separate them. GCNs and GATs tightly connect Red, Orange, and Light Green, as well as Grey and Yellow. DGI poorly separates two subgroups of Grey. H-GCN poorly separates two subgroups of Sea Green. DPSW is insufficient to separate points of different colors. Also the points of the same color are less close together.
- CenGCN_D shows the significant capacity of visualizing the Twitter network, sufficient to separate points of different colors and tightly cluster the points of the same color. CenGCN_E also shows the significant capacity of visualizing this network, but it slightly confuses Yellow and Grey. Thus, these results suggest the usefulness of vertex centrality for network visualization.
- From the figure, we can see that some complementary variants of CenGCN also show significant performance in this experiment. Examples include CenGCN_AD and CenGCN_AE. But CenGCN_AD slightly confuses Orange and Light Green. It can be seen that CenGCN_TE fails to separate Green and Grey.

For quantitative comparison, we report KL divergences of algorithms in Fig. 4. KL divergences capture the errors between the input pairwise similarities and their projections in the 2-dimensional mapping. A lower KL divergence score indicates a better performance. We can see that CenGCN_D and CenGCN_E achieve the two smallest scores. Thus, they demonstrate better visualization performance than baselines.

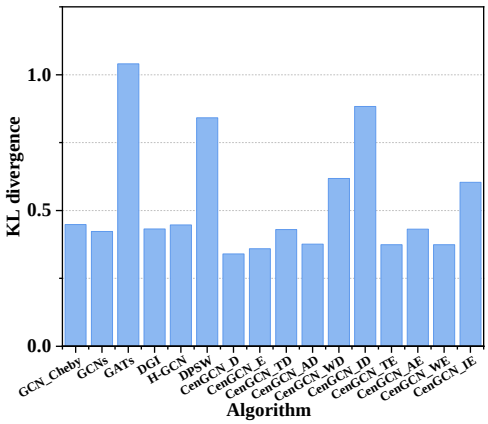
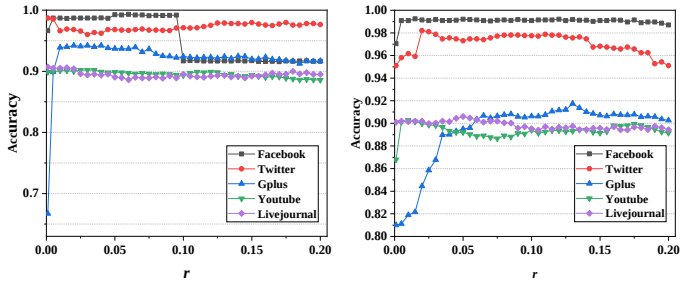


Fig. 4. KL divergence on Twitter

5.8 Parameter Sensitivity

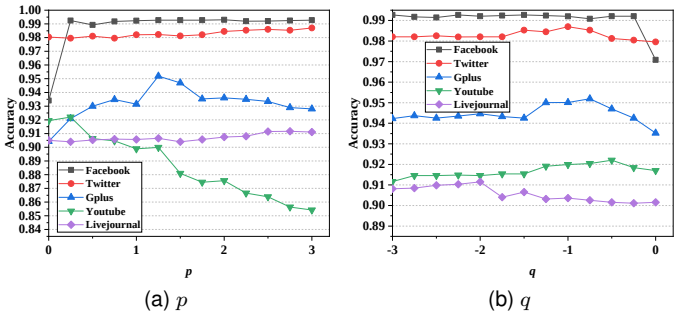
For CenGCN, three crucial hyper-parameters are p, q, and r. In this section, we investigate how they affect the performance of CenGCN on vertex classification. In addition to this, we also investigate the influence of the number of layers. For simplicity, we run experiments on CenGCN_D and CenGCN_E, omitting complementary variants.

5.8.1 *The ratio of hub vertices*



(a) CenGCN_D (b) CenGCN_E

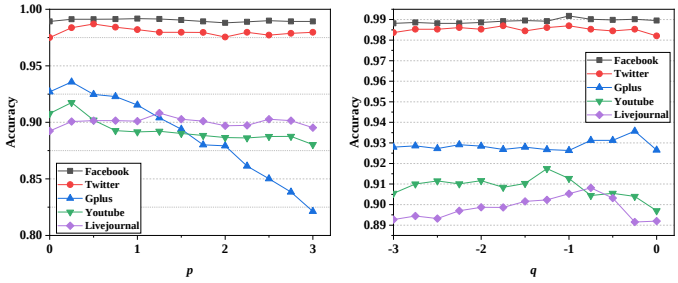
Fig. 5. Sensitivities w.r.t. r



(a) p (b) q

Fig. 6. Sensitivities of CenGCN_D w.r.t. p and q

We select r in the range from 0.001 to 0.2. A bigger r implies more hub vertices to be considered. Here, we set p and q to 1 and -1, respectively. The Fig. 5 shows sensitivities of CenGCN_D and CenGCN_E w.r.t. r. From the figure, we can see different variation tendencies in different networks,



(a) p (b) q

Fig. 7. Sensitivities of CenGCN_E w.r.t. p and q

as r goes from 0.001 to 0.2. We summarize findings in the five datasets as follows:

- On Facebook, accuracies of CenGCN_D and CenGCN_E abruptly become higher when r increases from 0.001 to larger values. CenGCN_D maintains its peak from 0.05 to 0.095. After that, its accuracy drastically decreases to a low value, which is maintained as r is from 0.1 to 0.2. CenGCN_E maintains its peak when r is between 0.005 and 0.165. Finally, its accuracy starts to slowly decrease.
- On Twitter, the accuracy of CenGCN_D immediately reaches its peak when r passes from 0.001. It then starts to decrease and slowly increases when r is over 0.05. The accuracy of CenGCN_E reaches its peak when r is 0.02. After that, it experiences a decrease, then an increase and finally a decrease.
- On Gplus, the accuracy of CenGCN_D significantly increases when r passes from 0.001 to 0.01. When r continues to 0.03, the accuracy gently increases. Finally, it decreases to a stable value. The accuracy of CenGCN_E maintains an increasing tendency until r is up to 0.13. This tendency first is very strong and then becomes slow. After r is over 0.13, the accuracy starts to decrease.
- On Youtube, the accuracy of CenGCN_D has a stable change. It first gently increases until r is around 0.03. Then it experiences a decrease, an increase and finally a decrease. The accuracy of CenGCN_E has a drastic increase when r goes from 0.001 to higher values. It reaches its peak when r is 0.01. After that, it first decreases and then increases. Finally, it decreases again when r is over 0.16.
- On Livejournal, the accuracy of CenGCN_D immediately reaches its peak when r is 0.001. As r passes to 0.2, the accuracy first decreases to a stable value and then increases. The accuracy of CenGCN_E reaches its peak as r goes from 0.001 to 0.05. After that, it starts to decrease to a value of around 0.895 with small fluctuations.

5.8.2 *The influence extent of vertex centrality*

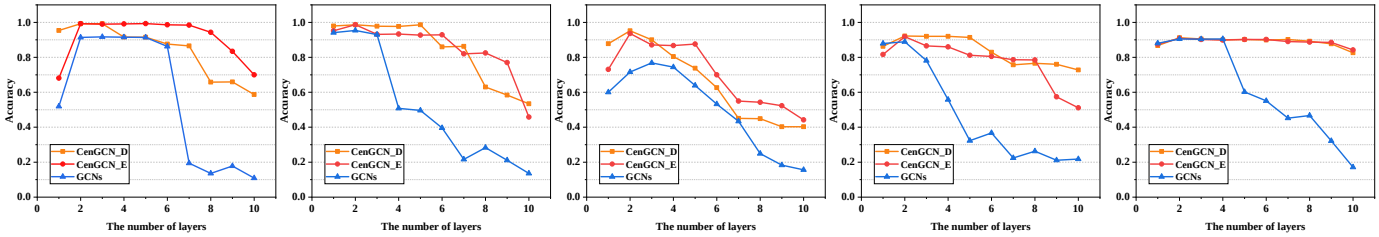
The two parameters p (> 0) and q (< 0) control the influence extent of vertex centrality. In this experiment, we investigate the sensitivities w.r.t. p and q. Fig. 6 and Fig. 7 show the sensitivity results of CenGCN_D and CenGCN_E, respectively. From the two figures, we can observe the following findings:

- On Facebook, the accuracy of CenGCN_D maintains stability when p and q are non-zero values. For CenGCN_E, its accuracy has a peak when p is 1 and q is -1. Outside of the peak, it also maintains a stable value.
- On Twitter, the accuracy of CenGCN_D steadily and slowly increases as p goes from 0 to 3. With q increasing, the accuracy increases from a stable value to its peak, where q is -1. For CenGCN, its accuracy is maximum when p=0.5. As p becomes larger, the accuracy maintains stability. It is also a stable value when q is over -3 and below 0.
- On Gplus, the accuracy of CenGCN_D has a conspicuous peak when p=1.25 and q=-0.75. It is stable as q goes from -3 to -1.5. Similar to CenGCN_D, the accuracy of CenGCN_E also has a conspicuous peak, where p=0.25 and q=-0.25. We can see that the accuracy decreases linearly when p is over 0.25.
- On Youtube, CenGCN_D needs a small p. As p goes from 0.25 to 3, its accuracy steadily decreases. The accuracy slowly increases when q increases to -0.5. CenGCN_e also needs a small p, but it needs a smaller q of -1.25.
- On Livejournal, the accuracy of CenGCN_D maintains a stable increase as p becomes larger, and maintains a stable decrease as q goes from -2 to 0. The accuracy of CenGCN_E has a peak when p=1.25. On either sides of this peak, it maintains stability. As q goes from -3 to -0.75, the accuracy steadily increases. After that, it drastically decreases.

5.8.3 *The number of layers*

In this experiment, we investigate how the number of layers affects the performance of CenGCN_D and CenGCN_E. Besides, the performance of GCNs is added here for a comparison. The p and q are set to their optimal values. Fig. 8 shows their performance in the five networks when the number increases from 1 to 10. We summarize noticeable findings as follows:

- CenGCN_D, CenGCN_E, and GCNs achieve great performance when the number of layers is 2. One exception is that on Gplus, GCNs have the best performance when the number is 3. Therefore, it is reasonable for GCN-based models to design a two-layer neural network.
- When the number is greater than 2 and continues to increase, the accuracies of GCNs show decreasing tendencies. When the number increases to 10, the accuracy of GCNs is below or slightly over 0.2. The result shows that GCNs suffer from shallow models.
- As the number increases, the extent by which CenGCN_D and CenGCN_E outperform GCNs also increases. On LiveJournal, when the number is at 10, CenGCN_D and CenGCN_E achieve accuracies of more than 0.8, significantly outperforming GCNs. On the other four networks, we can also observe a large gap between the two variants of CenGCN and GCNs. These figures demonstrate that CenGCN deepens GCNs.



(a) Facebook (b) Twitter (c) Gplus (d) Youtube (e) Livejournal

Fig. 8. Sentivities w.r.t. the number of layers

Existing GCN-based methods employ a two-layer neural network. Such a network merely mines the relationship between vertices whose distance is at most 2-hop, failing to exploit global network structure. How to deepen GCNs is still an open issue and worthwhile to study further. This experiment shows that CenGCN has excellent performance compared with GCNs when the number of layers increases to 10. Therefore, for deepening GCNs, a suggestion from this experiment is that we can utilise vertex centrality.

6 Conclusion

In this paper, we study how to address the inequality of information from vertices. We propose label propagation with labeled hub vertices to quantify the similarity between hub vertices and their neighbors. Based on this similarity and centrality indices, we transform the graph to capture the influence of hub vertices. When inputting the transformed graph into GCNs, we propose a hub attention mechanism to learn new weights linking to non-hub neighbors from the same hubs. In four experiments, the two variants, CenGCN_D and CenGCN_E, demonstrate their significant improvement over baselines and excellent performance when the number of layers increases to 10.

GCNs are rapidly developing and proving effective tools in network modeling and analysis. Although there are many studies about GCNs, a great number of issues remain to be addressed. Two serious issues are that GCNs suffer from local limits and shallow models. This study demonstrates a way to explore vertex imbalance and unequal information by vertex centrality, a macroscopic network characteristic, to enhance and enrich GCNs. In the future, we will consider more network characteristics, such as subgraphs.

References

[1] C. Stadtfeld, A. Vörös, T. Elmer, Z. Boda, and I. J. Raabe, "Integration in emerging social networks explains academic failure and success," *Proceedings of the National Academy of Sciences*, vol. 116, no. 3, pp. 792–797, 2019.

[2] I. A. Kovács, K. Luck, K. Spirohn, Y. Wang, C. Pollis, S. Schlabach, W. Bian, D.-K. Kim, N. Kishore, T. Hao *et al.*, "Network-based prediction of protein interactions," *Nature communications*, vol. 10, no. 1, p. 1240, 2019.

[3] F. Xia, W. Wang, T. M. Bekele, and H. Liu, "Big scholarly data: A survey," *IEEE Transactions on Big Data*, vol. 3, no. 1, pp. 18–35, 2017.

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

[5] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 9, pp. 1616–1637, 2018.

[6] J. Liu, X. Kong, F. Xia, X. Bai, L. Wang, Q. Qing, and I. Lee, "Artificial intelligence in the 21st century," *IEEE Access*, vol. 6, pp. 34 403–34 421, 2018.

[7] F. Xia, S. Yu, C. Liu, J. Li, and I. Lee, "Chief: Clustering with higher-order motifs in big networks," *IEEE Transactions on Network Science and Engineering*, 2021.

[8] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR 2017 : International Conference on Learning Representations 2017*, 2017.

[9] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018, pp. 3538–3545.

[10] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, 2016, pp. 3844–3852.

[11] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *ICLR 2019 : 7th International Conference on Learning Representations*, 2019.

[12] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, S. Pan, and H. Liu, "Graph learning: A survey," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 2, pp. 109–127, 2021.

[13] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2018, pp. 974–983.

[14] X. Wang, R. Wang, C. Shi, G. Song, and Q. Li, "Multi-component graph convolutional collaborative filtering," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI*. AAAI Press, 2020, pp. 6267–6274.

[15] D. Nathani, J. Chauhan, C. Sharma, and M. Kaul, "Learning attention-based embeddings for relation prediction in knowledge graphs," in *ACL 2019 : The 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 4710–4723.

[16] X. Kong, J. Zhang, D. Zhang, Y. Bu, Y. Ding, and F. Xia, "Gene of scientific success," *ACM Transactions on Knowledge Discovery from Data*, vol. 5, no. 2, p. 41, 2020.

[17] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 1263–1272.

[18] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in *ACM SIGCOMM computer communication review*, vol. 29, no. 4. ACM, 1999, pp. 251–262.

[19] L. M. Smith, K. Lerman, C. Garcia-Cardona, A. G. Percus, and R. Ghosh, "Spectral clustering with epidemic diffusion," *Physical Review E*, vol. 88, no. 4, p. 042813, 2013.

[20] Y. Gu, Y. Sun, Y. Li, and Y. Yang, "Rare: Social rank regulated large-scale network embedding," in *Proceedings of the 2018 World Wide Web Conference*. International World Wide Web Conferences Steering Committee, 2018, pp. 359–368.

[21] R. Feng, Y. Yang, W. Hu, F. Wu, and Y. Zhang, "Representation learning for scale-free networks," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018, pp. 282–289.

[22] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD*

international conference on Knowledge discovery and data mining. ACM, 2014, pp. 701–710.
[23] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 855–864.
[24] Y. Yan, Y. Bian, D. Luo, D. Lee, and X. Zhang, "Constrained local graph clustering by colored random walk," in *The World Wide Web Conference*. ACM, 2019, pp. 2137–2146.
[25] F. Xia, J. Liu, H. Nie, Y. Fu, L. Wan, and X. Kong, "Random walks: A review of algorithms and applications," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 4, no. 2, pp. 95–107, 2019.
[26] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR 2018 : International Conference on Learning Representations 2018*, 2018.
[27] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal processing magazine*, vol. 30, no. 3, pp. 83–98, 2013.
[28] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *ICLR 2014 : International Conference on Learning Representations (ICLR) 2014*, 2014.
[29] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.
[30] F. Hu, Y. Zhu, S. Wu, L. Wang, and T. Tan, "Hierarchical graph convolutional networks for semi-supervised node classification." in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019, pp. 4532–4539.
[31] M. E. Newman, "Power laws, pareto distributions and zipf's law," *Contemporary physics*, vol. 46, no. 5, pp. 323–351, 2005.
[32] A. Clauset, C. R. Shalizi, and M. E. Newman, "Power-law distributions in empirical data," *SIAM review*, vol. 51, no. 4, pp. 661–703, 2009.
[33] L. Li, D. Alderson, J. C. Doyle, and W. Willinger, "Towards a theory of scale-free graphs: Definition, properties, and implications," *Internet Mathematics*, vol. 2, no. 4, pp. 431–523, 2005.
[34] Y.-Y. Jo, M.-H. Jang, S.-W. Kim, and S. Park, "Realgraph: a graph engine leveraging the power-law distribution of real-world graphs," in *The World Wide Web Conference*. ACM, 2019, pp. 807–817.
[35] D. Koschützki, K. A. Lehmann, L. Peeters, S. Richter, D. Tenfelde-Podehl, and O. Zlotowski, "Centrality indices," in *Network analysis*. Springer, 2005, pp. 16–61.
[36] L. C. Freeman, "Centrality in social networks conceptual clarification," *Social networks*, vol. 1, no. 3, pp. 215–239, 1978.
[37] P. Bonacich, "Factoring and weighting approaches to status scores and clique identification," *Journal of mathematical sociology*, vol. 2, no. 1, pp. 113–120, 1972.
[38] S. P. Borgatti, "Centrality and network flow," *Social networks*, vol. 27, no. 1, pp. 55–71, 2005.
[39] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*. Springer, 2018, pp. 593–607.
[40] X. Yan, B. M. Sadler, R. J. Drost, L. Y. Paul, and K. Lerman, "Graph filters and the z-laplacian," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 774–784, 2017.
[41] R. Ghosh, S.-h. Teng, K. Lerman, and X. Yan, "The interplay between dynamics and networks: centrality, communities, and cheeger inequality," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 1406–1415.
[42] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 1225–1234.
[43] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.
[44] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences*, vol. 105, no. 4, pp. 1118–1123, 2008.
[45] A. N. Nikolakopoulos and G. Karypis, "Recwalk: Nearly uncoupled random walks for top-n recommendation," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. ACM, 2019, pp. 150–158.
[46] J. Ren, F. Xia, X. Chen, J. Liu, M. Hou, A. Shehzad, N. Sultanova, and X. Kong, "Matching algorithms: Fundamentals, applications and challenges," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 5, no. 3, pp. 332–350, 2021.
[47] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *ICLR 2015 : International Conference on Learning Representations 2015*, 2015.
[48] L. Wang, J. Ren, B. Xu, J. Li, W. Luo, and F. Xia, "MODEL: motif-based deep feature learning for link prediction," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 2, pp. 503–516, 2020.
[49] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
[50] J. Liu, F. Xia, L. Wang, B. Xu, X. Kong, H. Tong, and I. King, "Shifu2: A network representation learning based model for advisor-advisee relationship mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, pp. 1763–1777, 2021.
[51] J. Leskovec and J. J. Mcauley, "Learning to discover social circles in ego networks," in *Advances in neural information processing systems*, 2012, pp. 539–547.
[52] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowledge and Information Systems*, vol. 42, no. 1, pp. 181–213, 2015.
[53] P. A. Estévez, M. Tesmer, C. A. Perez, and J. M. Zurada, "Normalized mutual information feature selection," *IEEE Transactions on Neural Networks*, vol. 20, no. 2, pp. 189–201, 2009.
[54] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. 11, pp. 2579–2605, 2008.

APPENDIX
SUPPLEMENTARY EXPERIMENTAL RESULTS

This supplementary material provides more detailed experimental results, including standard deviations calculated for multiple experiments.

On the vertex classification and link prediction tasks, we perform several experiments to calculate the standard deviation of the method as a way of verifying the robustness of the model. Where DPSW is not affected by random seeds and therefore has a standard deviation of 0. The specific results are shown in Table 5, 6.

Feng Xia (M'07-SM'12) received the BSc and PhD degrees from Zhejiang University, Hangzhou, China. He was Full Professor and Associate Dean (Research) in School of Software, Dalian University of Technology, China. He is Associate Professor and former Discipline Leader (IT) in School of Engineering, IT and Physical Sciences, Federation University Australia. Dr. Xia has published 2 books and over 300 scientific papers in international journals and conferences. His research interests include data science, artificial intelligence, graph learning, and systems engineering. He is a Senior Member of IEEE and ACM.

TABLE 5
The accuracy of vertex classification.

Algorithm	Facebook	Twitter	Gplus	Youtube	LiveJournal
GCN_Cheby	0.915 ± 0.0018	0.972 ± 0.0029	0.787 ± 0.0062	0.812 ± 0.0077	0.810 ± 0.0071
GCNs	0.914 ± 0.0018	0.954 ± 0.0110	0.716 ± 0.0101	0.889 ± 0.0146	0.901 ± 0.0042
GATs	0.970 ± 0.0016	0.967 ± 0.0036	0.732 ± 0.0227	0.827 ± 0.0028	0.892 ± 0.0080
DGI	0.936 ± 0.0013	0.954 ± 0.0023	0.771 ± 0.0087	0.227 ± 0.0015	0.592 ± 0.0136
H-GCN	0.982 ± 0.0015	0.943 ± 0.0045	0.914 ± 0.0032	0.915 ± 0.0045	0.888 ± 0.0011
DPSW	0.892 ± 0.	0.789 ± 0.	0.922 ± 0.	0.892 ± 0.	0.872 ± 0.
CenGCN_D	**0.992 ± 0.0014**	**0.987 ± 0.0010**	**0.949 ± 0.0017**	**0.920 ± 0.0013**	0.912 ± 0.0015
CenGCN_TD	0.970 ± 0.0018	0.982 ± 0.0120	0.943 ± 0.0033	0.914 ± 0.0016	0.897 ± 0.0018
CenGCN_AD	0.970 ± 0.0012	0.969 ± 0.0121	0.933 ± 0.0026	0.915 ± 0.0025	0.910 ± 0.0045
CenGCN_WD	0.832 ± 0.0012	0.965 ± 0.0015	0.941 ± 0.0019	0.904 ± 0.0018	0.903 ± 0.0018
CenGCN_ID	0.888 ± 0.0024	0.967 ± 0.0018	0.861 ± 0.0044	0.893 ± 0.0027	0.893 ± 0.0020
CenGCN_E	**0.992 ± 0.0015**	**0.987 ± 0.0012**	0.936 ± 0.0018	0.919 ± 0.0015	**0.903 ± 0.0030**
CenGCN_TE	0.912 ± 0.0019	0.905 ± 0.0010	0.717 ± 0.0014	0.873 ± 0.0028	0.894 ± 0.0017
CenGCN_AE	0.916 ± 0.0011	0.930 ± 0.0072	0.742 ± 0.0027	0.866 ± 0.0026	0.900 ± 0.0015
CenGCN_WE	0.932 ± 0.0013	0.973 ± 0.0053	0.717 ± 0.0028	0.892 ± 0.0017	0.895 ± 0.0028
CenGCN_IE	0.912 ± 0.0017	0.971 ± 0.0018	0.870 ± 0.0021	0.902 ± 0.0031	0.877 ± 0.0043

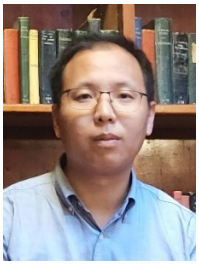
TABLE 6
The AUC score of link prediction.

Algorithm	Facebook	Twitter	Gplus	Youtube	LiveJournal
GCN_Cheby	0.672 ± 0.0126	0.842 ± 0.0013	0.725 ± 0.0091	0.676 ± 0.0026	0.759 ± 0.0199
GCNs	0.809 ± 0.0060	0.729 ± 0.0028	0.711 ± 0.0019	0.578 ± 0.0087	0.711 ± 0.0019
GATs	0.633 ± 0.0041	0.852 ± 0.0065	0.558 ± 0.0057	0.685 ± 0.0091	0.757 ± 0.0179
DGI	0.723 ± 0.0081	0.862 ± 0.0016	0.678 ± 0.0031	0.613 ± 0.0022	0.621 ± 0.0041
H-GCN	0.708 ± 0.0180	0.564 ± 0.0206	0.601 ± 0.0204	0.656 ± 0.0145	0.739 ± 0.0024
DPSW	0.767 ± 0.	0.581 ± 0.	0.797 ± 0.	0.714 ± 0.	0.753 ± 0.
CenGCN_D	**0.892 ± 0.0030**	**0.873 ± 0.0095**	**0.801 ± 0.0022**	**0.731 ± 0.0114**	0.848 ± 0.0089
CenGCN_TD	0.854 ± 0.0017	0.857 ± 0.0053	0.787 ± 0.0049	0.718 ± 0.0063	0.850 ± 0.0011
CenGCN_AD	0.885 ± 0.0098	0.855 ± 0.0026	0.775 ± 0.0026	0.713 ± 0.0298	0.837 ± 0.0245
CenGCN_WD	0.884 ± 0.0335	0.850 ± 0.0154	0.796 ± 0.0057	0.728 ± 0.0067	0.831 ± 0.0013
CenGCN_ID	0.882 ± 0.0025	0.847 ± 0.0129	0.699 ± 0.0071	0.713 ± 0.0108	0.828 ± 0.0019
CenGCN_E	0.891 ± 0.0014	0.871 ± 0.0058	0.769 ± 0.0191	0.727 ± 0.0149	**0.853 ± 0.0014**
CenGCN_TE	0.887 ± 0.0023	0.858 ± 0.0017	0.753 ± 0.0057	0.715 ± 0.0138	0.850 ± 0.0016
CenGCN_AE	0.868 ± 0.0012	0.856 ± 0.0069	0.746 ± 0.0024	0.756 ± 0.0114	0.841 ± 0.0015
CenGCN_WE	0.808 ± 0.0154	0.861 ± 0.0025	0.742 ± 0.0204	0.698 ± 0.0226	0.848 ± 0.0011
CenGCN_IE	0.840 ± 0.0438	0.856 ± 0.0048	0.776 ± 0.0042	0.681 ± 0.0077	0.842 ± 0.0067

Lei Wang received the BSc degree in software engineering from Dalian University of Technology, China, in 2018. He is currently working toward the master degree in the School of Software, Dalian University of Technology, China. His research interests include data mining, analysis of complex networks, and machine learning.

Xin Chen received the B.Sc. degree in information security from Harbin Engineering University, Harbin, China, in 2020. He is currently pursuing the master's degree in the School of Software, Dalian University of Technology, China. His research interests include graph learning, urban science, and social computing.

Tao Tang received the Bachelor Degree from Chengdu College, University of Electronic Science and Technology of China, Chengdu, China in 2019. He is currently pursuing the Ph.D. degree in School of Engineering, IT and Physical Sciences, Federation University Australia. His research interests include data science, recommender systems, and graph learning.



Xiangjie Kong (M'13-SM'17) received the B.Sc. and Ph.D. degrees from Zhejiang University, Hangzhou, China. He is currently a Professor with College of Computer Science and Technology, Zhejiang University of Technology. Previously, he was an Associate Professor with the School of Software, Dalian University of Technology, China. He has published over 160 scientific papers in international journals and conferences. His research interests include network science, data science, and computational social science. He is a Senior Member of IEEE.

Giles Oatley received his PhD in Artificial Intelligence in 2000, and his highest position in the UK was Reader (Associate Professor) in Intelligent Systems at Cardiff Metropolitan University, before continuing his academic career in Australia since 2016. For over 20 years he has researched in data mining with particular emphasis on crime informatics, the resultant analyses often embedded in decision support systems. He is a Fellow and Chartered IT Professional with the BCS, The Chartered Institute for IT, and Charted Professional with the ACS (Australian Computing Society).

Irwin King (F'19) received the B.Sc. degree in engineering and applied science from the California Institute of Technology, Pasadena, CA, USA, and the M.Sc. and Ph.D. degrees in computer science from the University of Southern California, Los Angeles, CA. He is currently a Professor at the Department of Computer Science and Engineering, and a former Associate Dean (Education), Faculty of Engineering at The Chinese University of Hong Kong. His research interests include machine learning, social computing, web intelligence, data mining, and multimedia information processing. In these research areas, he has over 210 technical publications in journals and conferences. He is a Fellow of IEEE.