Neurocomputing 497 (2022) 13–27

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A new approach for evaluating node importance in complex networks via deep learning methods ☆

Min Zhang [a,b], Xiaojuan Wang [a,*], Lei Jin [a], Mei Song [a], Ziyang Li [c]

[a] School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing, China
[b] School of Science, Beijing University of Posts and Telecommunications, Beijing, China
[c] School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan, China

ARTICLE INFO

Article history:
Received 19 August 2021
Revised 3 January 2022
Accepted 2 May 2022
Available online 6 May 2022
Communicated by Zidong Wang

Keyword:
Influential node
Complex network
Convolutional neural network
Graph neural network

ABSTRACT

The evaluation of node importance is a critical research topic in network science, widely applied in social networks, transport systems, and computer networks. Prior works addressing this topic either consider a single metric or assign weights for multiple metrics or select features by handcraft, which exist one-sidedness and subjectivity issues. In this paper, to tackle these problems, we propose a new approach named CGNN to identify influential nodes based on deep learning methods, including Convolutional Neural Networks (CNNs) and Graph Neural Networks (GNNs). CGNN obtains the feature matrices by the contraction algorithm and gets the labels by the Susceptible-Infected-Recovered (SIR) model, which will be leveraged for learning the hidden representations of nodes without utilizing any network metrics as features. We adopt three evaluation criteria to verify CGNN concerning effectiveness and distinguishability, including Kendall's τ correlation coefficient, monotonicity index (*MI*), and ranking distribution function (*RDF*). Nine baselines are employed to compare with CGNN on thirty synthetic networks and twelve real-world networks from different domains. Simulation results demonstrate that CGNN manifests better performance than the baselines, in which the values of τ are large and significantly increase, the values of *MI* approach to 1, and the points in the *RDF* curves distribute more uniformly. These results may provide reference significance for controlling epidemic spreading and enhancing network robustness.

1. Introduction

Complex networks are potent means for researching complex systems due to their theoretical significance and practical application, involving biological networks, nervous systems, transportation networks, power grids, computer systems. More and more researchers have been devoting themselves to the study of complex networks, such as link prediction, cascading failure, community detection, controllability, synchronization, and robustness [1–7]. Evaluating node importance has gained much attention in complex networks with many practical applications. Some quintessential examples include (1) In the recommender system, discerning essential commodities according to the customer preference can improve service quality and attain better benefits. (2) In the traffic system, identifying and protecting the critical junctions can alleviate traffic congestion. (3) In the social network, finding influential spreaders can control the diffusion of rumors or accelerate the flow of information.

As far as we know, there are three main methods to evaluate node importance in complex networks, including traditional methods, machine learning-based methods, and deep learning-based methods. Traditional methods could be categorized into two classes. One is the network metric according to the topology structure, such as degree centrality [8], betweenness centrality [9], the K-shell algorithm [10]. The other is the network destructiveness, the inspiration of which comes from node deletion will decrease network robustness, cascading failure [11], for instance. The worse the network robustness, the more influential the node is. Machine learning-based methods include Gradient Boosting Decision Tree (GBDT) [12], Least Squares Support Vector Machine (LS-SVM) [13], Ensemble Learning (EL) [14], Competitive Learning (CL) [15], K-Nearest Neighbor (KNN) [16], and so on. Deep learning-based methods contain Reinforcement Learning (RL) [17,18], Graph Neural Network (GNN) [19], Convolutional Neural Network (CNN) [20], and so on. These methods have laid a solid foundation for further study of evaluating node importance. Unfortunately, most of them

National Natural Science Foundation of China (Grant No. 61871046)
* Corresponding author.
E-mail address: wj2718@163.com (X. Wang).

https://doi.org/10.1016/j.neucom.2022.05.010
0925-2312/

have some shortcomings and limitations. Some traditional methods have a one-sided perspective, such as degree centrality considers the local network structure yet ignores the global network information; betweenness centrality concerns the global network structure but neglects the network dynamics. Moreover, some prior researches have an inevitable subjectivity, like several fusion methods have to assign weights for multiple metrics; machine learning-based and deep learning-based methods need to manually select network metrics as features.

To solve these problems, we propose a new approach named CGNN to evaluate node importance in complex networks, primarily based on deep learning methods. There are two challenges we are facing. **First, how to overcome the one-sidedness of a single metric.** We utilize CNNs and GNNs to learn the hidden representations of nodes, extracting more information of network structures and node features. **Second, how to break through the difficulty of assigning weights or selecting features.** We employ the contraction algorithm to generate feature matrices without calculating network metrics by handcraft. Besides, we adopt several evaluation criteria for effectiveness and distinguishability to verify CGNN, including Kendall's τ correlation coefficient, monotonicity index, and ranking distribution function. Nine baselines are applied to compare with CGNN on thirty synthetic networks and twelve real-world networks. Extensive experiments show that CGNN manifests better performance than the baselines.

This paper is organized as follows. Section 2 presents some related works, including traditional methods, machine learning-based methods, deep learning-based methods, and the contraction algorithm. Section 3 concerns the methodology used for this study, such as generating feature matrices via the contraction algorithm, obtaining node labels by the SIR model, and constructing the CGNN model. Section 4 conducts extensive experiments and presents a detailed analysis of the model. Section 5 is the conclusion of the paper.

2. Related work

Evaluating node importance in complex networks is a worthwhile task, which has been widely applied in most realistic scenarios. For example, how to identify the susceptible persons in novel pneumonia (COVID-19) network and then protect them to prevent the spread of the disease. This paper aims to sort nodes according to their importance, such as sequence the people based on their infectibility.

A lot of approaches have been proposed to evaluate node importance, which could be categorized into three classes: traditional methods, machine learning-based methods, and deep learning-based methods. Therefore, we will introduce some relevant works in Section 2.1 and Section 2.2. Besides, the contraction algorithm plays a vital role in generating feature matrices, the detailed process of which will be presented in Section 2.3.

2.1. Traditional methods

Traditional methods have been widely investigated in the past few years, which contain four types of network metrics methods (1–4) and two kinds of network destructiveness methods (5–6). **(1)** Considering the local network structure, the methods include degree centrality, H-index, clustering coefficient, closeness centrality, etc. Xu [21] proposed an adjacency information entropy method according to the adjacency degree of nodes. Sheng [22] presented a novel algorithm covering closeness centrality and the nearest neighbor nodes. The disadvantage of this kind of method is that it only considers the local information of nodes and ignores their environment, such as the node location. **(2)** Concerning the global network structure, the methods comprise betweenness centrality, network diameter, average path length, eigenvector centrality, etc. Fei [23] put forward a ranking method based on node intensity which depends on the distance between two nodes. Lv [24] introduced an approach that considered the changes of the average shortest path in networks. However, this type of approach has high computational complexity, which restricts the application in the large-scale network. **(3)** Regarding the node location, the methods consist of the K-shell algorithm, and so on. Wang [25] considered the iteration information produced in K-shell decomposition, i.e., two nodes are removed in a different order, but their K-shell values are identical. Li [26] differentiated the influence of various nodes by categorizing their neighbors into four classes, taking into account the K-shell values and removal orders. Sara [27] defined the influential ability of nodes as the product of eigenvector centrality and coreness. The downside to these methods is that the obtained results are coarse-grained, such as nodes may have the same K-shell values. That makes the discrimination of nodes small, resulting in the difficulty of comparing their importance. **(4)** According to the network dynamics, the methods have Google's PageRank algorithm, VoteRank algorithm, etc. Liu [28] proposed a dynamics-sensitive centrality, in which the topological features and dynamical properties were taken into consideration. However, the particular network structure may affect the results of this method, such as the tightly-knit community will cause the phenomenon of the topic draft in the World Wide Web. **(5)** In consideration of the propagation ability of nodes, the methods contain cascading failure, numerous infection models, etc. Chen [29] adopted the probability model to calculate the ranking scores of nodes from the perspective of propagation dynamics. The disadvantage of this method is that it needs to artificially define the propagation probability in advance, the difference of which will cause various ranking lists. **(6)** Considering the contractility of nodes, the method condenses some nodes into one node and then obtains the changes of the shortest paths in networks. Yu [11] proposed two evaluation methods to identify influential nodes based on the propagation ability and the contractility of nodes. The method has a drawback because it has high computational complexity and low efficiency.

In a further step, some researchers have been focusing on assigning weights for the above metrics to generate a fusion index. Yang [30] calculated the weights of multiple metrics by the Vlsekriterijumska Optimizacija I Kompromisno Resenje (VIKOR) method, which is one of the Multi-Criteria Decision-Making (MCDM) methods. Mo [31] utilized Dempster–Shafer (D-S) theory to compute the weights of four metrics, comprising degree centrality, betweenness centrality, efficiency centrality, and correlation centrality. Chen [32] adopted the genetic algorithm to get the optimal weights of five centrality metrics, including degree centrality, betweenness centrality, closeness centrality, core centrality, and neighborhood core centrality.

2.2. Machine learning and deep learning-based methods

Machine learning and deep learning-based methods open up a new direction and provide a new thought for evaluating node importance in recent years. Wen [13] adopted LS-SVM to train an evaluation model and obtained the evaluation results. Li [14] used the EL algorithm to identify essential nodes by virtue of estimating network robustness. Fan [17] found an optimal set of nodes via the RL algorithm, in which graph states and actions were represented by inductive graph representation learning. Tian [18] abstracted the identification process of important nodes as a Partially Observable Markov Decision Process (POMDP) and applied the RL algorithm to solve it. Zhao [19] proposed a deep learning framework named InfGCN, in which four network metrics and neighbor graphs

were taken as the input. Yu [20] converted the significant nodes identification problem into a regression problem by CNNs. The label was determined by infected scale, and the feature matrix was obtained by adjacency matrix and node degree. The downside to these methods is that the features are selected by handcraft, which inevitably brings about subjective influence.

2.3. Contraction algorithm

The contraction algorithm has demonstrated significant applications in circuits, network flow, lattice, and traffic assignment [33–36]. To describe the contraction algorithm, we will present the process of contracting node x and illustrate a quintessential example.

Step 1. Finding the first-order neighbors and the second-order neighbors of node x.

Step 2. Merging node x and its first-order neighbors as a new node $x\prime$ and removing the links between them.

Step 3. Connecting the new node $x\prime$ and the second-order neighbors of node x.

Step 4. Removing the links between the first-order neighbors and the second-order neighbors of node x.

As shown in Fig. 1(a), there are ten nodes and eighteen links in the Kite network. The first-order neighbors of node 1 are nodes of 2, 3, 4, and the second-order neighbors are nodes of 5, 6, 7, 8. Then we merge nodes of 1, 2, 3, 4 as a new node $1\prime$ and delete the links between them, involving (1,2), (1,3), (1,4), (2,3), (2,4), (3,4). After that, we connect the new node $1\prime$ and nodes of 5, 6, 7, 8. In the end, the links between nodes of 2, 3, 4 and nodes of 5, 6, 7, 8 are removed, including (2,5), (3,6), (3,8), (4,5), (4,6), (4,7).

3. Method

In this section, we will introduce the CGNN model to evaluate node importance. First of all, we utilize the contraction algorithm to generate feature matrices (see Section 3.1), which will be used to learn the hidden representations of nodes. Then we employ the SIR model to obtain node labels (see Section 3.2), which will be applied to minimize the loss function in the training process and be adopted as a standard benchmark in the testing process. At last, we illustrate the relevant details of the CGNN model (see Section 3.3).

3.1. Feature matrix

We use $G = (V, E)$ to represent a network, where V and E are two one-dimensional arrays that store the information of nodes and links in the network, respectively. $N = |V|$ denotes the number of nodes. Adjacency matrix A is a two-dimensional array that shows the relationship between nodes. If there exists a link between node i and node j, then $A_{ij} = 1$. Conversely, $A_{ij} = 0$. In a network, a path is a nodes sequence representing a route from one node to another along with links. The length of a path is the number of links in a route. The shortest path is the route with the least number of links. If the shortest path length of two nodes is r, one node is the rth-order neighbor of the other.

Prior researches generally obtain feature matrices by artificially selecting network metrics as features. However, part of metrics are difficult to compute in large-scale networks, betweenness centrality, for instance. In this paper, we generate feature matrices by the contraction algorithm without selecting metrics by handcraft. The construction process of feature matrices is divided into four steps as below.

Step 1. For each node $x \in G$, we select its l neighbors by the breadth-first search algorithm, i.e., choose l neighbors from the sequence consists of first-order neighbors, second-order neighbors, $\ldots$, rth-order neighbors. If the number of neighbors of node x is $l\prime < l$, we select its $l\prime$ neighbors. If two neighbors u and v belong to the same order, they are arranged based on node degree $d_u > d_v$.

Step 2. According to the obtained neighbors of node x, we get the sub-graph $G_A(x)$ and the symmetrical adjacency matrix $A(x)$. If $l\prime < l$,

$$A_{ij}(x) = 0,$$

where $i = 1, 2, \ldots, l+1, j = l\prime + 2, l\prime + 3, \ldots, l+1$.

Step 3. By the contraction algorithm mentioned in Section 2.3, we acquire the condensed sub-graph $G_B(x)$ and the condensed adjacency matrix $B(x)$ of node x.

Step 4. Making the subtraction and taking the absolute values between the adjacency matrix $A(x)$ and the condensed adjacency matrix $B(x)$, we derive the feature matrix $F(x)$ of node x, i.e.,

$$F(x) = \{A(x) - B(x)\}, \tag{1}$$

where $\{\cdot\}$ refers to taking the absolute values of elements in the matrix.

The physical significance of the feature matrix $F(x)$ is that it depicts the change of network state. The adjacency matrix $A(x)$

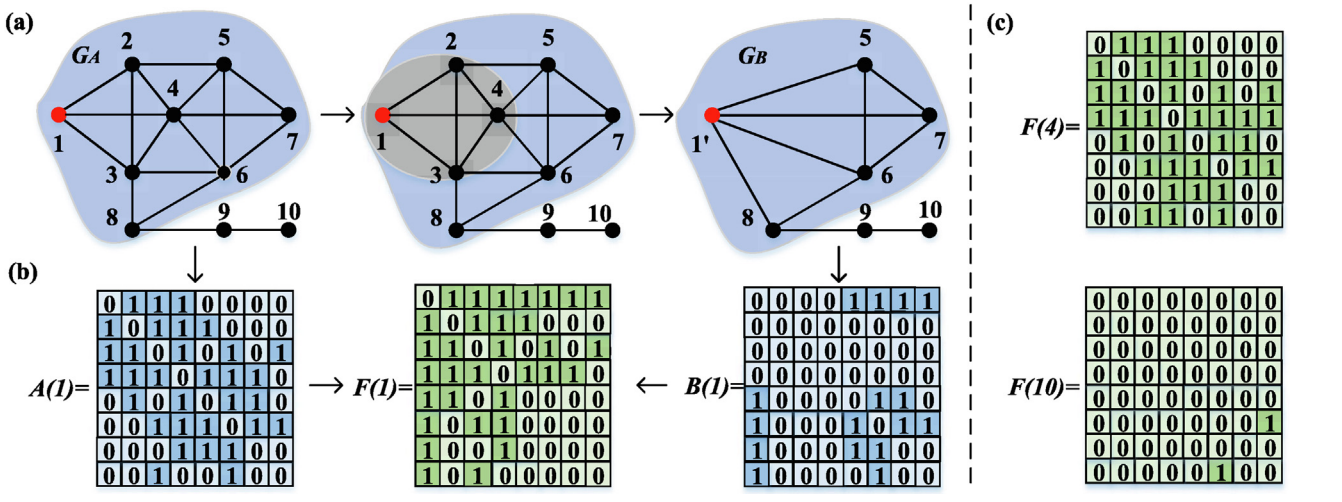


Fig. 1. (a) The process of contracting node 1 by the contraction algorithm in the Kite network. G_A is the sub-graph, and G_B is the condensed sub-graph. (b) The construction process of the feature matrix regarding node 1. A is the adjacency matrix, B is the condensed adjacency matrix, and F is the feature matrix. (c) The feature matrices of node 4 and node 10.

and the condensed adjacency matrix $B(x)$ reflect the network state before and after contracting node x respectively. As illustrated in Fig. 1(b), A_{ij} reflects the current network state that whether a direct influence exists between node i and node j. For example, $A_{1j} = 1, j = 2, 3, 4$, shows that nodes of 2, 3, 4 could directly affect node 1; $A_{86} = 1$ denotes that node 8 is the first-order neighbor of node 6. Similarly, B_{ij} describes the condensed network state. For example, $B_{1j} = 0, j = 2, 3, 4$ and $B_{1k} = 1, k = 5, 6, 7, 8$ show that node 1 is not affected by nodes of 2, 3, 4, but nodes of 5, 6, 7, 8; $B_{86} = 1$ describes that node 8 still has a direct influence on node 6. $F_{ij} = \{A_{ij} - B_{ij}\}$ represents the change of network state. For example, $F_{1j} = 1, j = 2, 3, 4, 5, 6, 7, 8$ reflects the change in direct neighbors of node 1, and $F_{86} = 0$ shows that there is no change in the state between node 6 and node 8 after contracting node 1. In addition, the more significant the change of network state, the more the numbers of $F_{ij} = 1$. For example, contracting node 4 will cause greater changes in the network state than contracting node 10. As shown in Fig. 1(c), the number of $F_{ij} = 1$ in $F(4)$ is more than that in $F(10)$.

As mentioned above, the feature matrix $F(x)$ generated by the contraction algorithm breaks through the difficulty of selecting features by handcraft. Besides, it also conquers the one-sidedness of a single metric because it contains multiple features of node x. For example, the larger the degree of node x, the more compact the network structure after contracting node x, and the more significant network state change. That means the feature matrix $F(x)$ could reflect the characteristic of the local network structure. If node x is in a hub location, the contraction of which will reduce the average path length and change the network state. That means the feature matrix $F(x)$ could depict the feature of the node location. By applying the deep learning method on the feature matrix $F(x)$ which severs as a receptive field, we learn the representation of node x to reflect multiple features of node x. Moreover, we adopt a baseline named GLS in Section 4, which considers the global influence and the local influence of nodes simultaneously. Experiment results show that CGNN performs better than GLS, demonstrating that CGNN overcomes the one-sidedness of a single metric.

3.2. Label

The SIR model is a classical infectious disease model in which S is the susceptible state, I is the infected state, and R is the recovery state. In the SIR model, each infected node has the probability θ to infect its first-order susceptible neighbors and has the probability β to recover. Moreover, each recovery node will not be infected. This process will iterate until there is no other node in the infected state.

To generate the label of node $x \in G$, the state of node x is initialized as infected, and that of other nodes is set as susceptible. By calculating the number of infected nodes and recovery nodes when the network achieves a stable state, we obtain the infection capability of node x and define it as the label. To ensure the objectivity of infection capability and reduce the impact of noise, the labels of nodes are the mean values of 100 independent experiments, namely

$$IA(x) = \frac{\sum_{j=1}^{100}(N_j^i + N_j^r)}{100 \times N}, \tag{2}$$

where $IA(x)$ is the infection capability of node x, N_j^i and N_j^r are the numbers of infected nodes and recovery nodes in the j-th experiment.

The infection probability threshold θ_c is obtained by virtue of the mean-field theory [37–39], i.e.,

$$\theta_c = \frac{\langle k \rangle}{\langle k^2 \rangle - \langle k \rangle}, \tag{3}$$

where k is the node degree, $\langle \cdot \rangle$ signifies calculating the mean value. We will apply $\theta = \zeta \times \theta_c$ to conduct experiments in Section 4, in which $\zeta \geqslant 0$ is a variable parameter that could adjust the infection probability θ.

3.3. Model

As illustrated in Fig. 2, the CGNN model consists of four parts, involving the input layer, feature processing layer, hidden layer, and output layer. Relevant implement details of the CGNN model on an instance are shown below.

(1) *Input layer*. The Kite network G with ten nodes and eighteen links is fed into the model without selecting classic metrics as node features in advance.

(2) *Feature processing layer*. As illustrated in Section 3.1, for each node $x \in G$, we obtain its l neighbors and adjacency matrix $A(x)$. By the contraction algorithm, we get the condensed adjacency matrix $B(x)$. In the light of Eq. (1), we derive the feature matrix $F(x)$ of node x.
For example, let $l = 7$, we obtain seven neighbors of node 1, in which the first-order neighbors are nodes of 2, 3, 4, and the second-order neighbors are nodes of 5, 6, 7, 8. Therefore, we get the sub-graph G_A and the local adjacency matrix $A_{8\times8}$ shown in Fig. 1. By the contraction algorithm, the condensed sub-graph G_B is formed and the condensed adjacency matrix $B_{8\times8}$ is constructed. In the end, we have the feature matrix $F_{8\times8}$ of node 1 via Eq. (1). Since there are ten nodes in the network G, we could obtain ten feature matrices.

(3) *Hidden layer*. Based on the feature matrix $F(x)$, we learn the representation of node x by hidden layers, consisting of three parts of CNN layers, fully connected (FC) layer and GNN layers. In the **CNN layer**, there are 2 convolutional layers, 2 activation layers, and 2 pooling layers. In the first convolutional layer, input channel is 1, output channel is 10, kernel size is 3×3, stride is 1, and padding is 1. In the second convolutional layer, input channel is 10, output channel is 20, kernel size is 3×3, stride is 1, and padding is 1. The activation function is LeakyReLU and the pooling layer is 2×2 max pooling. In the **FC layer**, input feature is $((l+1)^2/4^2) \times 20$, and output feature is 6. The **GNN layer** is defined as:

$$e^{i+1} = \sigma(Ae^iW^i + b^i), \tag{4}$$

where e^i is the representations of nodes at the i-th layer, σ is the sigmoid function, A is the adjacency matrix of network G, W^i and b^i are learnable weights and bias parameters respectively. In this paper, we set two GNN layers, in which $W^0 \in R^{6\times2}, b^0 \in R^2, W^1 \in R^{2\times4}, b^1 \in R^4$. The loss function is the mean squared error (MSE), and the optimizer is Adam with a learning rate of 0.01.
For example, the hidden representation of node 1 initially is the feature matrix $F_{8\times8}$ which will be fed into the hidden layer, and then becomes a 6-dimensional vector via two CNN layers and one FC layer, and eventually be a 4-dimensional vector by two GNN layers.

(4) *Output layer*. Based on the hidden representations of nodes, the output layer will export a score for each node via L2-norm and the sigmoid function,

$$Score = \sigma(\|e\|_2),$$

where e denotes the hidden representations of nodes. Sorting the scores in descending order, we obtain the ranking list of influential nodes in network G. The reason why we export scores is that the labels represent the nodes' infection capability IA. Since $0 \leqslant IA \leqslant 1$ shown in Eq. (2), we here apply

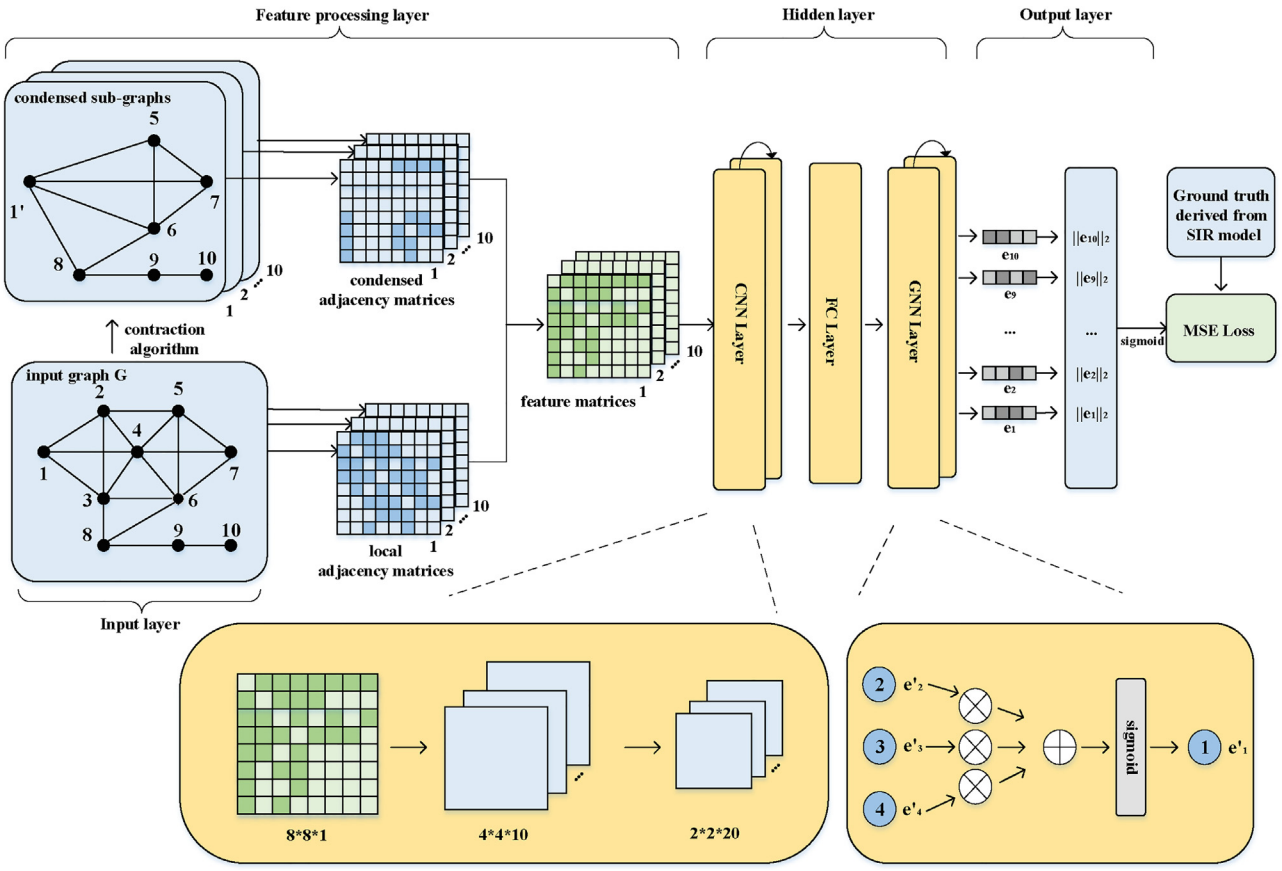


Fig. 2. The CGNN model is divided into four parts, including the input layer, feature processing layer, hidden layer, and output layer. The network G is fed into the input layer. The feature processing layer obtains the feature matrix of each node by the contraction algorithm. The hidden layer learns nodes' representations by virtue of two CNN layers, one FC layer, and two GNN layers. The output layer enables a score for each node. The larger the value of the score, the more influential the node is.

L2-norm and the sigmoid function on nodes representations to approximate *IA*.

It is worth noting that the CGNN model is trained over several synthetic networks generated by the BA model. The labels of nodes obtained by the SIR model are applied to minimize the loss function. Moreover, to evaluate the CGNN model, the labels of nodes will be adopted as a standard benchmark for calculating Kendall's τ correlation coefficient in Section 4.

4. Experiment

In this section, we will conduct extensive experiments to evaluate and analyse the deep learning framework CGNN. To verify the effectiveness and the distinguishability of CGNN, we utilize three evaluation criteria of Kendall's τ correlation coefficient, monotonicity index, and ranking distribution function (see Section 4.1). Furthermore, we adopt nine baselines as comparisons, including RCNN, GLS, InfGCN, URank, betweenness centrality, degree centrality, K-shell, VoteRank, and H-index (see Section 4.2). Besides, we introduce the datasets of thirty synthetic networks and twelve real-world networks (see Section 4.3). And finally, we discuss the simulation results and analyse the model (see Section 4.4 and 4.5).

4.1. Evaluation criterion

(1) Kendall's τ correlation coefficient. We adopt Kendall's τ correlation coefficient to quantify the correlation between two ranking lists. The definition of τ is

$$\tau = \frac{2(N_c - N_d)}{N_L(N_L - 1)}, \tag{5}$$

where N_L is the number of nodes in the ranking list, N_c and N_d are the number of concordant pairs and discordant pairs, respectively. Let (x_i, y_i) and (x_j, y_j) be a pair of data. If $x_i > x_j, y_i > y_j$ or $x_i < x_j, y_i < y_j$, the pair (x_i, y_i) and (x_j, y_j) is concordant. If $x_i > x_j, y_i < y_j$ or $x_i < x_j, y_i > y_j$, the pair (x_i, y_i) and (x_j, y_j) is discordant. If $x_i = x_j, y_i = y_j$, the pair is neither concordant nor discordant.

Kendall's τ correlation coefficient ranges from -1 to 1. The larger the values of τ, the higher the correlation of two ranking lists. If $\tau = 1$, the two ranking lists have a highly consistent correlation. If $\tau = 0$, the two ranking lists are uncorrelated. If $\tau = -1$, the two ranking lists have an utterly opposite correlation. In Section 4.4 and 4.5, we will utilize τ to quantify the correlation between the ranking list obtained by the SIR model and the ranking lists derived by baselines and CGNN.

(2) Monotonicity index (*MI*). Since some nodes may have the same ranking, we adopt *MI* to quantitatively evaluate the distinguishability of nodes in the ranking list as a whole. The definition of *MI* is

$$MI = \left(1 - \frac{\sum_{\alpha \in L\prime} N_\alpha(N_\alpha - 1)}{N_L(N_L - 1)}\right)^2, \tag{6}$$

where N_L is the number of nodes in the ranking list, $L\prime$ is the ranking value list, $\alpha \in L\prime$ is the ranking value, N_α is the number of nodes with the same ranking value α. *MI* ranges from 0 to 1. The larger the values of *MI*, the higher the distinguishability of nodes in the ranking

list. If $MI = 1$, each node has a ranking value. If $MI = 0$, all the nodes have the same ranking value.

(3) Ranking distribution function (*RDF*). We adopt *RDF* to describe and visualize the distinguishability of nodes in the ranking list in detail. The definition of *RDF* is

$$RDF(\alpha) = P(\alpha\prime = \alpha), \tag{7}$$

where $\alpha\prime$ is a random variable, α is the ranking value, P is the probability function. The wider the ranking values distribute, the higher the distinguishability of nodes in the ranking list. Furthermore, the cumulative ranking distribution function (*CRDF*) is defined as

$$CRDF(\alpha) = P(\alpha\prime \leqslant \alpha). \tag{8}$$

If many nodes have the same ranking values, the *CRDF* curve will grow rapidly. On the contrary, the *CRDF* curve will become large bit by bit. The slower the *CRDF* curve grows, the higher the distinguishability of nodes in the ranking list.

4.2. Baseline

(1) Degree centrality (DC) [8].

$$DC(i) = \frac{k(i)}{N-1}, \tag{9}$$

where $k(i)$ is the degree of node i, N is the number of nodes in the network.

(2) Betweenness centrality (BC) [9].

$$BC(i) = \sum_{u \neq i \neq v} \frac{g_{uv}^{i}}{g_{uv}}, \tag{10}$$

where g_{uv} represents the number of shortest paths between node u and node v, g_{uv}^{i} is the number of shortest paths passing through node i.

(3) K-shell (KS) [10]. The detailed process of the K-shell algorithm is presented as follows. Firstly, the algorithm deletes the 1-degree nodes and their links. After the deletion operation, new nodes with 1-degree will appear in the network. Iterate the above deletion operation until there are no 1-degree nodes in the network. All the deleted nodes with 1-degree constitute the first layer, namely 1-shells, and the K-shell value of nodes is 1. Secondly, the algorithm removes the nodes with 2-degree and creates 2-shells. Repeat this process to get 3-shells, 4-shells, and so on. Finally, each node will belong to a specific shell, and all the nodes in the same shell have identical K-shell values. The larger the values of K-shell, the more influential the node is.

(4) InfGCN [19]. To extract the features of non-Euclidean data, Thomas [40] proposed the Graph Convolutional Network (GCN), which has been successfully applied to social networks, recommendation systems, knowledge graphs, and other fields [41,42]. InfGCN is a GCN-based model to identify influential nodes, consisting of an input layer, a GCN layer, three FC layers, and an output layer. In the input layer, the neighbor network and the feature vector of each node are fed into the model, in which the feature vector is composed of four classic metrics. In the GCN layer, input feature is 16^2 and output feature is 8. The weight matrices of three FC layers are $W_1 \in R^{8\times16}, W_2 \in R^{16\times8}, W_3 \in R^{8\times2}$ respectively. Each of the FC layers is followed by Exponential Linear Unit (ELU) function. In the output layer, the outputs are fed into the LogSoftMax classifier.

(5) RCNN [20]. CNN [43] is one of the most widely applied neural networks, playing a crucial role in engineering applications, such as malicious traffic detection, image recognition, and commodity classification [44–46]. In this paper, we adopt RCNN proposed by Yu as a baseline, involving two convolutional layers, two pooling layers, and a FC layer. In the first convolutional layer, input channel is 1, output channel is 16, kernel size is 5×5, stride is 1, and padding is 2. In the second convolutional layer, input channel is 16, output channel is 32, kernel size is 5×5, stride is 1, and padding is 2. The activation function is ReLU, the pooling layer is 2×2 max pooling, and the FC layer is $32 \times l/4 \times l/4$.

(6) URank [21]. Xu proposed the URank algorithm for unweighted-undirected networks based on node adjacency information entropy, which is defined as

$$E(i) = -\sum_{j\in\Gamma_i} P_{i_j} \log_2 P_{i_j}, \tag{11}$$

where j is the neighbor of node i, Γ_i is the set of neighbors of node i, $P_{i_j} = k_i/A_j$ is the selection probability, k_i is the degree of node i, $A_j = \sum_{u\in\Gamma_j} k_u$ is adjacency degree of node j. The larger the values of entropy, the more vital the nodes are.

(7) GLS [22]. Sheng proposed the GLS algorithm, a fusion approach measuring the influence of nodes by their global impact and local impact. The influence of node i is

$$\begin{aligned} &Influ(i) = glbinflu(i) \times locinflu(i), \\ &glbinflu(i) = k_i \sum_{j\in\Gamma_i} 1.1^{\#Com(i,j)}, \ locinflu(i) = \sum_{j\in\Gamma_i} DC(j)p(j), \end{aligned} \tag{12}$$

where $glbinflu(i)$ and $locinflu(i)$ are the global and local impacts, Γ_i is the set of neighbors of node i, $\#Com(i,j)$ is the number of common neighbors between node i and node j, $DC(j) = \frac{d(j)}{n-1}$ is the degree centrality of node j, $p(j) = \frac{d(j)}{\sum_{u\in\Gamma_j} d(u)}$.

(8) VoteRank (VR) [47]. The main idea of the VoteRank algorithm is to choose vital spreaders according to voting scores. The selected spreader does not participate in the subsequent voting, and the voting ability of its neighbors decreases subsequently. The VoteRank algorithm consists of five steps. The first step is initializing. For each node in the network, the voting score is 0, and the number of votes is 1. The second step is voting. Each node and its neighbors vote for each other and then obtain their voting scores. The third step is selecting. The most influential spreader is determined based on the voting scores. The fourth step is updating, i.e., weakening the voting ability of nodes that vote for the most influential spreader. The fifth step iterates the second step to the fourth step until several spreaders are obtained.

(9) H-index (HI) [48]. To characterize the scientific output of a researcher, Hirsch proposed the H-index, which denotes H papers of a researcher that have been cited at least H times. The higher the values of H-index, the more influential the researcher is. The calculation process of H-index is presented as follows. Firstly, ranking the publications in descending order according to the number of citations. Secondly, defining the H-index as the highest-ranking that is less than or equal to the number of citations. Nowadays, H-index has played a substantial role in various applications, such as identifying influential nodes. The larger the values of H-index, the more influential the node is.

4.3. Datasets

To verify the effectiveness and the distinguishability of the CGNN model, we adopt thirty synthetic networks and twelve real-world networks to conduct experiments. Synthetic networks are generated by the BA model [49] with different node number N and average degree $\langle k\rangle$, namely $N = 500, 1000, 2000, 3000, 4000$ and $\langle k\rangle = 2, 4, 6, 8, 10, 12$. The BA model is an evolutionary network with scale-free property, the node degree of which follows the power-law distribution. In the BA model, the degrees of most nodes are small except that a few nodes have larger degrees. Because complex systems exhibit power-law distribution, such as in the personal income network, most people earn less money, and a few gather a lot of

wealth, we generate the synthetic networks by the BA model. Table 1 lists the statistical indicators of partial synthetic networks. Besides, we select twelve real-world networks from different fields, including (1) Zachary karate club (ZA) [50]. A human social network, in which each node represents a member in the club and each link shows the relationship between two members. (2) Gene fusion (GF) [51]. A gene network, in which each node represents a gene and each link denotes two genes are fused during the emergence of cancer. (3) Caenorhabditis elegans (CE) [52]. A metabolic network, in which each node is the substrate and each link means the metabolic reaction. (4) Euroroads (ET) [53]. An infrastructure network, in which each node stands for a city and each link denotes a Euroroad connecting two cities. (5) Network science (NS) [54]. A co-authorship network, in which each node is an author and each link describes the co-authorship. (6) Hamster (SH) [55]. A human social network, in which each node represents a user and each link shows the relationship between two users in the website. (7) Erdos (EO) [56]. A co-authorship network contains people who have directly or indirectly written papers with Paul Erdos. (8) Pretty good privacy (PG) [57]. An online contact network, in which each node is a user and each link indicates the interaction between users. (9) Sister cities (SC) [58]. A cities network, in which each node means a city

Table 1
The statistical indicators of synthetic networks with $\langle k\rangle = 2$.

	N	M	ρ	$\langle k\rangle$	k_{max}	NCC	LCC	c	AS	HE
SN-1	500	499	0.0040	2	39	1	500	0	−0.1352	3.1064
SN-2	1000	999	0.0020	2	53	1	1000	0	−0.1241	3.3021
SN-3	2000	1999	0.0010	2	96	1	2000	0	−0.1033	4.4226
SN-4	3000	2999	0.0006	2	86	1	3000	0	−0.1035	4.0748
SN-5	4000	3999	0.0005	2	134	1	4000	0	−0.0736	4.9712

Table 2
The statistical indicators of real-world networks.

	N	M	ρ	$\langle k\rangle$	k_{max}	NCC	LCC	c	AS	HE
ZA	34	78	0.1390	4.5882	17	1	34	0.5706	-0.4756	1.6933
GF	291	279	0.0066	1.9175	34	32	110	0.0006	-0.3546	3.8599
CE	453	2025	0.0198	8.9404	237	1	453	0.6465	-0.2258	4.4850
ET	1174	1417	0.0021	2.4140	10	26	1039	0.0167	0.1266	1.2425
NS	1461	2742	0.0026	3.7534	34	268	379	0.6937	0.4616	1.8486
SH	2426	16631	0.0057	13.711	273	148	2000	0.5376	0.0474	3.1010
EO	6927	11850	0.0005	3.4214	507	1	6927	0.1239	-0.1155	12.671
PG	10680	24316	0.0004	4.5536	205	1	10680	0.2659	0.2382	4.1465
SC	14274	20573	0.0002	2.8826	99	1433	10320	0.0427	0.3867	3.2210
CA	26475	106762	0.0001	4.0325	2628	1	26475	0.2082	-0.1946	69.496
CM	30460	120029	0.0002	7.8810	202	896	27519	0.6460	0.1782	2.7583
TO	34761	171403	0.0001	6.1977	2760	1	34761	0.2890	-0.2148	51.544

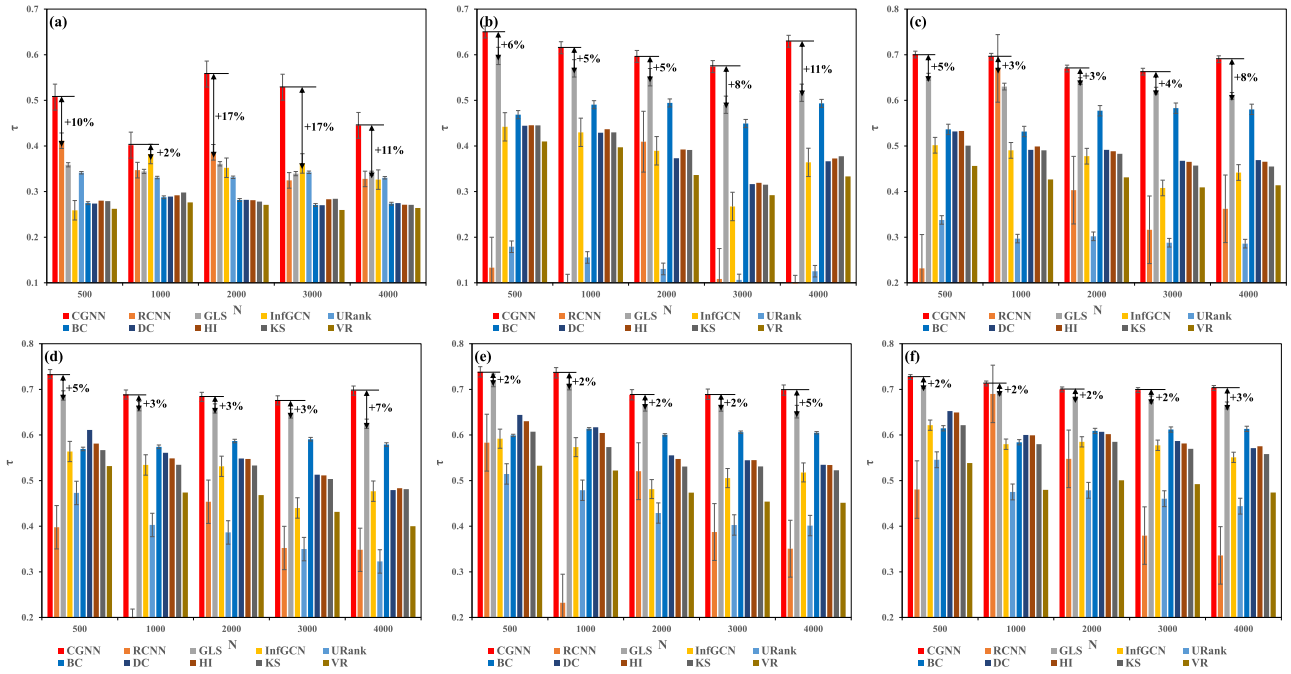


Fig. 3. The Kendall's τ correlation coefficients between the ranking lists obtained by various methods and those obtained by the SIR model over thirty synthetic networks with different $\langle k\rangle$ and N. The CGNN model manifests better performance than the baselines, and the increments of τ are displayed. The abscissa axis represents the number of nodes, and the ordinate axis denotes Kendall's τ correlation coefficient. The parameters are $l = 16$, $\beta = 1$, $\zeta = 2$. (a) Synthetic networks with $\langle k\rangle = 2$. (b) Synthetic networks with $\langle k\rangle = 4$. (c) Synthetic networks with $\langle k\rangle = 6$. (d) Synthetic networks with $\langle k\rangle = 8$. (e) Synthetic networks with $\langle k\rangle = 10$. (f) Synthetic networks with $\langle k\rangle = 12$.

and each link represents the relationship of "sister city" or "twin city". (10) Caida (CA) [59]. An autonomous system network, in which each node is an independent system and each link denotes communication. (11) Condensed matter (CM) [60]. A co-authorship network in the area of condensed matter physics. (12) Internet topology (TO) [61]. A computer network, in which each node is the IP address and each link denotes the connection between them. The statistical indicators of real-world networks are summed up in Table 2.

In Table 1 and Table 2, N is the number of nodes, M is the number of links, ρ is the network density, $\langle k\rangle$ is the average degree, k_{max} is the largest degree, NCC is the number of connected components, LCC is the number of nodes in the largest connected component, c is the average clustering coefficient, AS is the degree assortativity, HE is the degree heterogeneity.

4.4. Results

4.4.1. Results on synthetic network.

To begin with, we show the simulation results on synthetic networks based on Kendall's τ correlation coefficient (5). It can be seen from Fig. 3 that CGNN performs better than the baselines under the circumstances that $\langle k\rangle$ varies from 2 to 12 and N changes from 500 to 4000. It means that the ranking list derived by CGNN is more correlative to the ranking list obtained by the SIR model. Fig. 3 also shows the increments of τ of CGNN relative to the best baseline. As illustrated in Section 3.1, the feature matrix F contains multiple node features, such as local network structure and node location. Therefore, the hidden representation of each node obtained by CGNN integrates numerous features, resulting in better performance than the traditional methods. For example, CGNN behaves better than the fusion index GLS which considers the global and local impact. That means the difficulty of the one-sidedness of a single metric could be overcome. Moreover, the weights are trained by the Adam optimizer, and the feature matrices are obtained by the contraction algorithm. It indicates that CGNN breaks through the difficulty of assigning weights and selecting features by handcraft.

Moreover, we illustrate the simulation results on synthetic networks from the perspective of the monotonicity index (6). Table 3 displays the MI values of partial synthetic networks. The values of MI obtained by CGNN and RCNN are substantially larger than those derived by other baselines. It implies that most nodes can be accurately distinguished from each other based on CGNN and RCNN. It is worth noting that Kendall's τ correlation coefficients of RCNN are small, shown in Fig. 3(a). It indicates that the ranking list derived by RCNN is not intensely correlative, although it has high distinguishability. Overcoming the one-sidedness is reflected in the larger MI values. As shown in Table 3, the values of MI are equal to 0

Table 3
The monotonicity indices of various methods over partial synthetic networks. The values of MI obtained by CGNN and RCNN are larger than those derived by other baselines, which indicates that CGNN and RCNN obtain superior performances in terms of distinguishability. The parameters are $\langle k\rangle = 2, l = 16, \beta = 1, \zeta = 2$. All these quantities are accurate to four decimal places.

	CGNN	RCNN	GLS	InfGCN	URank	BC	DC	HI	KS	VR
SN-1	0.9898	**0.9925**	0.9800	0.9765	0.9617	0.2843	0.2713	0.0579	**0.0000**	0.0068
SN-2	**0.9946**	0.9879	0.9874	0.9834	0.9733	0.3124	0.2941	0.0717	**0.0000**	0.0086
SN-3	**0.9956**	0.9942	0.9890	0.9744	0.9760	0.2914	0.2767	0.0624	**0.0000**	0.0060
SN-4	**0.9970**	0.9942	0.9913	0.9681	0.9777	0.2756	0.2618	0.0614	**0.0000**	0.0081
SN-5	**0.9972**	0.9937	0.9922	0.9724	0.9778	0.2824	0.2681	0.0642	**0.0000**	0.0068

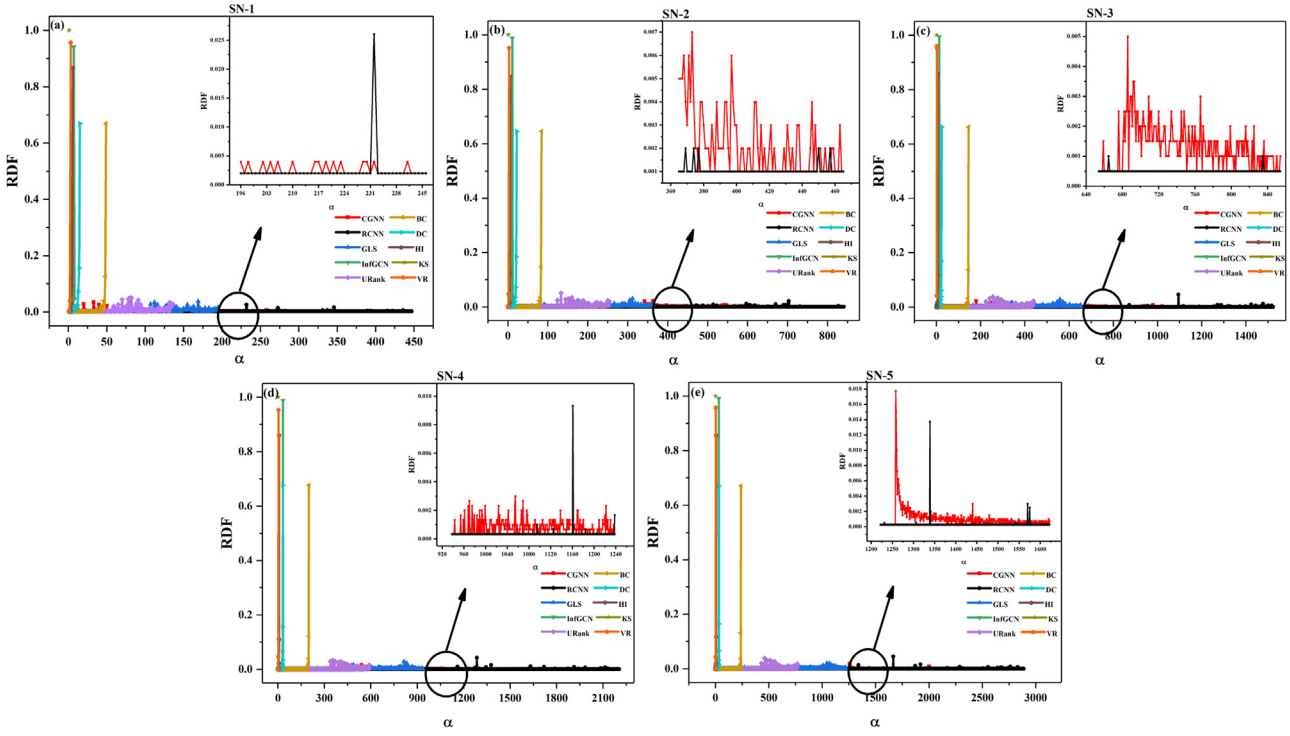


Fig. 4. The ranking distribution functions of various methods over partial synthetic networks, in which the insets are the ranking distributions of the $N \times 10\%$ ranking values derived by all models. The points in the RDF curves obtained by CGNN and RCNN distribute more uniformly. The parameters are $\langle k\rangle = 2, l = 16, \beta = 1, \zeta = 2$.

in the KS algorithm. The reason lies that it only considers the node location. Consequently, the KS values of nodes are identical, and all nodes have the same ranking value. Similar to the above mentioned, the problems of assigning weights and artificially selecting features are solved.

What is more, we analyse the simulation results on synthetic networks according to the ranking distribution function (7). As shown in Fig. 4, the abscissa axis means the ranking value, and the ordinate axis denotes ranking value distribution. Apparently, the points in the *RDF* curves of CGNN and RCNN are nearly coincident, which distribute more uniformly than other baselines. It suggests that each node almost has its ranking value under the applications of CGNN and RCNN. The insets are the ranking distribution of the $N \times 10\%$ ranking values derived by all methods. Obviously, only the *RDF* curves of CGNN and RCNN exist. It means the largest ranking values of other methods are relatively small, and most nodes have an identical ranking. The lower distinguishability of traditional methods shows their one-sidedness and demonstrates the advantage of the CGNN. For example, as illustrated in Fig. 4(a), 67% nodes have the same ranking value 49 obtained by the BC method. Similarly, the second challenge is conquered.

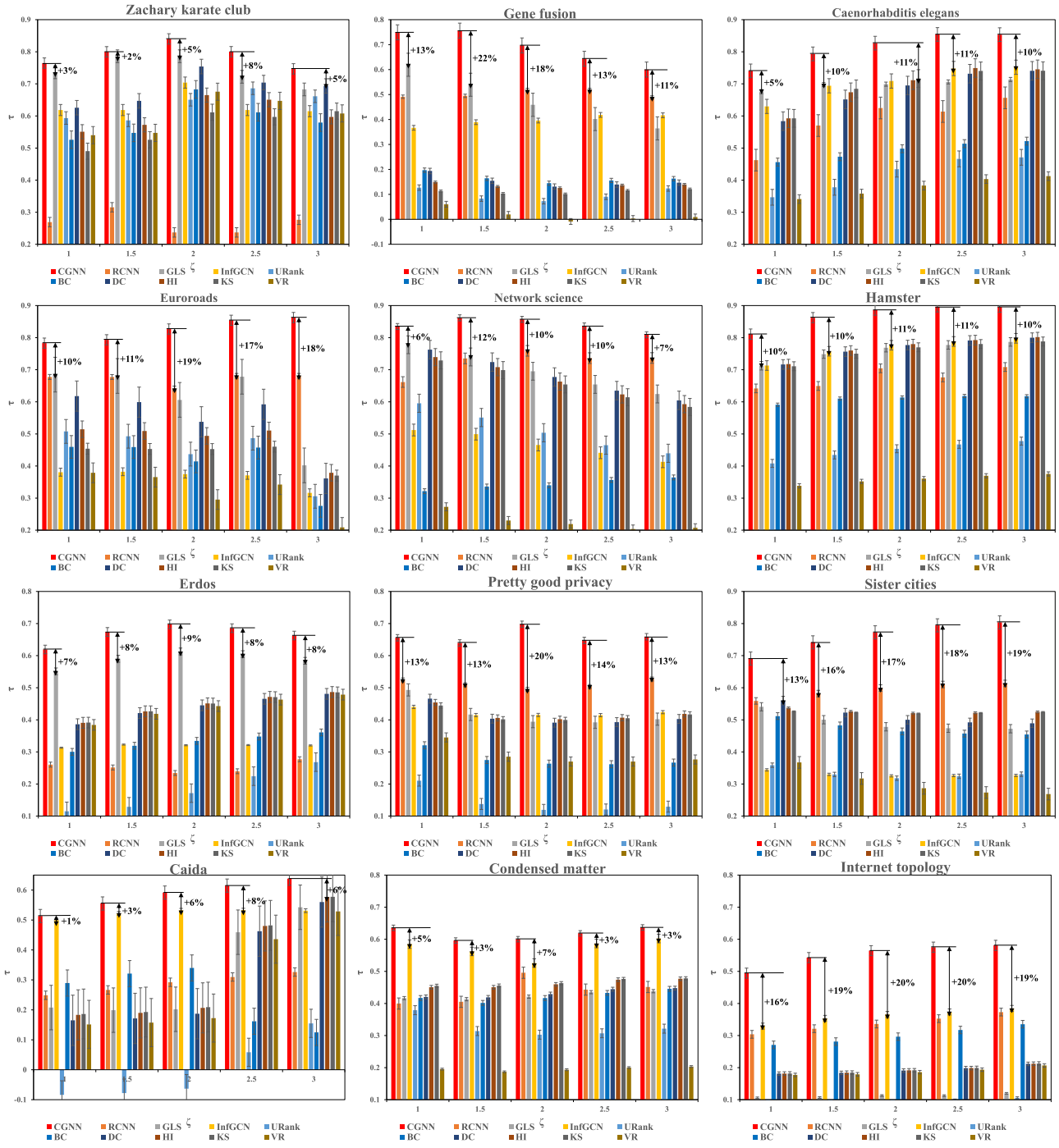


Fig. 5. The Kendall's τ correlation coefficients between the ranking lists obtained by various methods and those obtained by the SIR model over twelve real-world networks. The CGNN model performs better than the baselines and the increments of τ are displayed. The abscissa axis is a variable parameter that could adjust the infection probability θ, and the ordinate axis denotes Kendall's τ correlation coefficient. The parameters are $l = 16, \beta = 1$.

4.4.2. Results on real-world network.

First and foremost, we consider the simulation results on real-world networks based on Kendall's τ correlation coefficient (5). As illustrated in Fig. 5, the values of τ derived by CGNN are much larger than those obtained by the baselines. It indicates that CGNN improves the effectiveness of identifying important nodes. For example, in the Gene fusion network, the largest and the smallest increments of τ are 22% and 11% respectively, and in the Internet topology network, the increments of τ are 20% and 16% separately. The reason is that CGNN could capture multiple features of nodes, resulting in more comprehensive results than the single metric. To overcome the second challenge, the contraction algorithm is applied to obtain node features automatically, and the weights are trained by the Adam optimizer.

In a further step, we show the simulation results on real-world networks from the perspective of monotonicity index (6). As shown in Table 4, the *MI* values of CGNN, RCNN and InfGCN are larger than those of other baselines. For example, in the Euroroads network (ET), the *MI* values of CGNN, RCNN and InfGCN are 0.9990, 0.9975 and 0.9989 respectively, the largest and the smallest *MI* values of other baselines are 0.9824 and 0.2129 separately. However, Kendall's τ correlation coefficients of CGNN are exceedingly higher than those of RCNN and InfGCN illustrated in Fig. 5. It indicates that CGNN could obtain a strong correlation and high distinguishability ranking list. The larger *MI* values show that the two challenges have been overcome.

Furthermore, we present the simulation results on real-world networks according to the ranking distribution function (7). Fig. 6 reveals that the ranking values obtained by CGNN and RCNN distribute more uniformly, similarly to Fig. 4. That means the ranking lists have high distinguishability. Besides, the insets display the cumulative ranking distribution (see Eq. (8)) of two methods with the largest Kendall's τ correlation coefficient. For example, Fig. 5 shows the τ values of CGNN and RCNN are the largest in the Sister cities network when $\zeta = 2$. As a result, the inset of Fig. 6(i) shows the *CRDF* curves of CGNN and RCNN. In the Sister cities network, the *CRDF* curve of CGNN grows slower than that of RCNN when α is small, and the reverse process occurs when α is large. That means the nodes with the same ranking value distribute differently in the ranking list obtained by CGNN and RCNN. For CGNN, the top influential nodes have their ranking, and the unimportant nodes have the same ranking. Consequently, the *CRDF* curve grows slower at first and then faster. Conversely, for RCNN, the top influential nodes may have an identical ranking, and the unimportant nodes have a different order. Hence, the *CRDF* curve grows faster at the beginning and then slower. Similar to the above analysis, the two challenges could be solved.

4.4.3. Running time.

To train and evaluate CGNN, we conduct experiments on the Intel(R) Core(TM) i7-8550U CPU processor and 8 GB running memory, adopt Pytorch as the deep learning framework, and use Python as the computer programming language. As illustrated in Fig. 7, the abscissa axis represents five synthetic networks and twelve real-world networks, and the ordinate axis denotes the running time. The insets display the running time of other approaches except for betweenness centrality and the VoteRank algorithm. Fig. 7 shows that degree centrality has the best performance from the perspective of time-consuming. The reason is that degree centrality only considers the local network structure. CGNN takes slightly longer than degree centrality because of relatively many model parameters. In the future, we will reduce the number of parameters and layers to shorten running time. Betweenness centrality and the VoteRank algorithm have poor performances. The reason lies in that the calculations of the shortest paths number and voting scores cost much time. Taken together, CGNN achieves moderate time consumption but behaves better in other aspects.

4.5. Model Analysis.

4.5.1. Ablation study.

In CGNN, we design two components for learning the hidden representations of nodes to evaluate their importance. In this Section, we make an ablation study on the four datasets to evaluate different components in CGNN. The experimental results are illustrated in Table 5.

We observe that CGNN achieves the best performance in terms of effectiveness and distinguishability on the four datasets. Besides, we discover that CNN and GNN have the same priority level. It means that CNN and GNN play an indispensable role in CGNN, the lack of any component will reduce the effectiveness and distinguishability of the evaluation. The reason is that CNN can extract and encode information of each node from feature matrix F by convolutional layer and pooling layer. Moreover, CNN can compress the feature information of each node, significantly reducing the computational complexity and improving the efficiency of the downstream GNN layer. GNN can better learn node representation by aggregating the features of high-order neighbors and capturing the information of network structure.

4.5.2. Aggregators study.

In CGNN, we adopt GNN (formula (4)) to aggregate the features of high-order neighbors and capture network structure information. To explore the impact of aggregators, we replace it with the following settings.

Table 4
The monotonicity indices of various methods over twelve real-world networks. The values of *MI* obtained by CGNN, RCNN, and InfGCN are larger than those derived by other baselines, which suggests that CGNN, RGNN, and InfGCN obtain better performances with regard to distinguishability. The parameters are $l = 16, \beta = 1, \zeta = 2$. All these quantities are accurate to four decimal places.

	CGNN	RCNN	GLS	InfGCN	URank	BC	DC	HI	KS	VR
ZA	0.9612	**0.9995**	0.9541	0.5289	0.9576	0.7754	0.7079	0.5766	0.4958	0.4398
GF	0.9138	**0.9512**	0.9074	0.7419	0.8935	0.1618	0.1524	0.0735	0.0492	0.0322
CE	0.9967	0.9987	0.9984	**0.9995**	0.9984	0.8741	0.7922	0.7311	0.6962	0.6375
ET	**0.9990**	0.9975	0.9824	0.9989	0.9731	0.9374	0.4442	0.2534	0.2129	0.2650
NS	**0.9189**	0.9116	0.9167	0.9158	0.9119	0.1049	0.7069	0.6711	0.6634	0.6302
SH	**0.9860**	0.9848	0.9856	0.9846	0.9852	0.7123	0.8980	0.8835	0.8714	0.7685
EO	0.9947	0.9863	0.9951	**0.9997**	0.9940	0.1619	0.2539	0.2491	0.2479	0.1840
PG	**0.9998**	0.9990	0.9984	0.9997	0.9961	0.5122	0.6193	0.5172	0.4806	0.3008
SC	**0.9656**	0.9635	0.9518	0.9655	0.9360	0.4828	0.4614	0.3278	0.2747	0.1660
CA	**0.9972**	0.9920	0.9971	0.9970	0.9970	0.5113	0.4783	0.4356	0.4286	0.3689
CM	0.9926	0.9972	0.9974	**0.9976**	0.9973	0.4576	0.8462	0.8111	0.7933	0.7096
TO	**0.9985**	0.9870	0.9984	0.9494	0.9984	0.5384	0.5266	0.4918	0.4856	0.4100

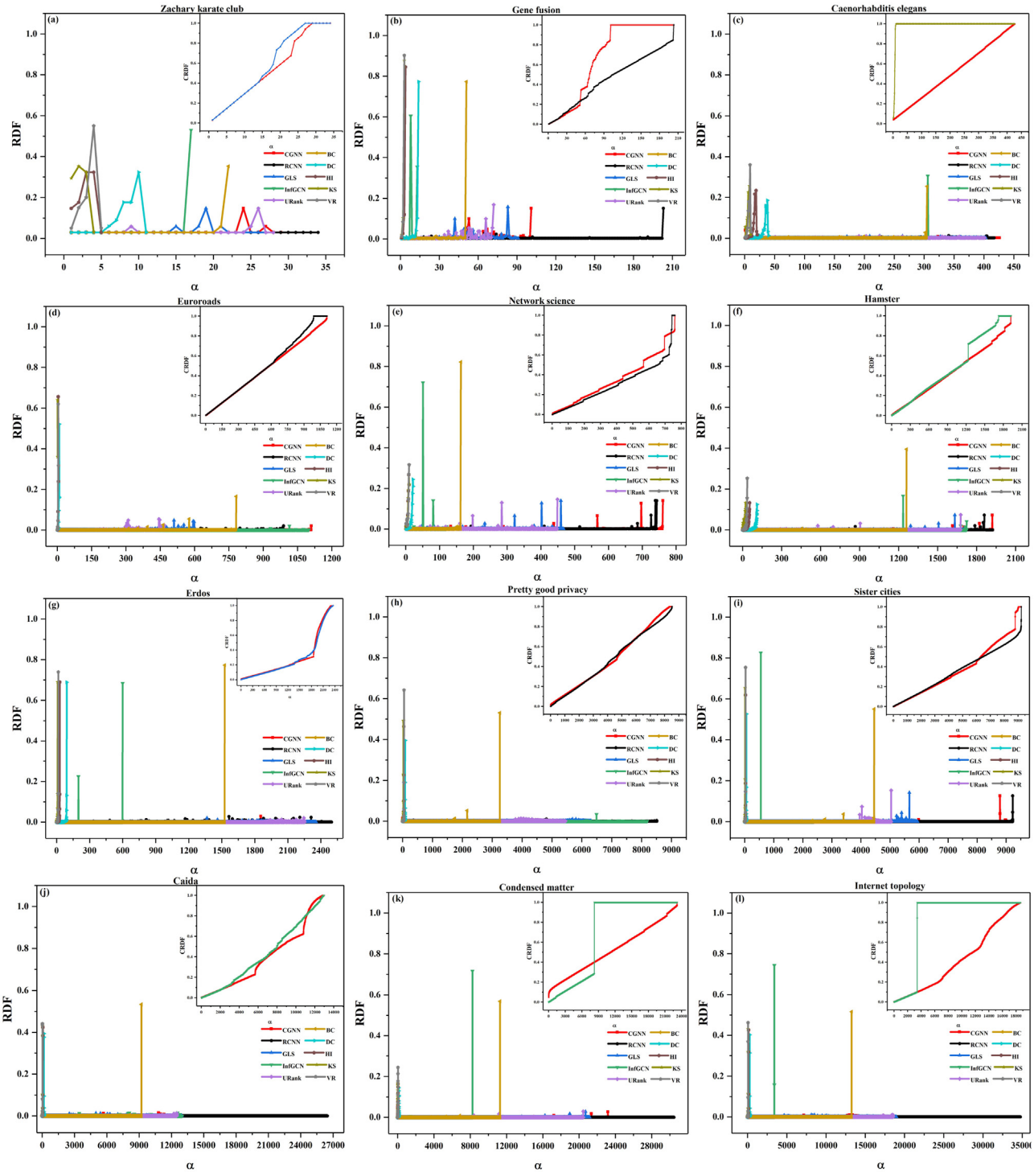


Fig. 6. The ranking distribution functions of various methods over twelve real-world networks, in which the insets are the cumulative ranking distribution functions of two methods with the largest Kendall's τ correlation coefficient. The points in the *RDF* curves obtained by CGNN and RCNN distribute more uniformly. The parameters are $l = 16, \beta = 1, \zeta = 2$.

- GraphSAGE [62] concatenates the node's current representation e_i with the mean neighboring feature vector e_{Γ_i}, and applies a FC layer with nonlinear transformation σ:

$$f_{GraphSAGE} = \sigma(W(e_i \| e_{\Gamma_i})), \ e_{\Gamma_i} = mean(\{e_j, j \in \Gamma_i\}),$$

  where $\|$ represents the concatenation operation, W is the weight matrix, Γ_i is the set of neighbors of node i.
- GAT [63] computes the normalized attention coefficients a_{ij} and makes a linear combination of the features:

$$f_{GAT} = \sigma(\sum_{j \in \Gamma_i} a_{ij} W e_j), \ a_{ij} = \frac{exp(LeakyReLU(\vec{a}^T [We_i \| We_j]))}{\sum_{k \in \Gamma_i} exp(LeakyReLU(\vec{a}^T [We_i \| We_k]))},$$

where $\|$ represents the concatenation operation, $\cdot^T$ is transposition, W and $\vec{a}$ is the weight matrix and the weight vector respectively, Γ_i is the set of neighbors of node i.

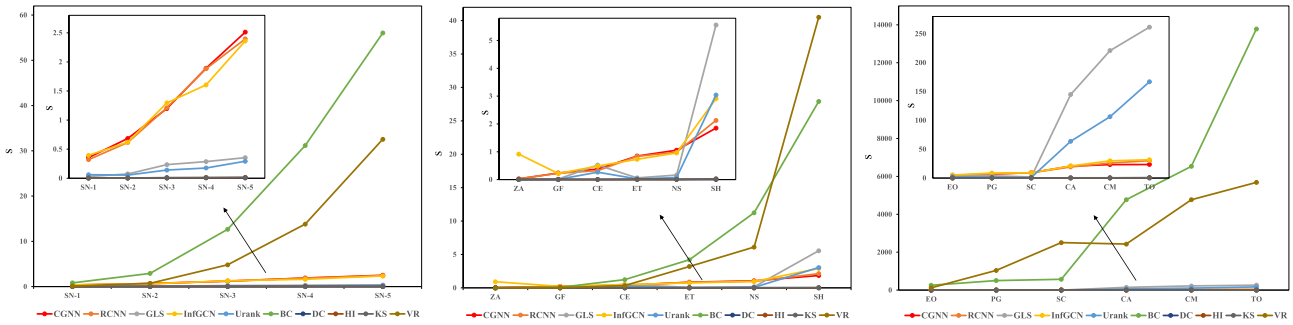


Fig. 7. The running time of various methods over five synthetic networks and twelve real-world networks. The time consumption of degree centrality is the least. On the contrary, betweenness centrality and the VoteRank algorithm have poor performances. The time consumption of CGNN is moderate. The parameters are $l = 16, \beta = 1, \zeta = 2$, and the average degree of synthetic networks is $\langle k \rangle = 2$.

Table 5
Ablation study of CGNN on the four datasets. The parameters are $l = 16, \beta = 1, \zeta = 2$. All these quantities are accurate to four decimal places.

Model	SN-2		SN-4		PG		CA	
	τ	*MI*	τ	*MI*	τ	*MI*	τ	*MI*
w/o GNN	0.2215	**0.9999**	0.4470	0.9992	0.5036	0.9990	0.3141	0.9920
w/o CNN	0.6785	0.9993	0.4508	0.9989	0.3802	0.9980	0.5726	**0.9972**
CGNN	**0.7394**	**0.9999**	**0.7011**	**0.9999**	**0.5957**	**0.9997**	**0.6126**	**0.9972**

Table 6
Effect of four types of aggregators. The parameters are $l = 16, \beta = 1, \zeta = 2$. All these quantities are accurate to four decimal places.

Model	SN-2		SN-4		PG		CA	
	τ	*MI*	τ	*MI*	τ	*MI*	τ	*MI*
w/ GraphSAGE	0.3214	**0.9999**	0.3801	**0.9999**	0.4653	**0.9999**	0.2316	**0.9977**
w/ GAT	0.2245	**0.9999**	0.2101	**0.9999**	0.2898	0.9997	0.1301	0.9972
w/ DAGG	0.4333	**0.9999**	0.3974	**0.9999**	0.2259	**0.9999**	0.1778	0.9919
CGNN	**0.7394**	**0.9999**	**0.7011**	**0.9999**	**0.5957**	0.9997	**0.6126**	0.9972

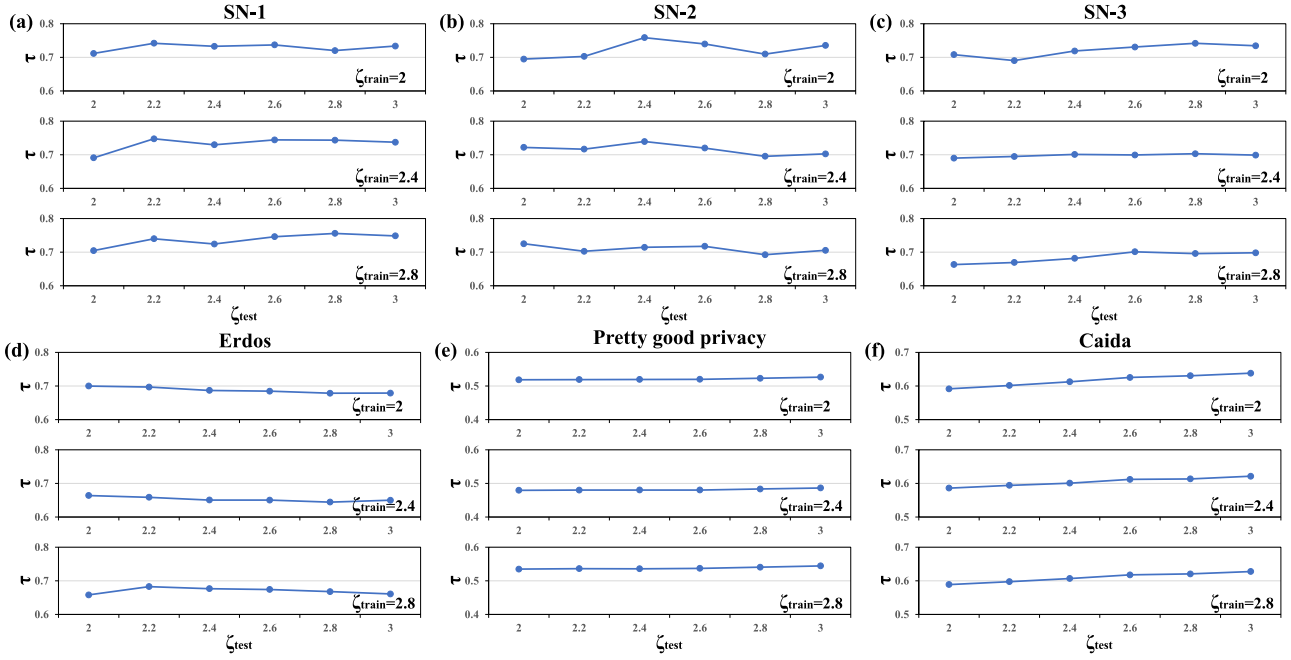


Fig. 8. The Kendall's τ correlation coefficients between the ranking lists obtained by the CGNN model and those obtained by the SIR model under varying ζ_{test} and ζ_{train}. The performance of CGNN is relatively stable with the change of ζ_{test} and ζ_{train}. The parameters are $l = 16, \beta = 1$.

Table 7
The monotonicity indices of the CGNN model with varying ζ_{test} and ζ_{train}. The performance of CGNN is relatively stable with the change of ζ_{test} and ζ_{train}. The parameters are $l = 16, \beta = 1$. All these quantities are accurate to four decimal places.

Network	ζ_{test} = 2			ζ_{test} = 3		
	ζ_{train} = 2	ζ_{train} = 2.4	ζ_{train} = 2.8	ζ_{train} = 2	ζ_{train} = 2.4	ζ_{train} = 2.8
SN-1	0.9999	0.9999	0.9999	0.9997	0.9998	0.9997
SN-2	0.9999	0.9999	0.9999	0.9999	0.9999	0.9999
SN-4	0.9996	0.9999	0.9999	0.9997	0.9998	0.9999
EO	0.9951	0.9940	0.9841	0.9951	0.9940	0.9841
PG	0.9992	0.9961	0.9938	0.9992	0.9961	0.9938
CA	0.9972	0.9972	0.9972	0.9972	0.9972	0.9972

- DAGG [64] directly applies $softmax(ReLU(E_A \cdot E_A^T))$ to replace $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ for avoiding unnecessary and repeated calculations, the formula of which is

$$f_{DAGG} = (I_N + softmax(ReLU(E_A \cdot E_A^T)))X\Theta,$$

where I_N is the identity matrix, A is the adjacent matrix, D is the degree matrix, $E_A \in R^{N \times d_e}$ is a learnable node embedding dictionary, X is the input, Θ is the learnable weight.

As shown in Table 6, replacing aggregators will have worse performance in terms of the values of τ. The possible reason is that GraphSAGE focuses on learning a representation method of nodes instead of learning features of nodes. GAT specifies different weights to different neighbors, which may lead to overfitting. DAGG uses E_A instead of $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ and hence loses network structure information, resulting in poor performance in the test set. Besides, the values of *MI* of four types of aggregators are nearly consistent. The reason is that the deep learning methods can generate a score with sixteen decimal places for each node, resulting in the ranking list having high distinguishability.

4.5.3. Discussion.

It is worth noting that the SIR model could not be adopted as a baseline. One reason is the SIR model has the infection probability θ, illustrated in Section 3.2. Therefore, the infection capabilities of nodes are random and should be calculated many times to obtain the approximate values, which will take a long time when the network is extensive in scale. Besides, the infection probability θ could not be obtained in advance when facing an unknown network, which will result in the wrong infection capabilities of nodes and an incorrect ranking list. However, the behavior of CGNN is scarcely influenced by the probability θ which could be adjusted by ζ. As shown in Fig. 8 and Table 7, the performance of CGNN has no significant difference in most cases with varying ζ_{test} when ζ_{train} is fixed.

5. Conclusion

In this paper, we propose an efficient method named CGNN to evaluate node importance in complex networks based on deep learning methods, including CNNs and GNNs. Some prior researches have a certain one-sidedness and subjectivity, such as considering a single metric, assigning weights for several metrics, and manually selecting features. To solve these problems, CGNN obtains the feature matrix by the contraction algorithm and learns the hidden representations of nodes without leveraging metric knowledge of networks. To verify the effectiveness and the distinguishability of CGNN, we adopt three evaluation criteria of Kendall's τ correlation coefficient, monotonicity index (*MI*), and ranking distribution function (*RDF*). Furthermore, we apply nine baselines to compare with CGNN on thirty synthetic networks and twelve real-world networks, respectively. The simulation results show that CGNN performs better than the baselines, in which the values of τ significantly improve, the values of *MI* are closer to 1, and the points in the *RDF* curves distribute more uniformly, with the time complexity as an exception. These results provide insight into controlling the diffusion of rumors, accelerating the spread of important information, and protecting the key hubs of transportation.

CRediT authorship contribution statement

Min Zhang: Conceptualization, Methodology, Software, Writing - original draft. **Xiaojuan Wang:** Validation, Supervision. **Lei Jin:** Software, Writing - original draft. **Mei Song:** Writing - review & editing. **Ziyang Li:** Writing - review & editing.

Declaration of Competing Interest

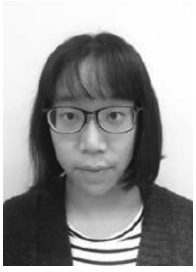
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] B. Barzel, A.L. Barabási, Network link prediction by global silencing of indirect correlations, Nat. Biotechnol. 31 (8) (2013) 720–725.
[2] S.V. Buldyrev, R. Parshani, G. Paul, H.E. Stanley, S. Havlin, Catastrophic cascade of failures in interdependent networks, Nature 464 (7291) (2010) 1025–1028.
[3] M. Zhang, X.J. Wang, L. Jin, M. Song, Cascade phenomenon in multilayer networks with dependence groups and hierarchical structure, Phys. A Stat. Mech. Appl. 581 (2021) 126201.
[4] B. Amiri, L. Hossain, J.W. Crawford, R.T. Wigand, Community detection in complex networks multi-objective enhanced firefly algorithm, Knowl.-Based Syst. 46 (2013) 1–11.
[5] Y. Liu, J.J. Slotine, A.L. Barabasi, Controllability of complex networks, Nature 473 (7346) (2011) 167–173.
[6] A. Arenas, A.D. Guilera, J. Kurths, Y. Moreno, C. Zhou, Synchronization in complex networks, Phys. Rep. 469 (3) (2008) 93–153.
[7] Y. Shang, Subgraph robustness of complex networks under attacks, IEEE. Trans. Syst. Man. Cy-S 49 (4) (2019) 821–832.
[8] P.F. Bonacich, Factoring and weighting approaches to status scores and clique identification, J. Math. Sociol. 2 (1) (1972) 113–120.
[9] L.C. Freeman, A set of measures of centrality based on betweenness, Sociometry 40 (1) (1977) 35–41.
[10] M. Kitsak, L.K. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H.E. Stanley, H.A. Makse, Identification of influential spreaders in complex networks, Nat. Phys. 6 (2010) 888–893.
[11] Z. Yu, Evaluation of node importance and invulnerability simulation analysis in complex load-network, Neurocomputing 416 (2020) 158–164.
[12] S. Deng, C. Wang, M. Wang, Z. Sun, A gradient boosting decision tree approach for insider trading identification: An empirical model evaluation of China stock market, Appl. Soft. Comput. 83 (2019) 105652.
[13] X.X. Wen, C.L. Tu, M.G. Wu, X.R. Jiang, Fast ranking nodes importance in complex networks based on LS-SVM method, Phys. A Stat. Mech. Appl. 506 (2018) 11–23.
[14] X.Y. Li, J.M. Liu, Z.J. Zhang, K.K. Gai, A new complex network robustness attack algorithm, in: Proceedings of the 2019 ACM international symposium on blockchain and secure critical infrastructure, 2019, pp. 13–17.
[15] T.C. Silva, L. Zhao, Uncovering overlapping cluster structure via stochastic competitive learning, Inf. Sci. 247 (2013) 40–61.
[16] G. Zhao, P. Jia, C. Huang, A. Zhou, Y. Fang, A machine learning based framework for identifying influential nodes in complex networks, IEEE Access 8 (2020) 65462–65471.

[17] C.J. Fan, L. Zeng, Y.Z. Sun, Y.Y. Liu, Finding key players in complex networks through deep reinforcement learning, Nat. Mach. Intell. 2 (6) (2020) 317–324.
[18] M. Tian, Z.C. Dong, X.P. Wang, Reinforcement learning approach for robustness analysis of complex networks with incomplete information, Chaos. Soliton. Fract. 144 (4) (2021) 110643.
[19] G. Zhao, P. Jia, A. Zhou, B. Zhang, InfGCN: Identifying influential nodes in complex networks with graph convolutional networks, Neurocomputing 414 (2020) 18–26.
[20] E.Y. Yu, Y.P. Wang, Y. Fu, D.B. Chen, M. Xie, Identifying critical nodes in complex networks via graph convolutional networks, Knowl-Based Syst. 198 (2020) 105893.
[21] X. Xu, C. Zhu, Q.Y. Wang, X.Q. Zhu, Y. Zhou, Identifying vital nodes in complex networks by adjacency information entropy, Sci. Rep. 10 (1) (2020) 1–12.
[22] J.F. Sheng, J.Y. Dai, B. Wang, G.H. Duan, J. Long, J.K. Zhang, K.R. Guan, S. Hu, L. Chen, W.H. Guan, Identifying influential nodes in complex networks based on global and local structure, Phys. A Stat. Mech. Appl. 541 (3) (2020) 123262.
[23] L.G. Fei, Q. Zhang, Y. Deng, Identifying influential nodes in complex networks based on the inverse-square law, Phys. A Stat. Mech. Appl. 512 (2018) 1044–1059.
[24] Z.W. Lv, N. Zhao, F. Xiong, N. Chen, A novel measure of identifying influential nodes in complex networks, Phys. A Stat. Mech. Appl. 523 (2019) 488–497.
[25] Z.X. Wang, Y. Zhao, J.K. Xi, C.J. Du, Fast ranking influential nodes in complex networks using a k-shell iteration factor, Phys. A Stat. Mech. Appl. 461 (2016) 171–181.
[26] C. Li, L. Wang, S.W. Sun, C.Y. Xia, Identification of influential spreaders based on classified neighbors in real-world complex networks, Appl. Math. Comput. 320 (2018) 512–523.
[27] A. Sara, B. Hassan, Identification of influential spreaders in complex networks using HybridRank algorithm, Appl. Math. Comput. 8 (1) (2018) 1–10.
[28] J.G. Liu, J.H. Lin, Q. Guo, T. Zhou, Locating influential nodes via dynamics-sensitive centrality, Sci. Rep. 6 (2016) 21380.
[29] D.B. Chen, H.L. Sun, Q. Tang, S.Z. Tian, M. Xie, Identifying influential spreaders in complex networks by propagation probability dynamics, Chaos 29 (3) (2019) 033120.
[30] Y.Z. Yang, L. Yu, X. Wang, Z.L. Zhou, Y. Chen, T. Kou, A novel method to evaluate node importance in complex networks, Phys. A Stat. Mech. Appl. 526 (15) (2019) 121118.
[31] H.M. Mo, Y. Deng, Identifying node importance based on evidence theory in complex networks, Phys. A Stat. Mech. Appl. 529 (6825) (2019) 121538.
[32] X. Chen, M. Tan, J. Zhao, T.H. Yang, D.Z. Wu, R.L. Zhao, Identifying influential nodes in complex networks based on a spreading influence related centrality, Phys. A Stat. Mech. Appl. 536 (1) (2019) 122481.
[33] T. Michal, G. Krzysztof, A contraction algorithm for finding all the DC solutions of piecewise-linear circuits, J. Circuit. Syst. Comp. 4 (3) (2011) 319–336.
[34] D. Kim, P.M. Pardalos, A dynamic domain contraction algorithm for nonconvex piecewise linear network flow problems, J. Global. Optim. 17 (1) (2000) 225–234.
[35] T. Doi, M.G. Endres, Unified contraction algorithm for multi-baryon correlators on the lattice, Comput. Phys. Commun. 184 (1) (2013) 117–123.
[36] A. Chen, K.L. Hong, Y. Hai, A self-adaptive projection and contraction algorithm for the traffic assignment problem with path-specific costs, Eur. J. Oper. Res. 135 (1) (2001) 27–41.
[37] Y. Liu, M. Tang, T. Zhou, Y. Do, Improving the accuracy of the k-shell method by removing redundant links from a perspective of spreading dynamics, Sci. Rep. 5 (2015) 13172.
[38] Y. Moreno, R.P. Satorras, A. Vespignani, Epidemic outbreaks in complex heterogeneous networks, Eur. Phys. J. B 26 (4) (2002) 521–529.
[39] B. Macdonald, P. Shakarian, N. Howard, G. Moores, Spreaders in the network SIR model an empirical study, arXiv preprint arXiv1208.4269, (2012).
[40] T. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks. 2016.
[41] B.W. Jin, C. Gao, X.N. He, D.P. Jin, Y. Li, Multi-behavior recommendation with graph convolutional networks, in: Proceedings of the 43rd International ACM SIGIR conference on research and development in information retrieval, 2020, pp. 659–668.
[42] L.W. Yang, Z.W. Liu, Y.T. Dou, J. Ma, P.S. Yu, ConsisRec: enhancing GNN for social recommendation via consistent neighbor aggregation, in: Proceedings of the 44th International ACM SIGIR conference on research and development in information retrieval, 2021, pp. 2141–2145.
[43] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, K.J. Lang, Phoneme recognition using time-delay neural networks, IEEE Trans. Acoust. Speech Signal Process 37 (3) (1989) 328–339.
[44] M. He, X. Wang, J. Zhou, Y. Xi, X. Wang, Deep-feature-based autoencoder network for few-shot malicious traffic detection, Secur. Commun. Netw. 2021 (6) (2021) 1–13.
[45] R. Kumar, S. Joshi, A. Dwivedi, CNN-SSPSO: a hybrid and optimized CNN approach for peripheral blood cell image recognition and classification, Int. J. Pattern. Recogn. 35 (5) (2021) 2157004.
[46] M. He, X. Wang, C. Zou, B. Dai, L. Jin, A commodity classification framework based on machine learning for analysis of trade declaration, Symmetry 13 (6) (2021) 964.
[47] J.X. Zhang, D.B. Chen, Q. Dong, Z.D. Zhao, Identifying a set of influential spreaders in complex networks, Sci. Rep. 6 (2016) 27823.
[48] J.E. Hirsch, An index to quantify an individual's scientific research output, P. Natl. Acad. Sci. 102 (46) (2005) 16569.
[49] S.V. Buldyrev, R. Parshani, G. Paul, H.E. Stanley, Catastrophic cascade of failures in interdependent networks, Nature 464 (2009) 1025–1028.
[50] W.W. Zachary, An information flow model for conflict and fission in small groups, J. Anthropol. Res. 33 (4) (1977) 452–473.
[51] M. Höglund, A. Frigyesi, F. Mitelman, A gene fusion network in human neoplasia, Oncogene 25 (18) (2006) 2674–2678.
[52] H. Jeong, B. Tombor, R. Albert, Z.N. Oltvai, A.L. Barabási, The large-scale organization of metabolic networks, Nature 407 (6804) (2000) 651–654.
[53] L. Šubelj, M. Bajec, Robust network community detection using balanced propagation, Eur. Phys. J. B 81 (3) (2011) 353–362.
[54] M.E.J. Newman, Finding community structure in networks using the eigenvectors of matrices, Phys. Rev. E 74 (3) (2006) 036104.
[55] J. Kunegis, Hamster dataset, http://konect.cc/networks/petster-hamster/.
[56] V. Batagelj, Pajek datasets, http://vlado.fmf.uni-lj.si/pub/networks/data/.
[57] M. Boguñá, R.P. Satorras, A.D. Guilera, A. Arenas, Models of social networks based on social distance attachment, Phys. Rev. E 70 (5) (2004) 056122.
[58] J. Kunegis, Sister cities dataset, http://konect.cc/networks/twin/.
[59] J. Leskovec, J. Kleinberg, C. Faloutsos, Graphs over time: densification laws, shrinking diameters and possible explanations, ACM Trans. Knowl. Discov. from Data 1 (1) (2007) 1–40.
[60] J. Kunegis, Condensed matter dataset, http://konect.cc/networks/dimacs10-cond-mat-2003/.
[61] J. Kunegis, Internet topology dataset, http://konect.cc/networks/topology/.
[62] W.L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Proceedings of the 31st international conference on neural information processing systems, 2017, pp. 1025–1035.
[63] P. Velikovi, G. Cucurull, A. Casanova, R. Adriana, L. Pietro, B. Yoshua, Graph attention networks, in: Published as a conference paper at ICLR, 2018.
[64] L. Bai, L. Yao, C. Li, X. Wang, C. Wang, Adaptive graph convolutional recurrent network for traffic forecasting, arXiv:2007.02842v1 (2020).

Min Zhang is currently pursuing the Ph.D. degree in the School of Electronic Engineering, Beijing University of Posts and Telecommunications (BUPT), Beijing, China. Before that, she graduated from the same university with the M.S. degree in 2019. Her research interests include deep learning, network science, and differential equation.



Xiaojuan Wang is currently an associate professor in the School of Electronic Engineering, Beijing University of Posts and Telecommunications (BUPT), Beijing, China. She graduated from the same university with the Ph.D. degree in 2015. Her research interests include big data, complex networks, and sensors. Furthermore, she concentrated on using the six-axis sensor for human activity recognition and the multi-source fusion in sensors. Her past interests include wireless communication, software defined network.

Lei Jin is currently pursuing the Postdoc. degree in the School of Computer Science, Beijing University of Posts and Telecommunications (BUPT), Beijing, China. Before that, he graduated from the same university with the Ph.D. degree in 2020. He received the B.S. degree in the BUPT in 2015. His research interest includes machine learning and pattern recognition, focusing on 6Dof poses estimation and Human pose estimation. What's more, he concentrated on network security and analyzed traffic security while pursuing his Ph.D. degree.

Mei Song is currently a professor and a deputy director in the key lab of ICN&CAD, School of Electronic Engineering, Beijing University of Posts and Telecommunications (BUPT), Beijing, China. She received the B.S. degree and the M.S. degree from Tianjin University, the Ph.D. degree from BUPT. Her research interests include deep learning, internet of things, big data, complex networks. (Email: songm@bupt.edu.cn).

Ziyang Li is currently pursuing the B.S. degree in the School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan, China. His research interests include deep learning, network science, and pattern recognition.