Electronic Research Archive

ERA, 31 (7): 4185–4215.
DOI: 10.3934/era.2023213
Received: 03 October 2022
Revised: 04 March 2023
Accepted: 28 March 2023
Published: 31 May 2023

http://www.aimspress.com/journal/ERA

Review

A comprehensive review of graph convolutional networks: approaches and applications

Xinzheng Xu*, Xiaoyang Zhao, Meng Wei and Zhongnian Li

School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221116, China

* **Correspondence:** Email: xxzheng@cumt.edu.cn; Tel: +15952151616.

Abstract: Convolutional neural networks (CNNs) utilize local translation invariance in the Euclidean domain and have remarkable achievements in computer vision tasks. However, there are many data types with non-Euclidean structures, such as social networks, chemical molecules, knowledge graphs, etc., which are crucial to real-world applications. The graph convolutional neural network (GCN), as a derivative of CNNs for non-Euclidean data, was established for non-Euclidean graph data. In this paper, we mainly survey the progress of GCNs and introduce in detail several basic models based on GCNs. First, we review the challenges in building GCNs, including large-scale graph data, directed graphs and multi-scale graph tasks. Also, we briefly discuss some applications of GCNs, including computer vision, transportation networks and other fields. Furthermore, we point out some open issues and highlight some future research trends for GCNs.

Keywords: graph convolutional neural network; non-Euclidean structure data; graph convolution; graph pooling

1. Introduction

In our lives, non-Euclidean structured data exist in various forms, such as molecular structures in chemical analysis, sensor networks in communication networks and social networks in social sciences. Traditional convolutional neural networks (CNNs) cannot perform local feature extraction on graph data, mainly because these data have unstructured features, and the surrounding structure of each node may be unique. Researchers still hope to extend deep learning models in non-Euclidean domains,

because deep learning is a proven powerful tool for solving a series of problems in acoustics, computer vision and natural language processing.

In recent years, more and more researchers have paid attention to using deep learning methods to analyze graph data. These methods are mainly divided into two categories. One is to embed nodes in a low-dimensional vector space to constrain proximity to learn a uniform length expression for each node, and the representative work is network embedding. The other is to apply different deep-learning models to the graph. Representative work can be divided into five categories according to the model structure and training strategy. These categories are graph recurrent neural networks, graph convolutional networks (GCNs), graph autoencoders, graph reinforcement learning and graph adversarial methods [1]. Among them, the GCN is the most outstanding branch of the graph deep learning model. GCNs play the same role as a CNN, which is a feature extractor. GCNs can process non-Euclidean structured data that traditional deep learning models such as CNN and recurrent neural networks cannot process. The core part of the GCN is introduced in Section 2.1.1 of this paper. The main contents of this paper are as follows. Section 1 introduces the development history, spectral domain methods and spatial domain methods of GCNs. Section 2 introduces some basic models of GCN and graph pooling methods. Section 3 introduces the research progress of graph convolutional networks in the face of multiple challenges. Section 4 introduces applications of GCNs in various fields. Section 5 discusses the future trends and draws the conclusion.

1.1. Development history of GCNs

The original idea of the definition of graph convolutional operation comes from the domain of graph signal processing [2–4]. Bruna et al. proposed the first CNNs formula for the graph, which is defined as using convolution in the spectral domain based on the Laplacian of the graph [5]. Duvenaud et al. [6], Kipf and Welling [7] and Atwood and Towsley [8] proposed various GCN models, in which the traditional graph filter was replaced by an adjacency matrix with self-loops. When updating network weights, propagation rules were used to calculate the output of each neural network layer. Defferrard et al. [9] extended this type of GCN model by using fast local spectral filters and effective merging operations. In recent years, there has been a surge in the application of convolution for graphs, and GCN related models have appeared in various forms. Graph convolution can be further divided into two branches: spectral convolution and spatial convolution. When a polynomial spectral kernel is used, these two methods can be overlapped.

1.2. Spectral convolution of GCNs

Spectral convolution uses the graph Fourier transform or its extension to transform the node representation into the spectral domain. The convolution theorem was used to perform convolution operations.

For the first time, the graph Laplacian matrix L was used in the spectral CNN, which has a convolutional layer like the classic Euclidean CNN proposed by Bruna et al. [5], to introduce the graph data convolution from the spectral domain. The graph convolutional operation $*_G$ is defined as follows:

$$u_1 *_G u_2 = Q((Q^T u_1) \odot (Q^T u_2)) \tag{1}$$

The Laplacian matrix L is defined as $L=D-A$, where D is the degree matrix, and A is the adjacency matrix of the graph, and the definition applies to all Laplacian matrices in this paper. $u_1, u_2 \in R^N$ are

two signals defined on the node, and N is the number of nodes. Q is the eigenvectors of L. Multiplying Q^T can transform the graph signals u_1 and u_2 into the spectral domain, that is, the graph Fourier transform. Then, multiply by Q for inverse transformation; $\odot$ represents element multiplication.

Applying different filters to different input-output signal pairs, the definition of the spectral convolutional layer is as follows:

$$u_j^{l+1} = \rho\left(\sum_{i=1}^{f_l} Q\Theta_{i,j}^l Q^T u_i^l\right) j = 1, \cdots, f_{l+1} \tag{2}$$

where l is the layer, $u_j^l \epsilon R^N$ is the jth signal of the node in the lth layer, and f_l is the dimensionality of the hidden representation in the lth layer. $\Theta_{i,j}^l$ is a learnable filter, and $\rho(\cdot)$ is the activation function. Equation (2) indicates that the input signal passes through a set of learnable filters to gather information and then performs some nonlinear transformations to obtain the output signal.

Since the eigenvector Q needs to explicitly calculate the eigendecomposition of the graph Laplacian matrix, the time complexity is $O(N^3)$. Even if the feature value can be calculated in advance, it requires $O(N^2)$ time complexity, which means that it is unrealistic to extend this method to large graphs. On the other hand, because the filter depends on the eigenbasis Q, for graphs of different structures and sizes, the convolutional layer cannot be unified due to the change of dimensionality, and the parameters cannot be shared, so the spectral method cannot be applied to graphs of different structures and sizes.

To solve the problem of a spectral CNN's high computational complexity and the filter not being local, Defferrard et al. [9] proposed ChebNet, which uses a polynomial filter of the form

$$\Theta(\Lambda) = \sum_{k=0}^{k} \theta_k \Lambda^k \tag{3}$$

where $\theta_0, \cdots, \theta_k$ are learnable parameters, and k is the order of the polynomial. Λ is the eigenvalue of the graph Laplacian matrix L. The K-polynomial filter realizes strictly local localization by integrating the node features in the k-hop neighborhood [5] so that the number of learnable parameters decreases to $O(K) = O(1)$.

In order to further reduce the computational complexity, Chebyshev polynomials are used to approximate the calculation of the spectral convolution. By using the recursive relationship of Chebyshev polynomials $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ and $T_0(x) = 1$, $T_1(x) = x$, (3) is rewritten as follows:

$$\Theta(\Lambda) = \sum_{k=0}^{k} \theta_k T_k\left(\widetilde{\Lambda}\right) \tag{4}$$

where $\widetilde{\Lambda} = \frac{2\Lambda}{\lambda_{max}} - I$ is the rescaled eigenvalue, which is in the range of [-1, 1]. λ_{max} is the largest eigenvalue, $I \epsilon R^{N \times N}$ is the identity matrix, and $T_k(x)$ is a Chebyshev polynomial of order k. Equation (4) is essentially a K-order polynomial of the graph Laplacian, so it is K-range, that is, it is only affected by the K-order neighborhood of the node, so the computational complexity of (4) becomes $O(|\varepsilon|)$, where $|\varepsilon|$ is the number of edges. Combining (4) and (2), the convolutional layer of the ChebNet can be expressed as

$$u_j^{l+1} = \rho\left(\sum_{i=1}^{f_l} \sum_{k=0}^{K-1} \Theta_{i,j}^l (k+1) T_k\left(\widetilde{\Lambda}\right) u_i^l\right) j = 1, \cdots, f_{l+1} \tag{5}$$

where $\Theta_{i,j}^l$ is the k-dimensional parameter vector of the ith column of the input feature map and the jth column of the output feature map, which can be learned filters.

ChebNet and its extension combine spectral convolution with spatial structure; spectral

convolution is equivalent to spatial convolution when the spectral convolution function is polynomial or first order. For example, the GCN proposed by Kipf and Welling [7] further simplifies the filter by only intercepting the first-order Chebyshev polynomial, which is not only spectral convolution but also spatial convolution.

There are also some spectrum methods that do not use Chebyshev polynomial expansion. For example, CayleyNet [10] defines graph convolution with a complex coefficient Cayley polynomial and Cayley transform:

$$\Theta(\Lambda) = \theta_0 + 2Re\{\sum_{k=1}^{K} \theta_k(\theta_h\Lambda - iI)^k(\theta_h\Lambda + iI)^k\} \tag{6}$$

where $i = \sqrt{-1}$ is the imaginary unit, and θ_h is another spectrum scaling parameter. $Re(\cdot)$ denotes a real-valued function with a complex coefficient. Because of the nonlinear characteristic of the Cayley transform, we can adjust θ_h to achieve the detection of the concerned frequency bands to obtain better results.

In addition, Levie et al. [11] conducted research on the transferability of spectral GCNs, proving that spectral GCNs always generalize to unseen signals on graphs. They overturned a misconception that using a spectral GCN is inappropriate when the data consist of many different graphs and many different signals on these graphs.

1.3. Spatial convolution of GCNs

Spatial convolution determines graph convolution by aggregating neighbor feature information in the vertex domain.

MoNet, proposed by Monti et al. [12], is a general GCN framework that was created by designing a universal patch operator to integrate signals about nodes. For the neighborhood nodes $y \epsilon N(x)$ of node x on a graph, a d-dimensional pseudo coordinate $u(x, y)$ is defined to associate with it and fed into J learnable kernel functions $(\omega_1(u), \cdots, \omega_J(u))$. Therefore, the patch operator can be expressed as follows:

$$D_j(x)f = \sum_{y \in N(x)} \omega_j\big(u(x,y)\big)f(y),\ j = 1, \cdots, J \tag{7}$$

where $D_j(x)f$ denotes a patch on the manifold, and $f(y)$ is the signal value at node y. The graph convolution of the spatial domain based on the patch operator is defined as follows:

$$(f *_G g)(x) = \sum_{l=1}^{J} g_j D_j(x)f. \tag{8}$$

Among them, $*_G$ is graph convolution operation, and g_j is the convolution kernel. By properly selecting $u(x, y)$ and kernel function $\omega_J(u)$, many existing graph convolution models can be regarded as a special case of MoNet.

SplineCNN [13] also designs a universe patch operator to integrate local information but uses different convolution kernels based on the B-spline function.

GraphSAGE [14] is a representation learning method based on aggregate functions proposed by Hamilton et al. A fixed number of neighbors are uniformly sampled to provide minibatch variants. Multiple aggregate functions are used. The selection of aggregate functions includes average aggregator, LSTM aggregator and pooling aggregator. The GraphSAGE using the average aggregator is similar to the GCN model [7].

WGCN [15] distinguishes the in-neighbor and out-neighbor weights of nodes according to the local topology of nodes, which captures the structural information of nodes. Nodes are embedded into the latent space, and higher-order dependencies and latent geometric relationships of nodes are

obtained to learn nodes' representations.

Some studies [16–19] have shown that spatial graph convolutional networks are vulnerable to adversarial attacks, but it is unclear why the attacks succeed. Liang et al. [20] confirmed that it is non-robust aggregation functions that lead to the structural vulnerability of GCNs. The spatial graph convolution aggregation scheme, i.e., weighted mean, has a low breakdown point, and its output can be changed significantly by injecting a single edge. Then, Liang et al. [20] proposed trimmed mean and median aggregation functions with high breakdown points to improve GCNs' robustness against structural attacks.

2. Graph convolutional neural network models

2.1. Basic models based on graph convolutional neural networks

2.1.1. Graph convolutional neural network

The graph convolutional neural network (GCN) proposed by Kipf et al. [7] was applied to the semi-supervised classification problem of graphs. For a given undirected graph $G = (V, \xi, X)$, where V is the vertex set, the number of vertices is $|V| = n$. ξ is the edge set. $X = \{x_1, x_2, \cdots, x_n\}^T \in R^{n\times c}$ is the feature matrix, and $x_i \in R^c$ represents the c-dimensional feature vector of node i. The goal of the GCN model is to use a small part of the marked nodes, combined with the graph structure, to predict the marked scores of the remaining unmarked nodes for marking.

This model uses a first-order Chebyshev polynomial in each GCN layer and only uses the first two terms of the Chebyshev polynomial to approximate the convolution on the graph to construct the transfer rule of the graph convolutional network:

$$H^{(l+1)} = \sigma\left(\grave{D}^{\frac{-1}{2}}\grave{A}\grave{D}^{\frac{-1}{2}}H^{(l)}W^{(l)}\right) \tag{9}$$

where $\grave{A} = A + I_N$ is the adjacency matrix of the undirected graph G with a self-loop, $\grave{D}_{ii} = \sum_j \grave{A}_{ij}$ is the diagonal degree matrix, $H^{(l)}$ refers to the hidden representation in the lth layer, $H^{(0)} = X$, $W^{(l)}$ is the trainable weight matrix of a specific layer, and $\sigma = (\cdot)$ is the activation function. The multilayer GCN structure will have multilayer propagation, which can realize feature propagation between multilevel adjacent nodes.

The GCN model uses a two-layer GCN structure and applies a *softmax* classifier on the output features. According to the transfer rule of the network, the specific form of forwarding calculation for semi-supervised node classification on a graph with a symmetric adjacency matrix A is

$$Z = f(X, A) = softmax\left(\grave{A}ReLU\left(\grave{A}XW^{(0)}\right)W^{(1)}\right) \tag{10}$$

where $\grave{A} = \grave{D}^{\frac{-1}{2}}\grave{A}\grave{D}^{\frac{-1}{2}}$, and $W^{(0)} \in R^{C\times H}$ and $W^{(1)} \in R^{H\times F}$ represent the weight matrices from the input layer to the hidden layer and the hidden layer to the output layer, respectively. The hidden layer has H features. C is the number of input channels, and F is the number of feature maps of the output layer. The first layer of the activation function is the linear rectification function, and the second layer is the *softmax* function. The *softmax* function form is $softmax(x_i) = \frac{1}{Z}exp(x_i),\ Z = \sum_i exp(x_i)$, and *softmax* is suitable for every row. X is the input signal, a sample with C-dimensional features. Each sample output has a corresponding F-dimensional vector, where component f represents the probability of the sample

being classified into class *f*.

The graph convolutional layer of the GCN model is a special Laplacian smoothing, mixing nodes and their domain characteristics. The smoothing operation makes the features of neighboring nodes similar, which greatly simplifies the classification task, but the multilayer graph convolution will make the output feature excessively smooth, and the nodes of different clusters cannot be effectively distinguished. The shallow GCN model also has limitations: Many additional labeled nodes are required for verification, and it is affected by the localized characteristics of the convolution filter.

In addition, the GCN model has disadvantages such as poor scalability, inability to handle large-scale graphs and inability to handle directed graphs. The specific improvement measures will be explained in the Research Progress part of Section 3. Table 1 summarizes the advantages and disadvantages of several classic GCN models.

Table 1. An overview of the advantages and disadvantages of the classic GCN models.

Type of model	Description	Advantages	Disadvantages
GCN	For the first time, the convolution operation in image processing is applied to graph-structured data processing	·Non-Euclidean structured data ·Capture global information of graphs ·Well characterized node features and edge features	·Poor flexibility, poor scalability, limited to undirected graphs and shallow layers ·GCN training needs to know the structural information of the entire graph
GraphSAGE	Improve GCN scalability and training methods	·Overcome the limitations of memory and video memory during GCNs training ·Able to handle large scale graphs ·The parameters of the aggregator and weight matrix are shared for all nodes	·Unable to process weighted graph ·Neighbor sampling will lead to large gradient variance during backpropagation ·The limited number of samples will lead to the loss of important local information of some nodes
GAT(Graph attention network)	Add attention mechanism to give importance to edges between nodes	·High computational efficiency ·Attention mechanism ·No need to know the whole graph structure ·Can be used for transductive learning and inductive learning	·Prone to oversmoothing ·Poor training mode ·Large number of parameters

2.1.2. Graph sampling and aggregation

GraphSAGE [14] is an inductive learning framework. In the specific implementation, it only retains edges from training samples to training samples during training and includes two steps: Sample and Aggregate. Sample refers to how to sample the number of neighbors, and Aggregate refers to aggregating the embeddings of neighboring nodes to update their own embedding information. Figure 1 shows the process of GraphSAGE generating a target node (red) embedding and predicting for downstream tasks.

The first step is to sample the neighbors. In the second step, the sampled neighbor embedding is passed to the node, and an aggregation function is used to aggregate the neighbor information to update the node's embedding. The third step is to predict the label of the node according to the updated embedding.

2.1.3. Graph attention network

Velickovic et al. [21] were inspired by the attention mechanism and proposed an attention-based GAT model for node classification tasks on graph-structured data, following the self-attention mechanism [22]. When each node updates the hidden representation, the attention of its neighbors must be calculated.

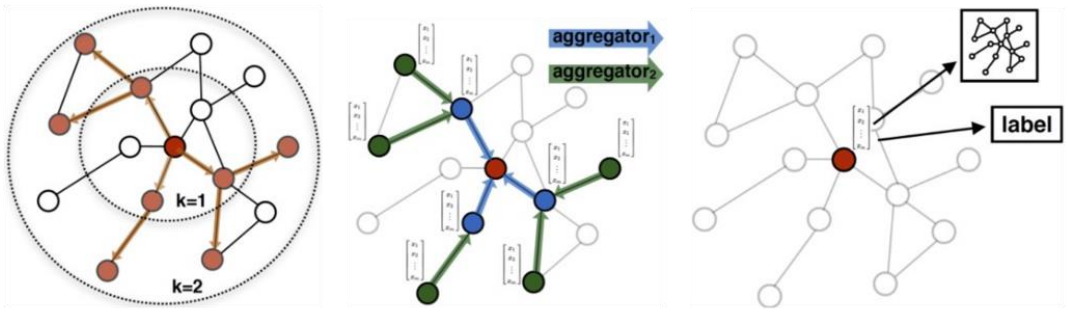


Figure 1. GraphSAGE [14] sampling and aggregation approach.

The model input is the set of node features $h = \{\vec{h}_1, \vec{h}_2, \cdots, \vec{h}_N\}, \vec{h}_i \in R^F$, where N is the number of nodes, and F is the number of features for each node, that is, the length of the feature vector. The output set is $h' = \{\vec{h'}_1, \vec{h'}_2, \cdots, \vec{h'}_N\}, \vec{h'}_i \in R^{F'}$. The number of features of each node becomes F'. The input feature undergoes at least one linear transformation to obtain the output feature, so a weight matrix $W \in R^{F' \times F}$ is defined to be applied to the feature of each node. The function a for calculating the attention coefficient e_{ij} is a single-layer feedforward network, $\vec{a} \in R^{2F'}$ is determined by a weight vector, and *LeakyReLU* is used for processing, so the specific form of the attention mechanism is

$$\alpha_{ij} = softmax_j(e_{ij}) = \frac{\exp\left(LeakyReLU(\vec{a}^T[W\vec{h}_i || W\vec{h}_j])\right)}{\sum_{k \in N_i} \exp\left(LeakyReLU(\vec{a}^T[W\vec{h}_i || W\vec{h}_k])\right)} \tag{11}$$

where $e_{ij} = a(W\vec{h}_i, W\vec{h}_j)$ is the attention coefficient, and $||$ is the concatenation operation. The normalized attention coefficients α_{ij} are used to calculate the linear combination of their corresponding features and used as the final output feature of each node. In addition to the weighted summation, a nonlinear function is also needed, namely,

$$\vec{h'}_i = \sigma\left(\sum_{j \in N_i} \alpha_{ij} W\vec{h}_j\right). \tag{12}$$

To make the model more stable, a multi-head attention mechanism is also proposed, that is, not only using a function a to calculate the attention coefficient but also using a set of k functions, and each function calculates a set of attention coefficients and a set of weighted summations of the coefficient α_{ij}. In each convolution layer, k attention mechanisms work independently, calculate their results separately and connect them to obtain the convolution result:

$$\vec{h'}_i = \mathop{\Big\|}_{k=1}^{K} \sigma\left(\sum_{j \in N_i} a_{ij}^k W^k \vec{h}_j\right) \tag{13}$$

where a_{ij}^k is calculated using the attention coefficient calculated by the kth function a^k. The whole process is shown in Figure 2.

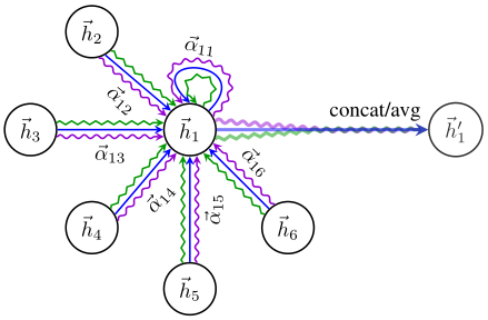


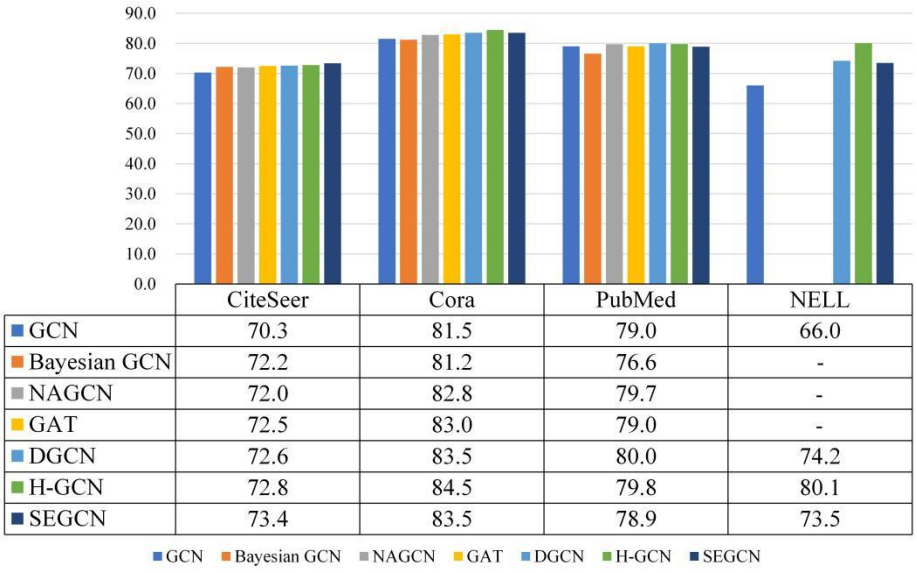
Figure 2. An illustration of multi-head attention (with K=3 heads) by node 1 in its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or averaged to obtain $\vec{h}'_1$.

The characteristics and advantages of the GAT model are as follows: (1) The calculation is efficient, with parallel computing of the attention of each node and its neighbor nodes on all edges, and the output features on all nodes can be calculated in parallel. (2) It can be applied to graph nodes of different degrees by assigning different weights to nodes of the same neighborhood. The attention mechanism is shared for all edges, without matrix operations and without prior knowledge of the graph structure. This solves the problem that the GCN model must be based on the corresponding graph structure to learn the Laplacian matrix. (3) It can be directly applied to the inductive learning problem. There is no problem that there is always an order in the neighborhood node of each node [14]. The model can be applied to completely unknown graph tasks. This is to make up for the shortcoming of the GCN model that a model trained on one graph structure cannot be applied to other graph structures. (4) GAT can be redefined as a special case of MoNet, that is, select the pseudo coordinate function $u(x,y) = f(x)||f(y)$, where $f(x)$ represents the feature (potentially MLP-transformed) of node x, $||$ is concatenation, and set the kernel function $\omega_J(u) = softmax(MLP(u))$ ($softmax$ is executed on the entire neighborhood of the node). MoNet's patch operator will be similar to GAT. Here, the GAT model uses node features for similarity calculation, not the structural attributes of nodes.

2.1.4. Other models

Zhuang et al. [23] proposed a dual graph convolutional network (DGCN) for graph-based semisupervised learning that not only considers local consistency but also encodes global consistency using PPMI. Hu et al. [24] proposed a deep hierarchical graph convolutional network model (H-GCN) to address the problem that traditional models based on neighborhood aggregation are limited to shallow layers and have difficulty obtaining global information. Bayesian GCN [25] and Self-Ensembling GCN (SEGCN) [26] can make full use of unlabeled nodes. The neighborhood adaptive graph convolutional network (NAGCN) [27] can effectively learn the representation of each node. Hyperbolic graph convolutional networks (HGCNs) [28] can learn the inductive node representation of hierarchical or scale-free graph structure data and demonstrate powerful representation ability to model graphs with hierarchical structures. HGCNs work on hyperbolic manifolds with tangent spaces, which are just local approximations of a manifold. Dai et al. [29] proposed a hyperbolic-to-hyperbolic graph convolutional network (H2H-GCN) that directly works on hyperbolic manifolds. H2H-GCN [29] avoids the distortion caused by the tangent space approximation and keeps the global hyperbolic structure.

Figure 3 summarizes the performances of different GCN models on different graph datasets. CiteSeer, Cora and PubMed are citation network datasets, where each sample point is a scientific paper, and each paper cites at least one other paper or is cited by another paper, treating citation links between papers as (undirected) edges. NELL is a knowledge graph dataset with 55,864 relational nodes and 9891 entity nodes.



	CiteSeer	Cora	PubMed	NELL
GCN	70.3	81.5	79.0	66.0
Bayesian GCN	72.2	81.2	76.6	-
NAGCN	72.0	82.8	79.7	-
GAT	72.5	83.0	79.0	-
DGCN	72.6	83.5	80.0	74.2
H-GCN	72.8	84.5	79.8	80.1
SEGCN	73.4	83.5	78.9	73.5

Figure 3. The performance of different GCN models on different graph datasets.

The results in Figure 3 show that for the same graph data, there are some differences in the node classification accuracy of different GCN models, but the differences are not significant. These results confirm the power of GCN models for graph data processing tasks.

2.2. Graph pooling method

The tasks on the graph can be roughly divided into two categories: node-level tasks and graph-level tasks. Node-level tasks, including node classification, link prediction and node recommendation, are related to nodes in the graph; and graph-level tasks, including graph classification and graph generation, are related to the entire graph. Graph convolutional neural networks generally add pooling operations in graph-level tasks and use pooling operators to reflect the hierarchical structure of the network.

There are few existing graph pooling methods, which are mainly divided into three types: topology-based pooling, global pooling and hierarchical pooling. The topology-based pooling method only considers the topological structure of the graph, and representative works include ChebNet [9] and the hybrid model proposed by Rhee et al. [30]. The global pooling method only considers the characteristics of the graph, including representative works such as a general framework for graph classification (taking graph neural networks (GNNs) as a message passing scheme, using the Set2Set method to obtain the representation of the entire graph) proposed by Gilmer et al. [31] and SortPool (according to the structure of the graph, sorting the embeddings of the nodes) proposed by Zhang et al. [32]. The main motivation of the hierarchical pooling method is to build a model that can learn the feature-based or topology-

based node assignment in each layer and allow the neural network to end-to-end learn the assignment matrix to obtain a scaled-down graph. Typical hierarchical pooling methods include DIFFPOOL [33], EigenPooling [34] and SAGPool [35]. The three of these common and typical hierarchical pooling methods and their connections are described below.

DIFFPOOL [33] is a differentiable graph pool module proposed by combining GNN and the pooling operation, similar to CNN, which can capture the hierarchical information of the graph. The main mechanism of hierarchical pooling DIFFPOOL is to learn differentiable cluster assignments $S^{(l)}$ for nodes based on features or topology in each layer of the deep GNN, map the nodes to a set of clusters and use these clusters as coarse input to the next layer GNN. Figure 4 shows the working mechanism of DIFFPOOL.

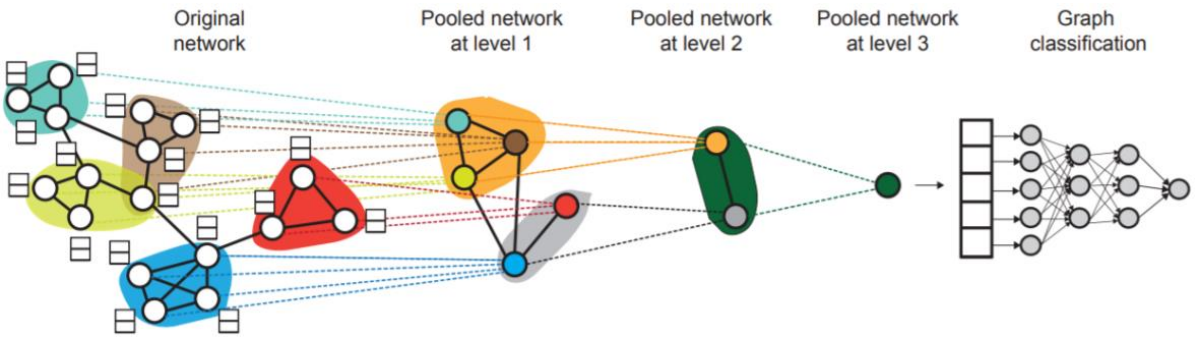


Figure 4. The working mechanism of DIFFPOOL [33]. Run a GNN model at each layer to obtain the embedding of the node. Then, these learned embeddings are used to cluster the nodes together to form a coarsened graph, and another GNN layer is run on the coarsened graph. The whole process is repeated for the L layer, and the final output representation is used to classify the graph.

A GNN model is run in each layer, and the adjacency matrix and features $\left(A^{(l)}, X^{(l)}\right)$ are input to obtain the node embedding $Z^{(l)} = GNN_{l,embed}\left(A^{(l)}, X^{(l)}\right)$ and cluster assignment matrix $S^{(l)} = softmax\left(GNN_{l,pool}\left(A^{(l)}, X^{(l)}\right)\right)$. After the DIFFPOOL differentiable pooling layer, the input of the next layer coarsening graph is obtained, $\left(A^{(l+1)}, X^{(l+1)}\right) = DIFFPOOL\left(A^{(l)}, Z^{(l)}\right)$, where $X^{(l+1)} = {S^{(l)}}^T Z^{(l)} \in R^{n_{l+1} \times d}$, $A^{(l+1)} = {S^{(l)}}^T A^{(l)} S^{(l)} \in R^{n_{l+1} \times n_{l+1}}$.

EigenPooling [34] is based on the idea of DIFFPOOL [33] and combines the pooling layer of the pooling operator based on the graph Fourier transform with the traditional GCN convolutional layer to form a graph neural network framework for graph classification. The pooling operator and the graph coarsening method based on the subgraph jointly constitute the pooling layer, and the features of the entire graph are extracted hierarchically. EigenPooling uses the clustering method to cut the large graph into subgraphs $G^{(k)}$ that have no intersection with each other. The assignment matrix S is determined by spectral clustering, and the feature matrix X_{coar} of the coarsened graph is determined by the graph Fourier transform. Different from the coarsening graph feature matrix and the assignment matrix in DIFFPOOL, both are obtained through a single graph network. Figure 5 shows an example of EigenPooling for graph classification.

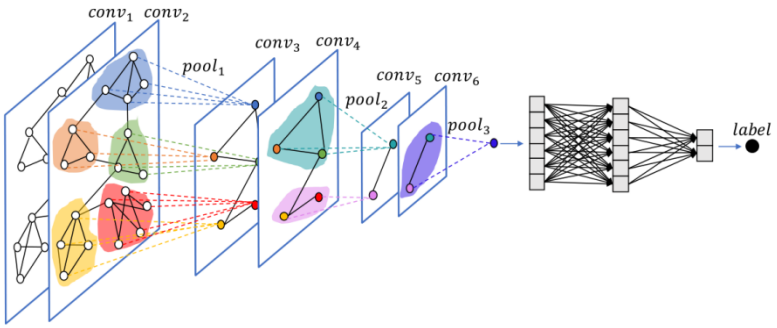


Figure 5. An example of graph classification by EigenPooling [34]. In this example, the graph is coarsened three times to form a single supernode.

The assignment matrix of EigenPooling represents the relationship between node v_i and all subgraphs in graph G and is defined as

$$S[i,k] = 1 if\ and\ only\ if v_i \in \Gamma^{(k)} \tag{14}$$

where $\Gamma^{(k)}$ represents a list of all nodes in graph $G^{(k)}$. After graph G is coarsened, the subgraph becomes a supernode. The adjacency matrix of the coarsened graph G_{coar} is expressed as $A_{coar} = S^T A_{ext} S$, that is, the adjacency matrix representing the edge relationship between super nodes. $A_{ext} = A - A_{int}$ represents the adjacency matrix that only contains the edges between the subgraph and the subgraph. $A^{(k)}$ represents the adjacency matrix of the node connection in subgraph $G^{(k)}$, and A_{int} represents the adjacency matrix that does not contain the edge between the subgraphs.

The feature matrix X of graph G after being downsampled by Θ_l^T is represented as $X_l \in R^{K \times d}$, and the kth row represents the downsampled information of the kth subgraph $G^{(k)}$. Θ_l represents a matrix composed of the eigenvectors of all subgraphs (the eigenvectors of the Laplacian matrix). Concatenate all the downsampled X_l to obtain the final downsampled result $X_{pooled} \in R^{K \times d \cdot N_{max}}$. To simplify the operation, take $H \ll N_{max}$; then, the feature matrix of the coarsening graph is $X_{coar} = [X_0, \cdots, X_H]$.

The number of parameters of DIFFPOOL depends on the number of nodes, and it has quadratic space complexity. Subsequently, Cangea et al. [36] introduced the hierarchical pooling gPool [37] to solve the complexity problem but did not consider the topology of the graph.

To further improve the graph pooling method, a Self-Attention Graph Pooling method (SAGPool) [35] was proposed based on gPool which considers the characteristics and topological structure of the graph at the same time and uses an End2End method to generate hierarchical representation with reasonable time and space complexity. Figure 6 shows the workflow of the SAGPool layer, which analyzes the importance of nodes through self-attention and selects some important nodes to replace the original graph, such as the operation of node classification.

SAGPool uses the graph convolution method to obtain the self-attention score. If the convolution formula of Kipf & Welling [7] is used, the self-attention score $Z \in R^{N \times 1}$ is calculated as

$$Z = \sigma\left(\grave{D}^{\frac{-1}{2}} \grave{A} \grave{D}^{\frac{-1}{2}} X \Theta_{att}\right). \tag{15}$$

$\Theta_{att} \in R^{F \times 1}$ is the only parameter of the SAGPool layer. The parameters of SAGPool are the same, not considering the size of the input graph.

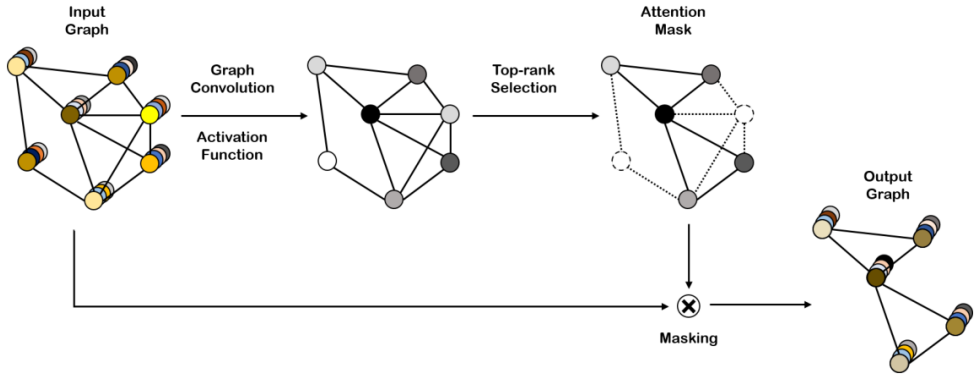


Figure 6. SAGPool layer structure [35].

SAGPool uses the node selection method of gPool; even if the graph is small (few nodes), it still retains part of the nodes in the input graph. The pooling ratio $k \in$ is a hyperparameter that maintains the number of nodes. The first $\lceil kN \rceil$ nodes are selected according to the value of Z:

$$idx = top - rank(Z, \lceil kN \rceil), Z_{mask} = Z_{idx}. \tag{16}$$

$top - rank(Z, \lceil kN \rceil)$ indicates that the Z index is used for sorting, and the first kN nodes are selected. idx represents the get index operation, and Z_{mask} is the feature attention mask.

The input graph is processed by the masking operation, as shown in Eq 17:

$$X' = X_{idx}, X_{out} = X' \odot Z_{mask}, A_{out} = A_{idx,idx} \tag{17}$$

where X_{idx} is the feature matrix of the index arranged in rows (each row represents the feature vector of a node), and $\odot$ is the broadcast elementwise product. X_{out} is the new feature matrix, and A_{out} is the new adjacency matrix. $A_{idx,idx}$ is the rowwise and columnwise indexed adjacency matrix.

3. Research progress of graph convolutional neural networks

Spectral domain GCNs are very powerful, but they still have some shortcomings: 1) Their scalability is poor. The training of GCNs needs to know the adjacency matrix of all training nodes and test nodes, so it is transductive and cannot handle large-scale graphs. 2) They are confined to undirected graphs because the Laplacian matrix used by GCNs requires that eigendecomposition must meet the condition of matrix symmetry, and the premise of spectral convolution is that the graph must be an undirected graph. 3) They are confined to shallow layers, because each layer of GCNs is a special Laplace smoothing, and adding a multilayer GCN structure will make the node attributes of the same connected component too similar, thus making the output features too smooth. At present, there are many works devoted to addressing these shortcomings. The spatial domain graph convolution method can compensate for these shortcomings of spectral domain GCNs, but because it saves memory at the expense of time efficiency, the complexity is high. This chapter separately introduces the latest research progress in solving large-scale graph data, directed graphs, multiscale graph tasks, dynamic graphs and relational graphs. Table 2 summarizes GCN-related methods and applications for complex graph tasks.

Table 2. An overview of complex graph task methods.

Complex graph task	Methods	Applications
Large-Scale graph	LGCNs [38], Cluster-GCN [39], LADIES [40], Bi-GCN [41]	Node Classification
Directed graph	MotifNet [42], TAGCN [43], RCNN [44], FDGCN [45]	Node Classification
Multiscale graph	N-GCN [46], MDGCN [47], Liao et al. [48], Luan et al. [49]	Node Classification, Hyperspectral Image Classification
Dynamic graph	Manessi et al. [50], EvolveGCN [51], DGCM [52], DPG [53]	link prediction, edge classification, node classification, Multi-Person Pose Estimation
Relational graph	R-GCN [54], MR-GCN [55], mRGCN [56]	Link Prediction, Entity Classification, Node Classification, Ingredient Recognition

3.1. Large-scale graph data

Gao et al. proposed large-scale learnable graph convolutional networks (LGCNs) [38], which convert graph structure data into one-dimensional grid data and use traditional convolution operations on the graph. A learnable graph convolutional layer (LGCL) was proposed, which automatically selects a fixed number of neighbor nodes for each feature, that is, it sorts the neighbor nodes and takes the top k number of the feature as the k values of such features of the target node. The neighbor nodes are not enough, and 0 is used as a filler so that a matrix of the neighbor information of the node can be obtained. Figure 7 shows the process of LGCL constructing grid data. For each feature, the top k values (the features of neighboring nodes are sorted in descending order) are selected from the neighboring nodes; for example, the selection result of the first feature of the center node (orange) {9, 6, 5, 3} is obtained by taking the top 4 values of {9, 6, 5, 3, 0, 0}. For the (k+1) three-component feature vector obtained from the center node, connect to obtain one-dimensional grid data with k+1 positions and 3 channels. Finally, a 1-D convolutional neural network is used to generate the final feature vector, and a new vector with five features is used as the new feature representation of the central node. To allow the LGCN model to be trained on large-scale graphs, a subgraph training strategy is proposed to reduce the overhead of memory and computing resources.

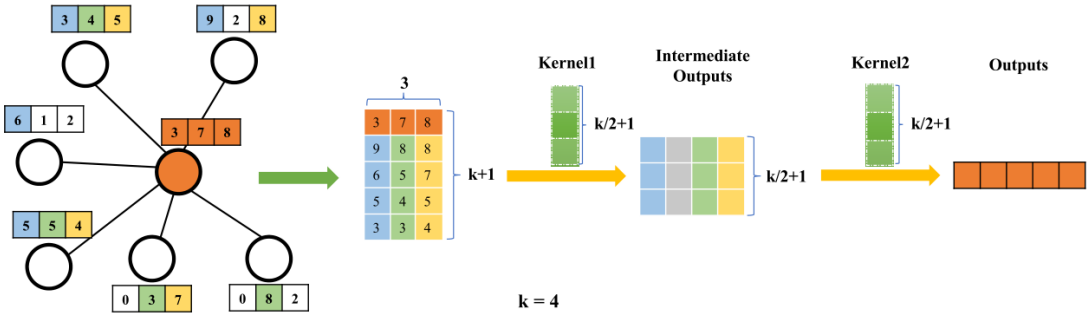


Figure 7. LGCL's process of constructing grid data [38]. The left part represents the process of selecting the top k ($k = 4$) values for each feature from the neighboring nodes. The right part represents a one-dimensional convolutional neural network.

The subgraph training strategy of large-scale graph data draws on the idea of random sampling and cropping patches. Given a graph, first, random sampling is performed for some initial nodes, and then breadth first search (BFS) is used to iteratively expand adjacent nodes into subgraphs. Through

multiple iterations, the higher-order neighbor nodes of the initial node will be added. Figure 8 shows a process of subgraph selection. First, we randomly sample, take three ($N_{init} = 3$) initial nodes (orange) and aim to obtain a subgraph with 15 ($N_s = 15$) nodes. Use breadth-first search (BFS) to iteratively expand neighboring nodes for the subgraph. In the first iteration, 5 blue nodes (not including themselves) among all first-order neighbor nodes of the 3 initial nodes are randomly selected. In the second iteration, 7 green nodes among the second-order neighbor nodes are randomly selected. Thus, a subgraph of 3 + 5 + 7 = 15 nodes is obtained, together with the corresponding adjacency matrix, as the input of LGCNs in the training iteration. With such an operation of randomly cropping subgraphs, large-scale graph data can be solved effectively.

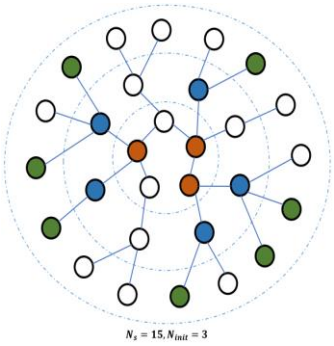


Figure 8. The process of subgraph selection [38].

Chiang et al. proposed a GCN algorithm called Cluster-GCN [39], which is based on a graph clustering structure and SGD training that can train a deep GCN on large-scale graph data. The graph clustering algorithm aims to construct partitions of nodes so that there are more graph links between nodes in the same partition than between nodes in different partitions to better capture the clustering and partition structure. Figure 9 shows the neighborhood spread difference between the traditional graph convolution and the clustering method used by Cluster-GCN. Cluster-GCN can avoid heavy neighborhood search and concentrate on neighbors in each same cluster.

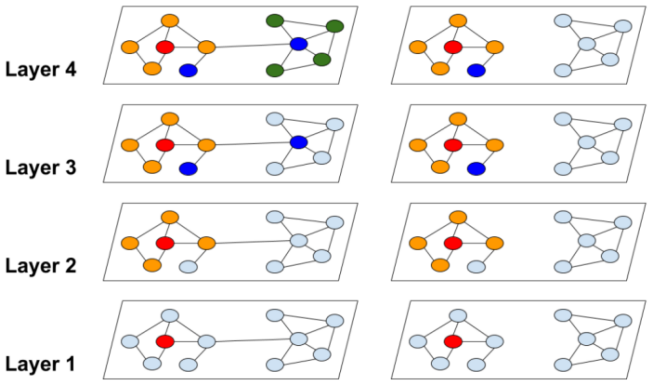


Figure 9. The difference in neighborhood expansion between traditional graph convolution and the clustering method used by Cluster-GCN [39]. The left side is the traditional graph convolution, and the right side is the Cluster-GCN.

The scheme of training a deep GCN on large-scale graph data and having another layer-dependent importance sampling (LADIES) [40] was proposed by Zou et al. LADIES selects its neighbor nodes according to the sampling nodes in the previous layer, constructs a bipartite graph between the two layers and calculates the importance probability. Then, a fixed number of nodes are sampled according to the calculated probability, and the process is performed recursively layer by layer to construct the entire calculation graph. As a new layer correlation sampling scheme, LADIES not only avoids the exponential expansion of the receptive field but also ensures the connectivity of the sampling adjacency matrix. LADIES can also solve the redundant calculation problem of Cluster-GCN.

Wang et al. [41] proposed a binary graph convolutional network (Bi-GCN), which binarizes both the network parameters and input node features. They modified the original matrix multiplication to a binary operation for acceleration. The Bi-GCN's network parameters and memory consumption of input node attributes can be significantly reduced by ~30 times, which can efficiently handle large-scale attributed graphs.

3.2. Directed graph

For directed graph processing, MotifNet [42] utilizes local graph motifs and uses motif-induced adjacencies in deep learning of graphs. Topology adaptive graph convolutional networks (TAGCN) [43], based on graph signal processing theory [3,4], process graph signals defined in the vertex domain and classify the entire graph signal. RCNNs [44] that deal with complex networks can also be used for directed networks.

In addition, in 2020, Li et al. proposed a fast directed graph convolutional network (FDGCN) [45], constructing a spectrum-based GCN of directed graphs. The spectrum-based GCN can make good use of the structural information of the graph. FDGCN is a scalable graph convolutional network based on the Laplacian of the directed graph. The convolution operator (fast local filter) has a linear relationship with the edges of the graph, so the directed graph can be directly calculated.

FDGCN combines the first-order Chebyshev polynomial approximation of the directed graph Laplacian spectral filter and the approximation of the fixed Perron vector to export the directed graph convolution operator. Based on the exported fast local convolution operator, a two-layer FDGCN model for semi-supervised classification was designed, as shown in Figure 10. FDGCN can take GCN as a special case. Combined with the graph generation model MMSBM [57] to improve FDGCN.

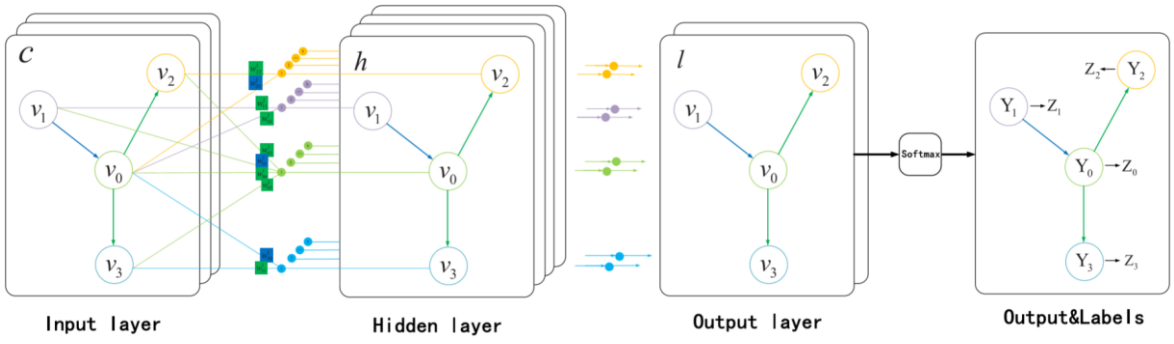


Figure 10. Fast directed graph convolutional network [45]. FDGCN consists of two layers: the hidden layer and the output layer. The hidden layer has h units, and each unit has an n-dimensional vector, which is calculated from the input features $X \in R^{n \times c}$. The output layer has l units, and l represents the number of categories.

3.3. Multiscale graph task

For multiscale graphs with different step lengths, Sami et al. proposed a GCN network model, N-GCN [47], which combines graph convolutional networks (GCNs) [7] with a random walk strategy. Figure 11 shows the structure of the N-GCN model. Multiple instantiations are performed for the GCN model on the node pairs found at different distances of the random walk, and the power of the adjacency matrix is provided for each instantiation. Each instantiation runs on a different scale of the graph, performs end-to-end training, directly learns near or far neighbor information and then connects the output of instances of all optimized classification targets to a classification subnetwork. Figure 11 shows the process: Calculate the K power of $\hat{A}$, send each power and X together into r GCNs, the output of all $K \times r$ GCNs are connected in series according to the column and fed into the fully connected layer, and finally output the row prediction $N \times C$ of the known label. Calculate the cross-entropy error between row predictions to update the parameters of the classification subnetwork and all GCNs.

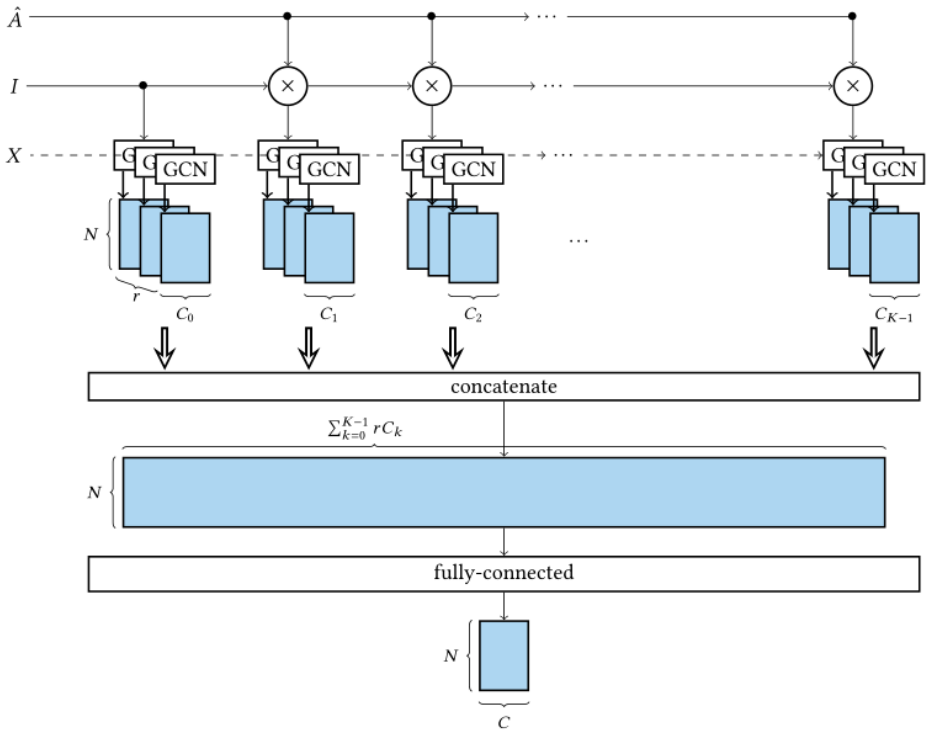


Figure 11. N-GCN model structure [47]. $\hat{A}$ is the normalized adjacency matrix, I is the identity matrix, X is the node feature matrix, $\otimes$ is the matrix multiplication operator, and C is each node output channel, that is, the size of the label space.

Later, Wan et al. proposed a multiscale dynamic GCN (MDGCN) [47] for the hyperspectral image classification problem, as shown in Figure 12. Using the multiscale information of the hyperspectral image, the dynamic map gradually refined in the convolution process is used to establish the multi-input map of different neighborhood scales, which can make full use of the spectral-spatial feature information of multiscale.

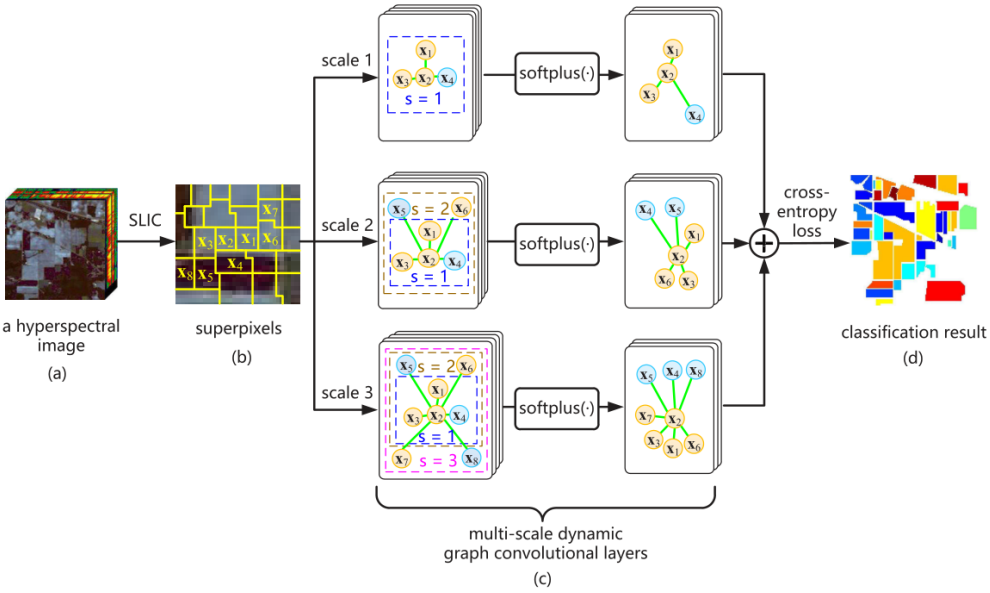


Figure 12. MDGCN model architecture [47]. (a) represents the original hyperspectral image. (b) represents the use of the SLIC algorithm [58] to segment superpixels. (c) represents the multiscale dynamic graph convolutional network layers. Circles represent nodes, and different colors represent different land-cover types. The green line represents the edge, and the edge weight is gradually updated with the convolution operation on the node, thereby dynamically refining the graph. There are three scales in the graph, and $softplus(\cdot)$ represents the activation function. (d) The classification result of the multiscale output, using cross-entropy loss to compensate for the labeling error between the output and the seed superpixels.

In addition, LanczosNet was proposed by Liao et al. [48] and Luan et al. [49], offering two GCN architectures that use different scale information. LanczosNet uses the Lanczos algorithm to construct the low-rank approximation of the graph Laplacian using the tridiagonal decomposition of the Lanczos algorithm, effectively collecting multiscale information through the fast approximate calculation of the matrix power. A learnable spectral filter was designed to effectively improve model capacity. Luan et al. [49] stated that any convolution with a well-defined analytical spectrogram can be written as a block Krylov matrix and a special form of learnable parameter matrix. Therefore, spectrogram convolution and deep GCN are extended to the block Krylov subspace form, and two architectures are proposed: Snowball and truncated Krylov, both of which connect multiscale information in different ways. Both can be extended to deeper structures. Under certain conditions, the equivalence of the two architectures can be achieved.

3.4. Dynamic graph

For dynamic graphs, Manessi et al. [50] combined long short-term memory networks (LSTMs) and graph convolutional networks (GCNs) [7] to learn long short-term dependencies and graph structures, which can capture time information and properly manage structured data. LSTM is a special recurrent neural network that can improve long short-term dependent learning. GCN can effectively process graph structure information. Aldo et al. [51] combined GCN and a recurrent neural network to

propose EvolveGCN, which uses GCN to learn node representation and GRU or LSTM to learn GCN parameters to capture the dynamics of dynamic graphs. EvolveGCN can cope with frequently changing graphs and does not need to know all the changes of the nodes in the graph in advance. When the amount of node feature information in the dataset is not rich enough, the graph data structure plays a more important role, and the effect of using LSTM will be better; otherwise, the effect of using GRU will be better. In addition, Qiu et al. [52] proposed a dynamic graph convolutional module (DGCM) for 2D multiperson pose estimation to solve large changes in human pose, and Song et al. [53] proposed a dynamic probabilistic graph convolution (DPG) model for facial action unit intensity estimation.

3.5. Relational graph

Michael et al. [54] used GCNs to model relational data (graph data with plenty of relations) and proposed the relational graph convolutional network (R-GCN), which was mainly used to solve two completion tasks of the knowledge base: link prediction and entity classification. Huang et al. [55] proposed a multirelational graph convolutional network (MR-GCN) to solve the semi-supervised node classification task of multirelational graphs using the eigendecomposition of Laplacian tensors instead of the discrete Fourier transform to perform graph convolution in the spectral domain. The spectral domain can be any unitary transform. Chen et al. [56] proposed a relational graph convolutional network (mRGCN) to study the problem of zero-shot food ingredient recognition. mRGCN encodes a multirelationship graph (including various relations between ingredient hierarchy, ingredient cooccurrence and ingredient attribute), which is convenient for better transfer of knowledge learned from familiar classes to unfamiliar classes in zero-training sample learning to recognize unseen ingredients.

4. Applications of graph convolutional neural networks

4.1. Computer vision

Graph convolutional neural networks for action recognition are a research hotspot in the computer vision field, mainly by using skeletons as edges and joints as vertices so that the human skeleton can be operated as a graph. Figure 13 shows the spatial temporal graph of the skeleton sequence. Zheng et al. used the OpenPose method to extract human skeleton information from video and construct a skeleton spatial temporal graph, and they proposed a spatial and temporal graph convolutional network (ST-GCN) [59] to extract skeleton features from continuous video frames for video classification. Tian et al. [60] introduced an attention mechanism and cooccurrence feature learning to improve ST-GCN, forming a multitask framework that includes the attention branch, cooccurrence feature learning branch and ST-GCN. To capture the high-level interactive features between the five parts of the human body and distinguish the subtle differences between some confusing actions, Chen et al. proposed a graph convolutional network [61] with a structure-based graph pooling (SGP) scheme and a joint-wise channel attention (JCA) module. SGP integrates the human skeleton graph based on prior knowledge of the human body type. The JCA module selectively focuses on distinctive joints and assigns different degrees of attention to different channels to improve the network's classification performance for confusing behavior. Dong et al. proposed action recognition based on the fusion of graph convolutional networks with high-order features [62], using high-order spatial features of skeletal data, the relative distances between 3D joints, and high-order temporal features, velocity features and acceleration

features. They used deep learning to extract the relative distances between 3D joints, learning velocity and acceleration from deep networks. Chen et al. [63] proposed a multiscale spatiotemporal graph convolutional network (MST-GCN) for action recognition by learning effective action representations. MST-GCN is composed of a stack of basic blocks that combine a multiscale spatial graph convolution (MS-GC) module and a multiscale temporal graph convolution (MT-GC) module. The MS-GC module captures local joint connections and nonlocal joint relationships for spatial modeling, and the MT-GC module effectively expands the temporal receptive field of long-term temporal dynamics. Table 3 presents the results of the above five methods on the NTU RGB+D dataset. The NTU RGB+D dataset includes two evaluation metrics: cross-subject (CS) and cross-view (CV). In the cross-subject evaluation, there are 40,320 and 16,560 clips for training and evaluation, respectively. In the cross-view evaluation, there are 37,920 and 18,960 clips for training and evaluation, respectively. Table 3 summarizes the top-1 classification accuracy on two evaluation metrics.

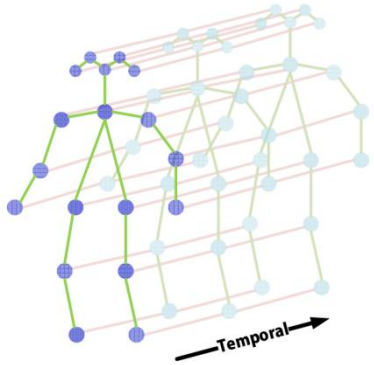


Figure 13. Multi-frame skeleton graph [59]. The blue nodes represent the human joints, the green lines represent the human bones (the edges where the joints are connected), and the pink lines represent the connections of the same joint between successive frames.

Table 3. Comparison of the top-1 accuracy values with five action recognition methods on the NTU RGB+D dataset.

Method	Cross-subject (%)	Cross-view (%)
ST-GCN [59]	81.5	88.3
Tian et al. [60]	86.8	92.1
Chen et al. [61]	86.1	93.1
Dong et al. [62]	90.5	95.8
MSF-GCN [63]	91.5	96.6

Human pose estimation, as a basic task in many computer vision applications, involves locating key points of the human body in still images. The pose graph convolutional network (PGCN) [64] establishes a directed graph between key points of the human body according to the structure of the human body, using an attention mechanism to focus on the structured relationship between the key points. PGCN is trained to map the graph into a set of key point representations of structural perception, which can encode both the body structure and the appearance information of specific key points. The global reasoning graph convolutional network (GRR-GCN) [65] projects features of the original space to non-Euclidean space to form a fully connected graph composed of vertices, uses a graph convolutional network to perform global relational reasoning on the fully connected graph and globally learns the relationships between still image regions. Table 4 presents the results of PGCN and GRR-

GCN on the MPII dataset. The MPII Human Pose datasets use the percentage of correct keypoints (PCK) metric for evaluation. PCK reports the percentage of keypoints that fall into a normalized distance of the ground truth. A space-time separable graph convolutional network (STS-GCN) for pose forecasting was proposed by Theodoros et al. [66]. STS-GCN is the first network to model human pose dynamics with only one graph convolutional network, which is the first to factorize the graph adjacency matrix, rather than depthwise. Zou et al. [67] proposed a new modulated GCN for 3D human pose estimation. Modulated GCN can disentangle the feature transformations of different nodes while keeping the model size small. In addition, this method can model edges other than the human skeleton, which significantly reduces the estimation error. Table 5 presents the results of STS-GCN and modulated GCN on the Human3.6M dataset. The MPJPE error metric was adopted, which measures the average Euclidean distance (in millimeters) between the ground truth and the prediction after aligning the root joint (the hip joint). In Table 5, STS-GCN_short-term (400) and STS-GCN_long-term (880) represent the MPJPE errors (in millimeters) of short-term (10 frames, 400 ms) and long-term (22 frames, 880 ms) predictions of 3D joint positions on the Human3.6M dataset, respectively.

Table 4. Comparisons of PCKh@0.5 scores of PGCN and GRR-GCN on the MPII test set.

Method	Head	Sho.	Elb.	Wri.	Hip	Knee	Ank.	Mean
PGCN [64]	98.0	96.9	92.7	89.0	91.8	89.4	86.1	92.4
GRR-GCN [65]	97.7	96.3	92.0	87.7	90.3	88.1	84.1	91.3

Table 5. Quantitative comparisons of STS-GCN and modulated GCN on Human3.6M under MPJPE.

Methods	Dire.	Disc.	Eat	Greet	Phone	Photo	Pose
STS-GCN_short-term (400) [66]	34.7	40.2	25.4	49.2	30.9	33.6	45.6
STS-GCN_long-term (880) [66]	64.5	72.3	46.2	85.5	59.3	67.2	94.5
Modulated GCN [67]	45.4	49.2	45.7	49.4	50.4	58.2	47.9
Methods	Purch.	Sit	SitD.	Smoke	Wait	WalkD.	Walk
STS-GCN_short-term (400) [66]	48.7	35.0	47.9	25.8	35.2	59.6	32.9
STS-GCN_long-term (880) [66]	86.2	67.4	86.2	45.4	66.1	96.2	48.0
Modulated GCN [67]	46.0	57.5	63.0	49.7	46.6	52.2	38.9

4.2. Transportation network

Traffic flow prediction is of great significance to traffic management and public safety. Traditional traffic flow prediction methods often ignore the temporal and spatial dependencies of traffic flow. For this reason, a spatiotemporal graph convolutional network [68] was proposed for the time series prediction problem in traffic networks. The spatiotemporal graph convolutional network metro (STGCNmetro) [69] was developed by constructing a deep structure composed of GCNs to obtain the spatiotemporal dependencies of the city-wide metro network and integrating it into three temporal modes (recent, daily and weekly), forming the final predicted value. In contrast to a model that only uses data directly related to traffic flow for GCN training, Zhao et al. proposed a spatiotemporal data fusion (STDF) architecture [70]. Data indirectly related to traffic flow is input into spatial embedding by temporal convolution (SETON), and each feature of time and space dimensions is encoded. The data directly related to the traffic flow are used as the input of the graph convolutional network. Through the fusion module, all features are combined for the final prediction.

Traffic speed prediction is a crucial component of intelligent transportation systems. Ge et al.

proposed a temporal graph convolutional network (GTCN) [71] composed of a spatiotemporal component and an external component to solve the problem of traffic speed prediction. The spatiotemporal component combines k-order spectrogram convolution and extended random convolution to obtain spatiotemporal dependencies, and the external component is used to consider influencing factors such as road structure characteristics and social factors such as the day of the week. The global spatial-temporal graph convolutional network (CSTGCN) [72] consists of three spatiotemporal components with the same structure and one external component. The spatiotemporal component models the recent, daily and weekly spatiotemporal correlations of traffic data in the time dimension and combines the local graph convolution and the global correlation mechanism to realize the local and global spatial correlation in the space dimension. The external component is used to extract factors that affect traffic speed, such as holidays and weather conditions. Table 6 presents the results of GTCN and CSTGCN on the PeMSD4 and PeMSD7 datasets. PeMSD4 and PeMSD7 were collected by the California Department of Transportation's Performance Measurement System (CalTrans PeMS) every 30 seconds. The traffic speed data were aggregated from the raw data into 5-min windows. Three widely used metrics were used to evaluate the performance of GTCN and CSTGCN: mean absolute error (MAE), root mean square error (RMSE) and mean absolute percentage error (MAPE).

Table 6. Comparison of GTCN and CSTGCN for traffic prediction on PeMSD7 and PeMSD4 datasets.

Data	Method	15 min			30 min		
		MAE	RMSE	MAPE (%)	MAE	RMSE	MAPE (%)
PeMSD7	GTCN [71]	1.47	4.18	3.99	2.65	5.76	7.61
	GSTGCN [72]	1.20	2.16	2.66	1.81	3.03	4.58
PeMSD4	GTCN [71]	1.41	2.89	3.18	2.16	3.96	4.70
	GSTGCN [72]	0.73	1.65	1.78	1.31	3.17	2.94

4.3. Biomedicine

In biomedicine, graph convolutional networks can be used in multiple tasks, such as disease-gene association identification, disease prediction, protein interface prediction and drug discovery. GCN-MF [73] is a framework for discovering disease-gene associations by combining graph convolutional networks and matrix factorization. Using GCN extracts the nonlinear relationship between diseases and genes and exploits measured similarities. In addition, the fully connected graph convolutional network FCGCNMDA [74] has been used to predict potential miRNA-disease associations, and GCNDA [75], which is based on the deep learning method fast learning with graph convolutional networks, has been used to predict circRNA-disease associations. GCGCN [76] is a cancer survival prediction method based on graph convolutional networks that combine multigenome data and clinical data. It considers the heterogeneity between different types of data and makes full use of abstract high-level representations of different data sources. Chen et al. [77] proposed a graph convolutional network structure based on a multilayer aggregation mechanism, which can suppress oversmoothness while obtaining deep structural information and increasing the similarities between nodes of the same type. This method achieves better performance in brain analysis and breast cancer prediction. Karthik et al. [78] proposed a fully learning brain surface analysis approach for processing multiple surface-valued data to output topic-based information. This method has the most advanced performance for ADNI stage classification and brain age prediction using cortical surface data.

4.4. Recommender systems

Different from traditional deep learning models, GCNs can simultaneously make use of content information and graph structure, so GCN-based methods can be better applied to recommender systems, where the main difficulty is how to extend GCN-based node embedding into graphs with billions of nodes and tens of billions of edges. Ying et al. proposed a GCN framework PinSage [79] based on a random walk. The scalability of GCNs was improved by adopting dynamic convolution, a producer-consumer architecture was developed, and an efficient MapReduce pipeline was designed. The use of importance pooling and curriculum training was also introduced to greatly improve embedding performance.

Regarding how to improve the recommendation system's ability to mine and analyze user preferences and behaviors, Xia et al. [80] proposed a two-channel hypergraph convolutional network (DHCN) for session recommendation, which innovatively integrates self-supervision into network training and improves the recommendation task by maximizing the information between sessions of network learning. Chen et al. [81] proposed the SGCN method, which explicitly deleted irrelevant neighbors in the message-passing stage of GCNs and reduced the negative impact of noise on the recommendation system to a large extent.

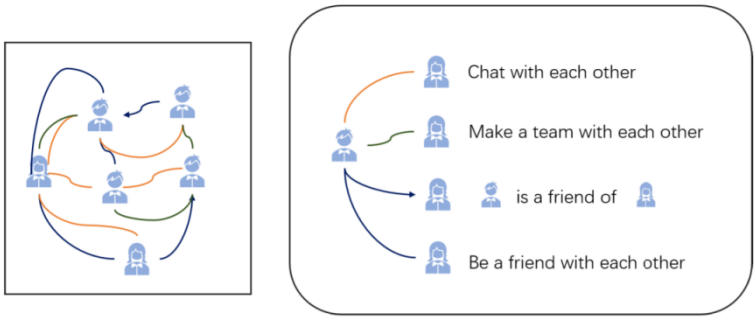


Figure 14. A multisocial graph with multisocial relationships [82], where the friend relationship is a directed graph, and the others are undirected graphs. Chat and team relationships reflect the relationship between friends to a certain extent, increase recommendation interference, meet the diversification of people's needs and avoid the oneness of the recommendation system.

The Multi-Social Graph Convolutional Network (MSGCN) [82] is a friend recommendation model based on multisocial networks that learns the latent characteristics of users, solving the friend recommendation problem with sparse data. Figure 14 shows a multisocial graph. By integrating topology information from multiple social networks to enrich the target user representation, the friend recommendation problem is converted into a personalized recommendation sorting problem using Bayesian theory. The hybrid graph convolutional neural network with multi-head attention for POI recommendation (HGMAP) [83] is a location recommendation framework that recommends points of interest (POIs) that users are interested in but have not yet visited. Two independent GCNs are used to learn social influence and geographical constraints and introduce the multi-head attention encoder to adaptively calculate the calculation score for each check-in to obtain the user's latent preference for nonvisited POIs. The user preference representation and social representation were used to simulate the user influence on POI recommendation.

4.5. Natural language processing

The information extraction tasks of natural language processing include event detection, relationship extraction and entity linking. Event detection aims to identify specific types of instances in text. Thien et al. proposed a neural network model based on graph convolutional network dependency trees and entity mention-guide pooling [84] for event detection. Relation extraction aims to detect the relationship between entities in the text. Attention Guided Graph Convolutional Networks (AGGCNs) [85] use a "soft pruning" strategy, which runs on a full dependency tree; and through the self-attention mechanism, structural information useful for relation extraction is learned in an end-to-end manner. Hong et al. proposed an end-to-end model based on GCN [86] for joint extraction of entities and relations. MSBD [87], based on a character graph convolutional network (CGCN) and multi-head self-attention mechanism (MS), is an improved end-to-end model that realizes the joint extraction of entities and relations. The entity and relation extraction problem is transformed into a text tagging problem, which simplifies the complexity of entity and relationship extraction and improves time efficiency.

In addition, text classification and sentiment analysis are also crucial issues in natural language processing. Text classification using a graph convolutional network is Text GCN [88], which builds a heterogeneous word document graph for a corpus based on word co-occurrence and document word relations and transforms the text classification problem into a node classification problem. Manish et al. [89] proposed NIP-GCN, which introduces the node interaction patterns (NIPs) derived from the linked structure of the graph into the GCN framework and provides prior information for the graph convolutional layers. Combining prior information with an additional pairwise document similarity objective yields superior results compared to Text GCN.

Graph convolutional neural networks have been applied to sentiment analysis. Xiao et al. [90] considered that semantic information, syntactic information and interactive information are crucial to target semantic analysis and proposed an attentional-encoding-based graph convolutional network (AEGCN). AEGCN combines semantic information and syntactic information to predict the emotional polarity of the target. Zhao et al. [91] proposed an aspect-level sentiment classification model (SDGCN) based on a graph convolutional network which can effectively capture the sentiment dependencies between multiple aspects in one sentence. Jiang et al. [92] proposed HDGCN, which enhances multihop graph inference by aggregating the information of direct dependence and long-term dependence into a convolution layer and connecting the premise and hypothesis sentences together to form a long sentence. The DualGCN [93] model of aspect-based sentiment analysis integrates syntactic knowledge and semantic information through SynGCN and SemGCN modules and accurately captures semantic relevance between words by constraining attention scores in the SemGCN module to determine the polarity of emotions of a given aspect in a sentence. Table 7 presents the results of the above four methods on two datasets of SemEval 2014 Task4, which contains reviews in the laptop domain and restaurant domain. Each review (one sentence) contains one or more aspects and their corresponding sentiment polarities, i.e., positive, neutral and negative. In Table 7, BERT represents the BERT model by feeding the sentence-aspect pair and using the representation of [CLS] for prediction. The accuracy and macro average F1 were selected for the evaluation.

Table 7. Test accuracy (%) and macro-F1 score with four sentiment analysis methods on aspect-based sentiment classification.

Methods	Restaurant		Laptop	
	Acc	Macro-F1	Acc	Macro-F1
AEGCN-BERT [90]	82.58	73.40	78.73	74.22
SDGCN-BERT [91]	83.57	76.47	81.35	78.34
HDGCN-BERT [92]	85.89	79.33	79.15	75.48
DualGCN-BERT [93]	87.13	81.16	81.80	78.10

4.6. Social networks

Graph convolutional neural networks are also widely used in social networks. For example, a graph convolutional network based on an autoencoder is used for online financial anti-fraud [94]. By using an autoencoder module to replace the graph convolution matrix, the depth of the graph convolutional network is increased without causing oversmoothing. A graph convolutional network was used to encode social information for the political perspective detection of news media [95], and a graph convolutional network was used to capture social information spread in social networks. A spatially aware graph convolutional network (SAGCN) [96] is used for multipedestrian trajectory prediction in autonomous driving, building an attention graph, where the nodes represent the time information of pedestrians, and the edges represent the pairwise correspondence between pedestrians. Additionally, used for position prediction is a graph convolutional network based on the seq2seq framework [97], which uses the seq2seq framework to generate the hidden state and cell state of the historical trajectories. The graph convolutional network is used to capture spatial correlation and temporal dynamics. Graph convolutional neural networks can also be used to measure the privacy metrics of online social networks [98].

5. Conclusions

This paper presents an overview of graph convolutional networks and their related approaches. First, the advantages and disadvantages of basic models of graph convolutional networks are analyzed. Graph convolutional networks (GCNs) have poor scalability, are limited to shallow networks and undirected graphs and cannot handle complex graph data well. In view of these drawbacks, this paper introduces recent works of graph convolutional networks in complex graph tasks (such as large-scale graphs, directed graphs, multiscale graphs, etc.) and introduces the research progress of graph convolutional networks. In addition, the achievements of graph convolutional networks in some application domains are summarized.

Graph convolutional networks can be applied to many domains, and the effect is remarkable. However, for the processing of complex graph data such as large-scale deep graphs, directed graphs and multiscale dynamic graphs, there is still room for improvement in related work of graph convolutional networks. Therefore, extending graph convolutional networks to complex graph tasks such as large-scale graphs, directed graphs, multiscale graphs and dynamic graphs is the key research direction for graph convolutional networks in the future.

Acknowledgments

This work was supported by the Fundamental Research Funds of Central Universities (No.

2019XKQYMS87).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. Z. Zhang, P. Cui, W. Zhu, Deep Learning on Graphs: A Survey, *IEEE Trans. Knowl. Data Eng.*, **34** (2022), 249–270. https://doi.org/10.1109/TKDE.2020.2981333
2. D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, P. Vandergheynst, The emerging field of signal processing on graphs: Extending high dimensional data analysis to networks and other irregular domains, *IEEE Signal Process. Mag.*, **30** (2013), 83–98. https://doi.org/10.1109/MSP.2012.2235192
3. A. Sandryhaila, J. M. F. Moura, Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure, *IEEE Signal Process. Mag.*, **31** (2014), 80–90. https://doi.org/10.1109/MSP.2014.2329213
4. A. Sandryhaila, J. M. F. Moura, Discrete signal processing on graphs, *IEEE Trans. Signal Process.*, **61** (2013), 1644–1656. https:// doi.org/10.1109/TSP.2013.2238935
5. J. Bruna, W. Zaremba, A. Szlam, Y. Lecun, Spectral networks and locally connected networks on graphs, *arXiv preprint*, (2013), arXiv: 1312.6203. https://doi.org/10.48550/arXiv.1312.6203
6. D. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, eet al., Convolutional networks on graphs for learning molecular fingerprints, *Adv. Neural Inf. Process. Syst.*, **28** (2015), 2224–2232.
7. T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, *arXiv preprint*, (2016), arXiv: 1609.02907.
8. J. Atwood, D. Towsley, Diffusion-convolutional neural networks, *Adv. Neural Inf. Process. Syst.*, **29** (2016), 1993–2001.
9. M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, *Adv. Neural Inf. Process. Syst.*, **29** (2016), 3837–3845.
10. R. Levie, F. Monti, X. Bresson, M. M. Bronstein, CayleyNets: Graph convolutional neural networks with complex rational spectral filters, *IEEE Trans. Signal Process.*, **67** (2019), 97–109. https://doi.org/10.1109/TSP.2018.2879624
11. R. Levie, W. Huang, L. Bucci, M. Bronstein, G. Kutyniok, Transferability of Spectral Graph Convolutional Neural Networks, *J. Mach. Learn. Res.*, **22** (2021), 12462–112520.
12. F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, M. M. Bronstein, Geometric deep learning on graphs and manifolds using mixture model CNNs,. *IEEE Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, 2017, 5425–5434. https://doi.org/10.1109/CVPR.2017.576
13. M. Fey, J. E. Lenssen, F. Weichert, H. Müller, SplineCNN: fast geometric deep learning with continuous b-spline kernels, *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, (2018), 869–877. https://doi.org/10.1109/CVPR.2018.00097
14. W. L. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, *Adv. Neural Inf. Process. Syst.*, **30** (2017), 1024–1034.
15. Y. Zhao, J. Qi, Q. Liu, R. Zhang, WGCN: Graph Convolutional Networks with Weighted Structural Features, in *2021 SIGIR*, (2021), 624–633. https://doi.org/10.1145/3404835.3462834

16. H. Wu, C. Wang, Y. Tyshetskiy, A. Docherty, K. Lu, L. Zhu, Adversarial examples for graph data: Deep insights into attack and defense, *arXiv preprint,* (2019), arXiv: 1903.01610. https://doi.org/10.48550/arXiv.1903.01610
17. D. Zügner, S. Günnemann, Adversarial attacks on graph neural networks via meta learning, *arXiv preprint,* (2019), arXiv: 1902.08412. https://doi.org/10.48550/arXiv.1902.08412
18. K. Xu, H. Chen, S. Liu, P. Chen, T. Weng, M. Hong, et al., Topology attack and defense for graph neural networks: An optimization perspective, in *Proc. Int. Joint Conf. Artif. Intell.*, (2019), 3961–3967. https://doi.org/10.24963/ijcai.2019/550
19. L. Chen, J. Li, J. Peng, A survey of adversarial learning on graph, *arXiv preprint*, (2003), arXiv:2003.05730. https://doi.org/10.48550/arXiv.2003.05730
20. L. Chen, J. Li, J. Peng, Y. Liu, Z. Zheng, C. Yang, Understanding Structural Vulnerability in Graph Convolutional Networks, in *Proc. Int. Joint Conf. Artif. Intell.*, (2021), 2249–2255. https://doi.org/10.24963/ijcai.2021/310
21. P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, *arXiv preprint*, (2017), arXiv:1710.10903. https://doi.org/10.48550/arXiv.1710.10903
22. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit. L. Jones, A. N. Gomez, et al., Attention is all you need, *Adv. Neural Inf. Process. Syst.*, **30** (2017), 5998–6008.
23. C. Zhuang, Q. Ma, Dual Graph Convolutional Networks for Graph-Based Semi-Supervised Classification, in *Proc. Int. Conf. World Wide Web*, (2018), 499–508. https://doi.org/10.1145/3178876.3186116
24. F. Hu, Y. Zhu, S. Wu, L. Wang, T. Tan, Hierarchical Graph Convolutional Networks for Semi-supervised Node Classification, in *Proc. Int. Joint Conf. Artif. Intell.*, (2019), 4532–4539. https://doi.org/10.24963/ijcai.2019/630
25. Y. Zhang, S. Pal, M. Coates, D. Üstebay, Bayesian graph convolutional neural networks for semi-supervised classification, in *Proc. Int. Joint Conf. Artif. Intell.*, **33** (2019), 5829–5836. https://doi.org/10.1609/aaai.v33i01.33015829
26. Y. Luo, R. Ji, T. Guan, J. Yu, P. Liu, Y. Yang, Every node counts: Self-ensembling graph convolutional networks for semi-supervised learning, *Pattern Recognit.*, **106** (2020), 107451. https://doi.org/10.1016/j.patcog.2020.107451
27. P. Gong, L. Ai, Neighborhood Adaptive Graph Convolutional Network for Node Classification, *IEEE Access,* **7** (2019), 170578–170588. https://doi.org/10.1109/ACCESS.2019.2955487
28. I. Chami, Z. Ying, C. Ré, J. Leskovec, Hyperbolic graph convolutional neural networks, in *Proc. Adv. Neural Inf. Process. Syst.*, (2019), 4868–4879.
29. J. Dai, Y. Wu, Z. Gao, Y. Jia, A Hyperbolic-to-Hyperbolic Graph Convolutional Network, in *2021 IEEE/CVF Conf. Computer Vision Pattern Recogn. (CVPR)*, (2021), 154–163. https://doi.org/10.1109/CVPR46437.2021.00022
30. S. Rhee, S. Seo, S. Kim, Hybrid Approach of Relation Network and Localized Graph Convolutional Filtering for Breast Cancer Subtype Classification, in *2018 Int. Joint Conf. Artif. Intell.*, (2018), 3527–3534. https://doi.org/10.24963/ijcai.2018/490
31. J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, G. E. Dahl, Neural Message Passing for Quantum Chemistry, *in 2017 Int. Conf. Machine Learn.*, (2017), 1263–1272.
32. M. Zhang, Z. Cui, M. Neumann, Y. Chen, An End-to-End Deep Learning Architecture for Graph Classification, in *Proc. Artif. Intell.*, (2018), 4438–4445. https://doi.org/10.1609/aaai.v32i1.11782

33. R. Ying, J. You, C. Morris, Hierarchical graph representation learning with differentiable pooling , in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, (2018), 4805–4815.
34. Y. Ma, S. Wang, C. C Aggarwal, J. Tang, Graph convolutional networks with eigenpooling, in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, (2019), 723–731. https://doi.org/10.1145/3292500.3330982
35. J. Lee, I. Lee, J. Kang, Self-attention graph pooling, in *Proc. 36th Int. Conf. Machine Learn.*, (2019), 3734–3743. Available from: http://proceedings.mlr.press/v97/lee19c/lee19c.pdf
36. C. Cangea, P. Velickovic, N. Jovanovic, T. Kipf, P. Lio, Towards sparse hierarchical graph classifiers, in *Proc. Adv. Neural Inf. Process. Syst.*, (2018). https://doi.org/10.48550/arXiv.1811.01287
37. H. Gao, S. Ji, Graph U-Nets, in *Proc. 36th Int. Conf. Machine Learn.,* (2019), 2083–2092. https://doi.org/10.1109/TPAMI.2021.3081010
38. H. Gao, Z. Wang, S. Ji, Large-Scale Learnable Graph Convolutional Networks, in *Proc. Knowl. Disc. Data Min.*, (2018), 1416–1424. https://doi.org/10.1145/3219819.3219947
39. W. Chiang, X. Liu, S. Si, Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks, in *Proc. Knowl. Disc. Data Min.*, (2019), 257–266. https://doi.org/10.1145/3292500.3330925
40. D. Zou, Z. Hu, Y. Wang, S. Jiang, Y. Sun, Q. Gu, Layer-Dependent Importance Sampling for Training Deep and Large Graph Convolutional Networks, in *Proc. Adv. Neural Inf. Process. Syst.*, (2019), 11249–11259.
41. J. Wang, Y. Wang, Z. Yang, Bi-GCN: Binary Graph Convolutional Network, in *2021 IEEE/CVF Conf. Comput. Vision Pattern Recogn. (CVPR)*, (2021), 1561–1570. https://doi.org/10.1109/CVPR46437.2021.00161
42. F. Monti, K. Otness, M. M. Bronstein, MOTIFNET: A Motif-Based Graph Convolutional Network for Directed Graphs, in *Proc. IEEE Data Sci. Workshop*, (2018), 225–228. https://doi.org/10.1109/DSW.2018.8439897
43. J. Du, S. Zhang, G. Wu, J. M. F. Moura, S. Kar, Topology adaptive graph convolutional networks, *arXiv preprint*, (2017), arXiv:1710.10370.
44. E. Yu, Y. Wang, Y. Fu, D. B. Chen, M. Xie, Identifying critical nodes in complex networks via graph convolutional networks, *Knowl.-Based Syst.*, **198** (2020), 105893. https://doi.org/10.1016/j.knosys.2020.105893
45. C. Li, X. Qin, X. Xu, D. Yang, G. Wei, Scalable Graph Convolutional Networks with Fast Localized Spectral Filter for Directed Graphs, *IEEE Access*, **8** (2020), 105634–105644. https://doi.org/10.1109/ACCESS.2020.2999520
46. S. Abu-El-Haija, A. Kapoor, B. Perozzi, J. Lee, N-GCN: Multi-scale Graph Convolution for Semi-supervised Node Classification, in *Proc. Conf. Uncertainty in Artif. Intell.*, (2019), 841–851.
47. S. Wan, C. Gong, P. Zhong, B. Du, L. Zhang, J. Yang, Multiscale Dynamic Graph Convolutional Network for Hyperspectral Image Classification, *IEEE Trans. Geosci. Remote. Sens*., **58** (2020), 3162–3177. https://doi.org/10.1109/TGRS.2019.2949180
48. R. Liao, Z. Zhao, R. Urtasun, R. S. Zemel, LanczosNet: Multi-Scale Deep Graph Convolutional Networks, *arXiv preprint.*, (2019), arXiv:1901.01484. Available from: https://openreview.net/pdf?id=BkedznAqKQ

49. S. Luan, M. Zhao, X. Chang, D. Precup, Break the Ceiling: Stronger Multi-scale Deep Graph Convolutional Networks, in *Proc. Conf. Workshop on Neural Inform. Process. Syst.*, **32** (2019), 10943–10953. Available from: https://proceedings.neurips.cc/paper_files/paper/2019/file/ccdf3864e2fa9089f9eca4fc7a48ea0a-Paper.pdf
50. F. Manessi, A. Rozza, M. Manzo, Dynamic Graph Convolutional Networks, *Pattern Recogn.,* **97** (2020), 107000. https://doi.org/10.1016/j.patcog.2019.107000
51. A. Pareja, G. Domeniconi, J. Chen, T. Ma, T. Suzumura, EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs, in *Proc. Int. Joint Conf. Artif. Intell.*, (2020), 5363–5370. https://doi.org/10.1609/aaai.v34i04.5984
52. Z. Qiu, K. Qiu, J. Fu, D. Fu, DGCN: Dynamic Graph Convolutional Network for Efficient Multi-Person Pose Estimation, in *Proc. Int. Joint Conf. Artif. Intell.*, (2020), 11924–11931. https://doi.org/10.1609/aaai.v34i07.6867
53. T. Song, Z. Cui, Y. Wang, W. Zheng, Q. Ji, Dynamic Probabilistic Graph Convolution for Facial Action Unit Intensity Estimation, in *Proc. IEEE Conf. Comput. Vision Pattern Recogn.*, (2021), 4845–4854. https://doi.org/10.1109/CVPR46437.2021.00481
54. M. S. Schlichtkrull, T. N. Kipf, P. Bloem, R. Berg, I. Titov, M. Welling, Modeling Relational Data with Graph Convolutional Networks, In *The Semantic Web: 15th Int. Conf., ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018,* 593–607. https://doi.org/10.1007/978-3-319-93417-4_38
55. Z. Huang, X. Li, Y. Ye, M. K. Ng, MR-GCN: Multi-Relational Graph Convolutional Networks based on Generalized Tensor Product, in *Proc. Int. Joint Conf. Artif. Intell.*, (2020), 1258–1264. https://doi.org/10.24963/ijcai.2020/175
56. J. Chen, L. Pan, Z. Wei, X. Wang, C. W. Ngo, T. S. Chua, Zero-Shot Ingredient Recognition by Multi-Relational Graph Convolutional Network, in *Proc. Int. Joint Conf. Artif. Intell.*, **34** (2020), 10542–10550. https://doi.org/10.1609/aaai.v34i07.6626
57. P. Gopalan, S. Gerrish, M. Freedman, D. Blei, D. Mimno, Scalable inference of overlapping communities, in *Proc. Conf. Workshop on Neural Inform. Process. Syst.,* (2012), 2249–2257.
58. R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Süsstrunk, SLIC superpixels compared to state-of-the-art superpixel methods, *IEEE Trans. Pattern Anal. Mach. Intell.*, **34** (2012), 2274–2282. https://doi.org/10.1109/TPAMI.2012.120
59. W. Zheng, P. Jing, Q. Xu, Action Recognition Based on Spatial Temporal Graph Convolutional Networks, in *Proc. 3rd Int. Conf. Comput. Sci. Appl. Eng.*, **118** (2019), 1–5. https://doi.org/10.1145/3331453.3361651
60. D. Tian, Z. Lu, X. Chen, L. Ma, An attentional spatial temporal graph convolutional network with co-occurrence feature learning for action recognition, *Multimed. Tools Appl.*, **79** (2020), 12679–12697. https://doi.org/10.1007/s11042-020-08611-4
61. Y. Chen, G. Ma, C. Yuan, B. Li, H. Zhang, F. Wang, et al., Graph convolutional network with structure pooling and joint-wise channel attention for action recognition, *Pattern Recogn.*, **103** (2020), 107321. https://doi.org/10.1016/j.patcog.2020.107321
62. J. Dong, Y. Gao, H. J. Lee, H. Zhou, Y. Yao, Z. Fang, et al., Action Recognition Based on the Fusion of Graph Convolutional Networks with High Order Features, *Appl. Sci.*, **10** (2020), 1482. https://doi.org/10.3390/app10041482

63. Z. Chen, S. Li, B. Yang, Q. Li, H. Liu, Multi-Scale Spatial Temporal Graph Convolutional Network for Skeleton-Based Action Recognition, in *Proc. Int. Joint Conf. Artif. Intell.*, **35** (2021), 1113–1122. https://doi.org/10.1609/aaai.v35i2.16197
64. Y. Bin, Z. Chen, X. Wei, X. Chen, C. Gao, N. Sang, Structure-aware human pose estimation with graph convolutional networks, *Pattern Recogn.,* **106** (2020), 107410. https://doi.org/10.1016/j.patcog.2020.107410
65. R. Wang, C. Huang, X. Wang, Global Relation Reasoning Graph Convolutional Networks for Human Pose Estimation, *IEEE Access*, **8** (2020), 38472–38480. https://doi.org/10.1109/ACCESS.2020.2973039
66. T. Sofianos, A. Sampieri, L. Franco, F. Galasso, Space-Time-Separable Graph Convolutional Network for Pose Forecasting, in *Proc. IEEE/ICCV Int. Conf. Comput. Vision*, (2021), 11209–11218. https://doi.org/10.48550/arXiv.2110.04573
67. Z. Zou, W. Tang, Modulated Graph Convolutional Network for 3D Human Pose Estimation, in *Proc. ICCV*, (2021), 11457–11467. https://doi.org/10.1109/ICCV48922.2021.01128
68. B. Yu, H. Yin, Z. Zhu, Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting, in *Proc. Int. Joint Conf. Artif. Intell.*, (2018), 3634–3640. https://doi.org/10.24963/ijcai.2018/505
69. Y. Han, S. Wang, Y. Ren, C. Wang, P. Gao, G. Chen, Predicting Station-Level Short-Term Passenger Flow in a Citywide Metro Network Using Spatiotemporal Graph Convolutional Neural Networks, *ISPRS Int. J. Geo-Inform.*, **8** (2019), 243. https://doi.org/10.3390/ijgi8060243
70. B. Zhao, X. Gao, J. Liu, J. Zhao, C. Xu, Spatiotemporal Data Fusion in Graph Convolutional Networks for Traffic Prediction, *IEEE Access*, **8** (2020), 76632–76641. https://doi.org/10.1109/ACCESS.2020.2989443
71. L. Ge, H. Li, J. Liu, A. Zhou, Temporal Graph Convolutional Networks for Traffic Speed Prediction Considering External Factors, in *Proc. Int. Conf. Mobile Data Manag.*, (2019), 234–242. https://doi.org/10.1109/MDM.2019.00-52
72. L. Ge, S. Li, Y. Wang, F. Chang, K. Wu, Global Spatial-Temporal Graph Convolutional Network for Urban Traffic Speed Prediction, *Appl. Sci.-basel*, **10** (2020), 1509. https://doi.org/10.3390/app10041509
73. P. Han, P. Yang, P. Zhao, S. Shang, Y. Liu, J. Zhou, et al., GCN-MF: Disease-Gene Association Identification by Graph Convolutional Networks and Matrix Factorization, *Knowl. Disc. Data Min.*, (2019), 705–713. https://doi.org/10.1145/3292500.3330912
74. J. Li, Z. Li, R. Nie, Z. You, W. Bao, FCGCNMDA: predicting miRNA-disease associations by applying fully connected graph convolutional networks, *Mol. Genet. Genom.*, **295** (2020), 1197–1209. https://doi.org/10.1007/s00438-020-01693-7
75. L. Wang, Z. You, Y. Li, K. Zhang, Y. Huang, GCNCDA: A new method for predicting circRNA-disease associations based on Graph Convolutional Network Algorithm, *PLoS Comput. Biol.*, **16** (2020), e1007568. https://doi.org/10.1371/journal.pcbi.1007568
76. C. Wang, J. Guo, N. Zhao, Y. Liu, X. Liu, G. Liu, et al., A Cancer Survival Prediction Method Based on Graph Convolutional Network, *IEEE Trans. NanoBiosci.*, **19** (2019), 117–126. https://doi.org/10.1109/TNB.2019.2936398
77. H. Chen, F. Zhuang, L. Xiao, L. Ma, H. Liu, R. Zhang, et al., AMA-GCN: Adaptive Multi-layer Aggregation Graph Convolutional Network for Disease Prediction, in *Proc. IJCAI*, (2021), 2235–2241. https://doi.org/10.24963/ijcai.2021/308

78. K. Gopinath, C. Desrosiers, H. Lombaert, Learnable Pooling in Graph Convolutional Networks for Brain Surface Analysis, *IEEE Trans. Pattern Anal. Mach. Intell.*, **44** (2022), 864–876. https://doi.org/10.1109/TPAMI.2020.3028391
79. R. Ying, R. He, K. Chen, Graph Convolutional Neural Networks for Web-Scale Recommender Systems, in *Proc. Knowl. Disc. Data Min.*, (2018), 974–983. https://doi.org/10.1145/3219819.3219890
80. X. Xia, H. Yin, J. Yu, Q. Wang, L. Cui, X. Zhang, Self-Supervised Hypergraph Convolutional Networks for Session-based Recommendation, in *Proc. Int. Joint Conf. Artif. Intell.*, **35** (2021), 4503–4511. https://doi.org/10.1609/aaai.v35i5.16578
81. H. Chen, L. Wang, Y. Lin, C. Yeh, F. Wang, H. Yang, Structured Graph Convolutional Networks with Stochastic Masks for Recommender Systems, in *Proc. SIGIR*, (2021), 614–623. https://doi.org/10.1145/3404835.3462868
82. L. Chen, Y. Xie, Z. Zheng, H. Zheng, J. Xie, Friend Recommendation Based on Multi-Social Graph Convolutional Network, *IEEE Access*, **8** (2020), 43618–43629. https://doi.org/10.1109/ACCESS.2020.2977407
83. T. Zhong, S. Zhang, F. Zhou, K. Zhang, G. Trajcevski, J. Wu, Hybrid graph convolutional networks with multi-head attention for location recommendation, *World Wide Web*, **23** (2020), 3125–33151. https://doi.org/10.1007/s11280-020-00824-9
84. T. H. Nguyen, R. Grishman, Graph Convolutional Networks with Argument-Aware Pooling for Event Detection, in *Proc. AAAI Confer. Artif. Intell.,* **32** (2018). https://doi.org/10.1609/aaai.v32i1.12039
85. Z. Guo, Y. Zhang, W. Lu, Attention Guided Graph Convolutional Networks for Relation Extraction, *Ann. Meet. Assoc. Comput. Linguist.*, (2019), 241–251. https://doi.org/10.18653/v1/P19-1024
86. Y. Hong, Y. Liu, S. Yang, K. Zhang, A. Wen, J. Hu, Improving Graph Convolutional Networks Based on Relation-Aware Attention for End-to-End Relation Extraction, *IEEE Access*, **8** (2020), 51315–51323. https://doi.org/10.1109/ACCESS.2020.2980859
87. Z. Meng, S. Tian, L. Yu, Y. Lv, Joint extraction of entities and relations based on character graph convolutional network and Multi-Head Self-Attention Mechanism, *J. Exp. Theor. Artif. Intell.*, 33 (2021), 349–362. https://doi.org/10.1080/0952813X.2020.1744198
88. L. Yao, C. Mao, Y. Luo, Graph Convolutional Networks for Text Classification, *Artif. Intell.*, (2019), 7370–7377. https://doi.org/10.1609/aaai.v33i01.33017370
89. M. Chandra, D. Ganguly, P. Mitra, B. Pal, J. Thomas, NIP-GCN: An Augmented Graph Convolutional Network with Node Interaction Patterns, in *Proc. SIGIR*, (2021), 2242–2246. https://doi.org/10.1145/3404835.3463082
90. L. Xiao, X. Hu, Y. Chen, Y. Xue, D. Gu, B. Chen, et al., Targeted Sentiment Classification Based on Attentional Encoding and Graph Convolutional Networks, *Appl. Sci.*, **10** (2020), 957. https://doi.org/10.3390/app10030957
91. P. Zhao, L. Hou, O. Wu, Modeling sentiment dependencies with graph convolutional networks for aspect-level sentiment classification, *Knowl.-Based Syst.*, **193** (2020), 105443. https://doi.org/10.1016/j.knosys.2019.105443
92. S. Jiang, Q. Chen, X. Liu, B. Hu, L. Zhang, Multi-hop Graph Convolutional Network with High-order Chebyshev Approximation for Text Reasoning, *arXiv preprint*, (2021), arXiv:2106.05221. https://doi.org/10.18653/v1/2021.acl-long.513

93. R. Li, H. Chen, F. Feng, Z. Ma, X. Wang, E. Hovy, Dual Graph Convolutional Networks for Aspect-based Sentiment Analysis, in *Proc. 59 Ann. Meet. Assoc. Comput. Linguist. And 11th Int. joint Conf. Nat. Language process.*, **1** (2021), 6319–6329.
94. L. Lv, J. Cheng, N. Peng, M. Fan, D. Zhao, J. Zhang, Auto-encoder based Graph Convolutional Networks for Online Financial Anti-fraud, *IEEE Comput. Intell. Financ. Eng. Econ.*, (2019), 1–6. https://doi.org/10.1109/CIFEr.2019.8759109
95. C. Li, D. Goldwasser, Encoding Social Information with Graph Convolutional Networks for Political Perspective Detection in News Media, in *Proc. 57th Ann. Meet. Assoc. Comput. Linguist.*, (2019), 2594–2604. https://doi.org/10.18653/v1/p19-1247
96. Y. Sun, T. He, J. Hu, H. Hang, B. Chen, Socially-Aware Graph Convolutional Network for Human Trajectory Prediction, in *2019 IEEE 3rd Inf. Technol. Network. Electron. Autom. Control Conf. (ITNEC)*, (2019), 325–333. https://doi.org/10.1109/ITNEC.2019.8729387
97. J. Chen, J. Li, M. Ahmed, J. Pang, M. Lu, X. Sun, Next Location Prediction with a Graph Convolutional Network Based on a Seq2seq Framework, *KSII Trans. Internet Inf. Syst.*, **14** (2020), 1909–1928. https://doi.org/10.3837/tiis.2020.05.003
98. X. Li, Y. Xin, C. Zhao, Y. Yang, Y. Chen, Graph Convolutional Networks for Privacy Metrics in Online Social Networks, *Appl. Sci.-Basel*, **10** (2020), 1327. https://doi.org/10.3390/app10041327

AIMS Press