Neurocomputing 387 (2020) 321–333

Contents lists available at ScienceDirect

Neurocomputing

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

Design of affinity-aware encoding by embedding graph centrality for graph classification

Wei Dong[a], Junsheng Wu[a,b,*], Zongwen Bai[a,c], Weigang Li[b], Wei Qiao[a,d]

[a] School of Computer Science and Engineering, Northwestern Polytechnical University, China
[b] School of Software, Northwestern Polytechnical University, China
[c] School of Physics and Electronic Information, Yan'an University, China
[d] Xi'an Research Institute of China Coal Technology & Engineering Group Corp, China

ARTICLE INFO

Article history:
Received 27 November 2018
Revised 4 November 2019
Accepted 9 January 2020
Available online 13 January 2020

Communicated by Dr. Jianjun Lei

Keywords:
Graph centrality
Affinity-aware encoding
Separate encoding function
Graph classification

ABSTRACT

Deep learning methods for graph classification are critical for graph data mining. Recently, graph convolutional networks (GCNs) have been able to achieve state-of-the-art node classification. A typical process for GCNs includes two iterative steps: node feature encoding and message passing. While the former encodes each graph node independently via the uniform encoding function, the latter updates the features of each node by weighted aggregation of the features of neighboring nodes, where the weights are generated by predefined or learned graph Laplacian. However, their accuracy deteriorates for graph classification tasks because the uniform encoding function encodes all the node features involved. In this study, we propose a novel affinity-aware encoding for graph classification. In our model, we implement a separate encoding function for the neighboring nodes of each node for updating the node features, where the nodes are arranged in the order of affinity values, such as graph centrality, in order to determine the correspondence between an encoding function and a specific neighboring node. Our separate encoding function performs non-Euclidean neighboring encoding for each node by weight sharing, which enables message passing. We also develop two variants based on our separate encoding function: the graph centrality-convolutional neural network (C-CNN) and the graph centrality-graph convolutional network (C-GCN). The former performs the separate encoding function on graph data directly by the function of message passing. The latter combines the separate encoding function with the normalized graph Laplacian implemented on the graph data. Experiments demonstrate that the results obtained by our models are consistent with those obtained by classical convolutional neural networks (CNNs) on the MNIST dataset, and they outperform existing GCNs on the 20NEWS, Reuters8, and Reuters52 datasets. We also apply our two variants to online car-hailing service data for traffic congestion recognition. Our methods show state-of-the-art results compared with GCNs.

1. Introduction

Deep learning methods such as convolutional neural networks (CNNs) [1–3] have emerged as powerful platforms for learning Euclidean structured data, such as images, videos, audios, and text. Their application has led to remarkable achievements in many areas of machine intelligence, enabling superhuman accuracy in challenging tasks of computer vision [4–7], speech recognition [8–10], and natural language processing [11,12]. In recent years, there has been an increasing interest in leveraging deep learning methods into graphs [2,13], thus extending deep learning applications from learning Euclidean structured data to understanding non-Euclidean structured data. Graphs are indispensable theoretical models to represent the relationship between objects in the real world, where each entity or object is described as a node, and their affiliation are abstracted as edges in the graphs. Graph models are used to study information spreading [14,15], to recommend new social relationship [16,17] or to represent molecular structures [18,19].

Graph convolutional networks (GCNs), one of the most significant methods that utilize deep learning to analyze undirected, unweighted, and connected graphs, play an essential role in addressing the issue of semi-supervised node classification, including spectral graph convolutional networks [20,21], neighbor aggregation (message passing) algorithms [22–28], and message passing via recurrent neural networks [29–31]. Several other GCNs have been designed to improve the underlying message passing scheme

* Corresponding author at: School of Software, Northwestern Polytechnical University, China
E-mail addresses: dw156@mail.nwpu.edu.cn (W. Dong), wujunsheng@nwpu.edu.cn (J. Wu).

https://doi.org/10.1016/j.neucom.2020.01.010
0925-2312/

by using attention mechanisms [22,24,32]. Furthermore, graph representation learning using message passing algorithms [33–35] can be contributed to develop GCNs. These GCNs have been attracting considerable interest recently due to their excellent performance.

A typical pipeline for GCNs includes two iterative steps: node feature encoding and message passing. While the former encodes each graph node independently with the uniform encoding function, the latter updates the features of each node by weighted aggregation of the features of neighboring nodes, where the weights are generated by predefined or learned graph Laplacian [20,21,23,28]. However, an essential inadequacy with these GCNs is that they directly perform subsequent node encoding on the aggregated features, which means that they apply uniform encoding functions to all the node features involved. This inadequacy leads to the deterioration of accuracy of graph classification, because the uniform encoding function encodes all the node features for each node via a weight matrix, without considering separate neighboring encoding for each node. Moreover, replacing or modifying the uniform encoding function is difficult owing to the non-Euclidean structure of graphs.

In this study, we propose novel affinity-aware encoding for graph classification. In our model, we implement a separate encoding function to the neighboring nodes of each node for updating node features, where the nodes are arranged in the order of affinity values such as graph centrality in order to determine the correspondence between an encoding function and a specific neighboring node. Graph centrality includes four frequently used metrics, namely, degree centrality (DC), betweenness centrality (BC) [36,37], closeness centrality (CC) [38], and eigenvector centrality (EC) [39], which will be considered in our separate encoding function. Other new metrics of graph centrality have been proposed in the past, such as message passing centrality [40] and a new metric of graph centrality for protein networks [41]. In the future, we will try to use these new metrics to build our separate encoding function. The proposed separate encoding function uses non-Euclidean neighboring encoding for each node by weight sharing, which enables message passing. The number of parameters in the separate encoding increases linearly with the number of neighbors of the maximum degree node. There are two variants of the model based on the separate encoding function. The first variant is to utilize the separate encoding function directly on non-Euclidean structured data, and it is named the graph centrality CNN (C-CNN), due to the similarity with the structure of the CNN. The second variant is to combine the separate encoding function and the normalized graph Laplacian of GCNs [23], and it is named the graph centrality GCN (C-GCN). The difference between the two types is the introduction of normalized graph Laplacian in the latter. Experiments demonstrate that the feasibility of our models is consistent with those of classical CNNs on the MNIST dataset, and the results of them are better than those of GCN methods on 20NEWS, Reuters8, and Reuters52 datasets. In addition, we apply our two variants on the online car-hailing service data of Xi'an and Chengdu in China. We combine the two variants with recursive neural networks (RNNs) to predict traffic congestion in the road graph of a city. To the best of our knowledge, our experimental results for traffic congestion recognition are state-of-the-art compared with those of GCN methods. The principal contributions of this study are two-fold:

Separate encoding: We design a novel separate encoding function to enhance the accuracy of graph classification tasks. It learns by weight sharing or perceiving the non-Euclidean neighboring features for each node. Compared to the uniform encoding function, the separate encoding function has the advantage of being able to learn node encoding via inconsistent neighborhood.

Expansibility: This work draws lessons from many CNN methods for the innovation of suitability and usability in GCNs because the separate encoding function is based on a graph convolution extended from the standard convolution in CNNs. For instance, one major drawback of GCNs [22–28] is that the graph Laplacian training and evaluation involve large-scale weight matrices and incur high matrix multiplication cost. In future work, we plan to import depthwise separable convolution (DSC) [42,43] into our separate encoding function to explicitly reduce the model costs. The use of DSC has tailored the standard convolution in AlexNet [4] and ResNet [7] with reduced model costs to be run on mobile and embedded devices.

The remainder of this paper proceeds as follows: Section 2 presents background information on spectral and spatial methods in GCNs. Section 3 details the proposed separate encoding function. Section 4 presents the experimental setup. Finally, the experimental results are discussed in Section 5, and the findings of this study are summarized in Section 6.

2. Related work

Veličković et al. [32] summarized early applications of neural networks on graphs, including RNNs, which can be used to process data as directed acyclic graphs [44,45]. Gori et al. [46] and Scarselli et al. [47] introduced graph neural networks (GNNs) based on RNNs. Subsequently, Li et al. [30] proposed the use of gated recurrent units [48] in the propagation process of GNNs. Nevertheless, when CNNs have been acquiring much concern in recent years, how to generalize convolution encoding on the graph is increasing interest. In this regard, two approaches have been developed: spectral and spatial methods.

2.1. Spectral methods

Bruna et al. [20] first proposed convolution encoding on graphs, and they pioneered early research on GCNs. However, because their convolution encoding is global, so their work incurs intensive computing costs. Henaff et al. [49] further improved Bruna et al.âs method [20] to make it suitable for large-scale and semi-supervised learning tasks. Their method applies the Spline Kernel to decrease weights in order to make encoding functions spatially localized. Defferrard et al. [21] proposed approximate filters using the means of Chebyshev encoding based on a powerful message passing algorithm, normalized graph Laplacian, which simplifies calculations and reduces the number of model weights required. Furthermore, Kipf and Welling [23] utilized localized first-order approximation to simplify the normalized graph Laplacian of Defferrard et al. âs method [21] for aggregating node features, and subsequently building a uniform encoding function based on these aggregated features. Further, Li et al. [50] designed both co-training and self-training approaches to overcome the limitations of GCNs [23] with shallow architectures. Finally, John et al. [51] replaced the adjacent matrix of the normalized graph Laplacian [23] with a motif-based matrix. Klicpera et al. [28] used personalized PageRank to improve the normalized graph Laplacian and to achieve better accuracy. In the applications of GCNs, Yao et al. [52] proposed a text-GCN model to apply to text classification. Wang et al. [53] utilized GCNs [23] for high-dimensional stream classification. Although these researchers have made various advancements in spectral GCNs in terms of increased accuracy, they only focus on how to optimize the message passing algorithms. In our study, we decouple the uniform encoding function [21,23,28] to achieve separate encoding of inconsistent neighboring features of each node and consequently design separate encoding functions to increase model accuracy.

2.2. Spatial methods

Compared with spectral methods, which use a unified message passing algorithm (graph Laplacian), spatial methods are of various types. However, they have the same goal of defining message passing algorithms for weighted aggregation of node features. Duvenaud et al. [54] designed an aggregation function to learn the specific weight matrix for each node degree. Atwood and Towsley [55] used the powers of a transition matrix to define the neighbor aggregation algorithm. Niepert et al. [56] built a new message passing framework to learn arbitrary graphs. Monti et al. [26] proposed mixture CNNs, which provide a unified structure for image and graph data. Hamilton et al. [24] introduced GraphSAGE for node representations. Chen and Zhu [57] developed a preprocessing strategy and two control variables to converge the value of cost function to the local optimum regardless of the neighbor sampling size. Hechtlinger et al. [58] use random walk algorithm to propose a new spatial message passing, which can be applied to many standard regressions or classification applications. Zhou and Li [59] presented a *k*-th order algorithm and an adaptive filtering module for a general-purpose architecture, which fits various applications on both node-level and graph-level tasks. Verma et al. [60] proposed a dynamic message passing algorithm that fits neighborhoods of different sizes. Chen et al. [61] proposed a method that can exploit a known underlying graph structure of labels. Lai et al. [62] proposed the depthwise separable graph convolution (DSGC) based on data manifold. Wu et al. [63] utilized a Gaussian mixture model to realize the mapping between the aggregation kernel and the nodes for irregular data. Recently, Veličković et al. [32] and Shanthamallu et al. [64] presented graph attention networks (GATs) based on attention mechanisms for node-level tasks. These methods introduce or upgrade non-Laplacian message passing algorithms, but they only consider node feature encoding with a uniform encoding function due to the inconsistent size of the neighborhood.

3. Methodology

3.1. Separate encoding function

To establish convolutional layers of the proposed separate encoding function, we introduce graph centrality. The following two steps are involved in achieving the separate encoding function.

Establishing convolutional layers: First, we utilize the number of convolutional layers L and the size of feature map (number of units) $|F^{(l)}|$ per layer l to determine the model structure $\boldsymbol{F} = [F^{(1)}, F^{(2)}, \ldots, F^{(l)}, \ldots, F^{(L)}]$, where $F^{(l)}$ is the feature map in layer l. Next, the $|F^{(l)}|$ most important nodes defined by an optional graph centrality metric are chosen as units in each layer l, as shown in Fig. 1. The pseudo-code of building convolutional layers is presented in Algorithm 1 , where $G(N, E)$ is a graph G including nodes N and edges E, $\boldsymbol{S}$ contains the size of the feature map in every layer, and $\boldsymbol{F}$ includes feature maps in our model.

Constructing connections: Connections between layers are built considering that each node in layer $l-1$ is connected to itself and its neighbors in layer l (solid lines in Fig. 1). However, some nodes in layer $l-1$ may not have neighbors and itself in layer l, because all nodes in layer l is a subset of nodes in layer $l-1$, i.e. $F^{(l)} \subseteq F^{(l-1)}$, where $|F^{(l)}| \leq |F^{(l-1)}|$. Our solution to this issue is to connect these isolated nodes in layer $l-1$ with their closest nodes in layer l; these closest nodes are described by the Dijkstra algorithm [65] in graph G. For example, in Fig. 1, unit "c" in layer 3 has no neighbors in layer 4, and in layer 4 "c" itself is not present, resulting in unit "c" in layer 3 having no connections to layer 4. We find that the closest node to node "c" is node "a" according to graph G (the blue graph); then, a connection between unit "c" in layer 3 and unit "a" in layer 4 (a dotted line between layers 3 and 4 in Fig. 1) is built. The pseudo-code of building connection is given in Algorithm 2 , where $G(N, E)$ is a graph G including nodes N and edges E, $\boldsymbol{F}$ includes feature maps in our model, and *closeness* is a dictionary to store the shortest path of each node pair in G. By contrast, one unit may have many connections. In an extreme case, the number of connections for a unit can be $|N|$, which leads to the size of the separate encoding function being $|N|$. Here, we introduce a hyper-parameter *limit* in Algorithm 2 to limit the maximum number of connections in a unit. ***CON*** denotes the model connections between layers, and it defines how each unit interacts with other units.

Algorithm 1: Establishing Convolutional Layers.

Input: Graph $G(N, E)$; The list of feature map's sizes $\boldsymbol{S} = [|F^{(1)}|, |F^{(2)}|, \ldots, |F^{(l)}|, \ldots, |F^{(L)}|]$; The metric of graph centrality $IC, \forall IC \in \{DC, BC, CC, EC\}$

Output: The list of feature maps $\boldsymbol{F} = [F^{(\text{input})}, F^{(1)}, \ldots, F^{(l)}, \ldots, F^{(L)}]$

```
F ← ∅, F^(input) ← N, F ← F ∪ F^(input);
for each |F^(l)| ∈ S do
    subgraphs ← get connected component subgraphs in G;
    for each subgraph ∈ subgraphs do
        if the number of nodes in subgraph > 1 then
            rati ← the number of nodes in subgraph / the number of nodes in G;
            F^(l) ← F^(l) ∪ top int(rati * |F^(l)|) most important nodes N_importance ⊆ N are selected by IC;
        if the number of nodes in subgraph = 1 then
            F^(l) ← F^(l) ∪ nodes in subgraph;
    F ← F ∪ F^(l);
return F;
```

The following three steps are involved in designing the graph convolution of the separate encoding function. First, we construct the neighborhood for each node. Second, weight sharing is introduced into the separate encoding function. Finally, two variants of the separate encoding function (C-CNN and C-GCN) are proposed based on the previous steps.

Constructing neighborhood: We first use the GCN [23], which is defined as

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

where $\hat{A} = A + I_{|N|}$, $I_{|N|}$ is an identity matrix, $A \in \mathbb{R}^{|N|\times|N|}$ is an adjacency matrix, $|N|$ is the number of nodes in G, $\hat{D} = \sum_j \hat{A}_{ij}$, $H^{(l)} \in \mathbb{R}^{|N|\times|C^{(l)}|}$ is feature matrix of nodes, $C^{(l)}$ is the feature channels and $|C^{(l)}|$ is the number of feature channels, and $W^{(l)} \in \mathbb{R}^{|C^{(l)}|\times|C^{(l+1)}|}$ is a trainable weight matrix in layer l.

In this work, we change the form of $H^{(l)} \in \mathbb{R}^{|N|\times|C^{(l)}|}$ to $H^{(l)} \in \mathbb{R}^{|B|\times|F^{(l)}|\times|C^{(l)}|}$, where B denotes the samples for the mini-batch, $|B|$ is the number of samples, and $|F^{(l)}|$ is the size of the feature map in layer l. In particular, $H^{(\text{input})} \in \mathbb{R}^{|B|\times|N|\times|C^{(l)}|}$ because $|F^{(\text{input})}| = |N|$ in Algorithm 1. The difference between $H^{(l)}$ of our study and paper [23] is that $H^{(l)} \in \mathbb{R}^{|N|\times|C^{(l)}|}$ is designed for node classification, its $|N|$ is unchanged in every layer, and it has two dimensions. By contrast, our $H^{(l)} \in \mathbb{R}^{|B|\times|F^{(l)}|\times|C^{(l)}|}$ has three dimensions for graph classification, and its $|F^{(l)}|$ can be changed by $\boldsymbol{F}$ in Algorithm 1. Specifically, we define *mask* function as follows

$$H_n^{(l+1)} = mask(H^{(l)})_n = \rho(H^{(l)} \odot M_n^{(l+1)}) \tag{2}$$

where $H_n^{(l+1)}$ is a neighborhood of unit n in layer $l+1$, $\odot$ is an element-wise product for two matrices or tensors, and $M_n^{(l+1)}$ is

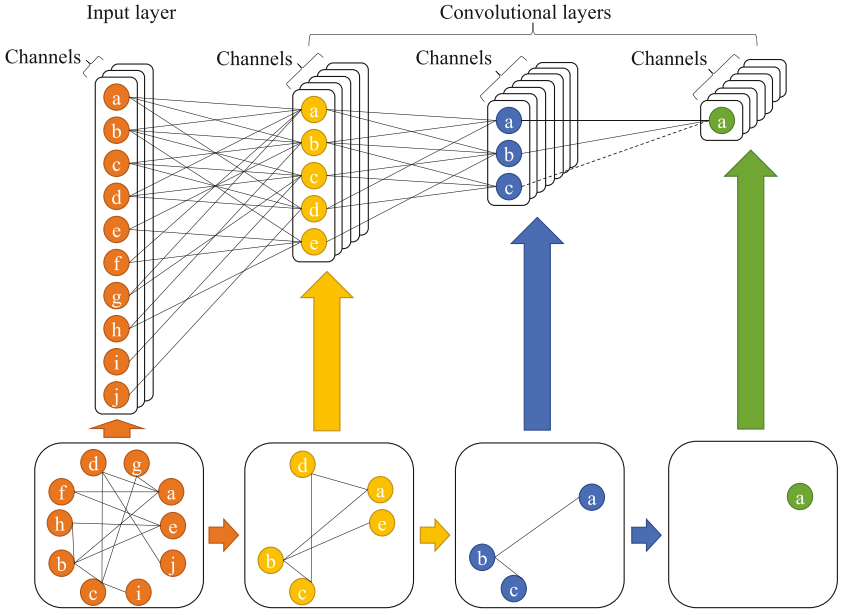


Fig. 1. Each node in graph G is a unit in layer l to build the model structure, the $|F^{(l)}|$ most important nodes defined by an optional graph centrality metric are chosen as units in each layer l. Connections between layers are built considering that each node n in layer $l-1$ is connected to itself, its neighbors (solid lines), and its closeness nodes (dotted line between layer 3 and layer 4) in layer l.

Algorithm 2: Constructing Connections.

Input: Graph $G(N, E)$; $\boldsymbol{F} = [F^{(\text{input})}, F^{(1)}, \ldots, F^{(l)}, \ldots, F^{(L)}]$; Dictionary *closeness* records the Dijkstra path length of node pairs sorted in ascending order; *limit* is to limit the number of connections.

Output: The list of connections $\boldsymbol{CON} = [CON^{(1)}, \ldots, CON^{(l)}, \ldots, CON^{(L)}]$, $\forall CON^{(l)}(K|V)$ is a dictionary, where K is a node n, V is a set including $n$âs neighbors, itself n and its closeness nodes.

$\boldsymbol{CON} \leftarrow \varnothing$;
for *each* $l \in [1, \ldots, L]$ **do**
  $CON^{(l)} \leftarrow \varnothing$;
  set $\boldsymbol{T} \leftarrow \varnothing$;
  for *each* $n \in F^{(l)}$ **do**
    $CON^{(l)}[n] \leftarrow \varnothing$;
    for *each* $k \in$ *keys in closeness*$[n]$ **do**
      if *the length of* $CON^{(l)}[n] >$ *limit* **then**
        break;
      if $k \in F^{(l-1)}$ **then**
        $CON^{(l)}[n] \leftarrow CON^{(l)}[n] \cup k$;
        $\boldsymbol{T} \leftarrow \boldsymbol{T} \cup k$;
  for *each* $n \in (F^{(l-1)} - \boldsymbol{T})$ **do**
    for *each* $k \in$ *keys in closeness*$[n]$ **do**
      if $k \in F^{(l)}$ **then**
        $CON^{(l)}[k] \leftarrow CON^{(l)}[k] \cup n$;
        if *the length of* $CON^{(l)}[k] >$ *limit* **then**
          break;
  $\boldsymbol{CON} \leftarrow \boldsymbol{CON} \cup CON^{(l)}$;
return $\boldsymbol{CON}$;

the mask matrix or tensor of unit n in $l+1$ layer, with its elements being 0 or 1 and shape being the same as that of $H^{(l)}$. If unit n and unit k have a connection defined in ***CON*** (Algorithm 2), elements in $k \in F^{(l)}$ of $M_n^{(l+1)}$ will be set to 1. Otherwise, they will be set to 0. The compression function ρ deletes 0 elements to show in Fig. 2. In addition, $H_n^{(l+1)} \in \mathbb{R}^{|B| \times |D_n^{(l+1)}| \times |C^{(l)}|}$, where $D_n^{(l+1)}$ is the connections of unit n in ***CON*** and $|D_n^{(l+1)}|$ is the number of connections. Furthermore, Eq. (2) can be extended as

$$H_{\text{mask}}^{(l+1)} = mask(H^{(l)}) = \rho(H^{(l)} \odot [M_1^{(l+1)}, \ldots M_n^{(l+1)}, \ldots M_{|F^{(l+1)}|}^{(l+1)}]) \tag{3}$$

where $H_{\text{mask}}^{(l+1)}$ is a list of length $|F^{(l+1)}|$, and it contains each $H_n^{(l+1)}$.

Introducing weight sharing: One problem in our structure is that each unit n has inconsistent neighborhood, i.e., each $H_n^{(l+1)}$ in list $H_{\text{mask}}^{(l+1)}$ has different $|D_n^{(l+1)}|$. Therefore, we find the maximum $|D_n^{(l+1)}|$, denoted as $|D_{\text{max}}^{(l+1)}|$, for normalizing inconsistent neighborhood. Because $|D_n^{(l+1)}| \le |D_{\text{max}}^{(l+1)}|$, element 0 can be padded into $H_n^{(l+1)}$ along dimension $D_n^{(l+1)}$ until $|D_n^{(l+1)}| = |D_{\text{max}}^{(l+1)}|$. Now in list $H_{\text{mask}}^{(l+1)}$, the shape of every $H_n^{(l+1)}$ is $(|B|, |D_{\text{max}}^{(l+1)}|, |C^{(l)}|)$, and they have consistent neighborhood for each node, which allows weight sharing to be performed on it. In the padding process shown in Fig. 3, we create some dummy zero nodes in order to make every colored node have the same degree $|D_{\text{max}}^{(l+1)}|$.

Designing models: Finally, two variants, the C-CNN and C-GCN, of our separate encoding function are established by previous steps. The C-CNN is defined as

$$H^{(l+1)} = \sigma(transpose(padding(H_{\text{mask}}^{(l+1)})W^{(l)})) \tag{4}$$

$$H^{(l+1)} = \sigma(transpose(padding(mask(H^{(l)}))W^{(l)})) \tag{5}$$

$$H^{(l+1)} = \sigma(transpose(padding(\rho(H^{(l)} \odot [M_1^{(l+1)}, \ldots M_n^{(l+1)}, \ldots M_{|F^{(l+1)}|}^{(l+1)}]))W^{(l)})) \tag{6}$$

where $W^{(l)} \in \mathbb{R}^{|D_{\text{max}}^{(l+1)}| \times |C^{(l)}| \times |C^{(l+1)}|}$ is the trainable weight tensor, $H^{(l+1)} \in \mathbb{R}^{|B| \times |F^{(l+1)}| \times |C^{(l+1)}|}$ is the output in layer $l+1$, and σ is an

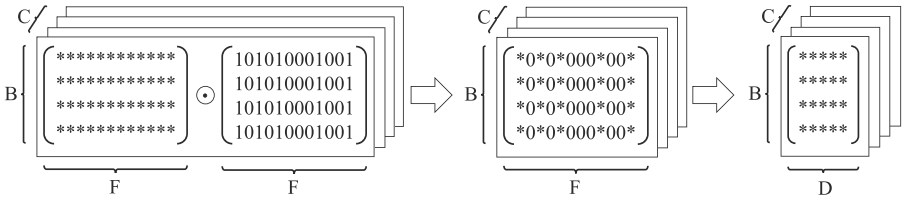


Fig. 2. Process of the *mask* function in Eq. (2). It is used to construct the neighborhood of unit n from layer l to layer $l+1$.

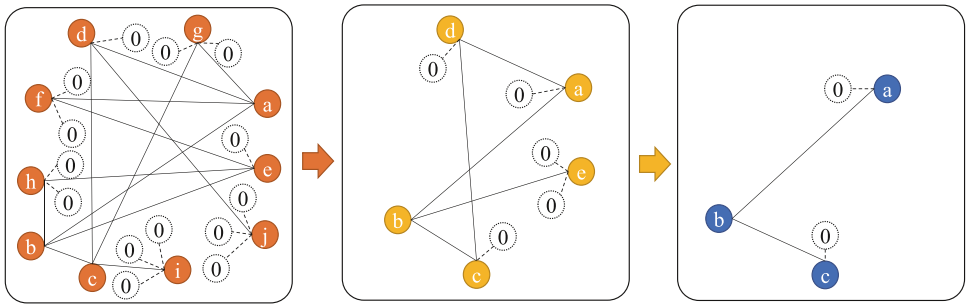


Fig. 3. Creating some dummy zero nodes so that every colored node has the same degree $D_{\max}^{(l+1)}$ in layer $l+1$.

activation function. The *padding* function is the process of padding $H_n^{(l+1)}$ as shown in Fig. (3). Each $H_n^{(l+1)}$ in list $H_{\text{mask}}^{(l+1)}$ is multiplied by tensor $W^{(l)}$. However, $H_n^{(l+1)}$ and $W^{(l)}$ must be, respectively, reshaped as $(|B|, |D_{\max}^{(l+1)}| \times |C^{(l)}|)$ and $(|D_{\max}^{(l+1)}| \times |C^{(l)}|, |C^{(l+1)}|)$, and consequently matrix multiplication of $H_n^{(l+1)}W^{(l)}$ in Eq. (6) can be performed. The *transpose* function transforms the shape of $H^{(l+1)}$ from $(|F^{(l+1)}|, |B|, |C^{(l+1)}|)$ to $(|B|, |F^{(l+1)}|, |C^{(l+1)}|)$, making it consistent with $(|B|, |F^{(l)}|, |C^{(l)}|)$ of $H^{(l)}$. Furthermore, the C-GCN is defined as

$$H^{(l+1)} = \sigma(transpose(padding(mask(mat(H^{(l)}, (\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2})^{(l)})))W^{(l)})) \quad (7)$$

where $(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2})^{(l)}$ is the normalized graph Laplacian in Eq. (1), and $\hat{A}^{(l)} = A_{|F^{(l)}|} + I_{|F^{(l)}|}$ as well as $(\hat{D}^{-1/2})^{(l)}$ correspond to a graph including $|F^{(l)}|$ nodes. Unlike Eq. (5) in which $H^{(l)}$ is directly performed by the *mask* function, in Eq. (7) each $H^{(l)}$ is first multiplied by the normalized graph Laplacian $(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2})^{(l)}$. Because the shape $(|F^{(l)}|, |F^{(l)}|)$ of the normalized graph Laplacian is different from $(|B|, |F^{(l)}|, |C^{(l)}|)$ of $H^{(l)}$, we define *mat* function to multiply the normalized graph Laplacian with each matrix in tensor $H^{(l)}$, where the shape of each matrix is $(|F^{(l)}|, |C^{(l)}|)$ and the number of matrices is $|B|$ in $H^{(l)}$.

3.2. Model cost

The cost of the separate encoding function is $\mathcal{O}(|D_{\max}^{(l)}||C^{(l-1)}||C^{(l)}|)$, which is less than $\mathcal{O}(|E||C^{(l-1)}||C^{(l)}|)$ [23] and $\mathcal{O}(|N||C^{(l-1)}||C^{(l)}|)$ [58,61] since $|D_{\max}^{(l)}| \le |N| \ll |E|+1$.

Theorem 1. *Given a connected graph G(N, E) introduced into the model, the cost of the separate encoding function in layer l is less than the number of nodes and edges of G, i.e.,* $|D_{\max}^{(l)}| \le |N| \le |E|+1$.

Proof. According to the separate encoding function, $|D_{\max}^{(l)}|$ is the maximum number of connections for one unit in layer l. It is possible to show that (a) if the size of feature map $|F^{(l)}|$ is properly defined without unconnected units, steps 13–18 in Algorithm 2 can be skipped. Thus, $|D_{\max}^{(l)}|$ is the maximum degree in G and $|D_{\max}^{(l)}| \le |N| \le |E|+1$; (b) if $|F^{(l)}|$ is improperly defined, steps 13–18 in Algorithm 2 must be carried out. Here, we consider the worst case—a unit n in layer l connects all units in layer $l-1$, then $|F^{(l-1)}| = |N|$ and $|D_{\max}^{(l)}| = |N|$. Because the number of edges $|E| \ge |N|-1$ in any connected graph G, $|D_{\max}^{(l)}| \le |N| \le |E|+1$. Therefore, $\mathcal{O}(|D_{\max}^{(l)}||C^{(l-1)}||C^{(l)}|) \le \mathcal{O}(|N||C^{(l-1)}||C^{(l)}|) \le \mathcal{O}(|E||C^{(l-1)}||C^{(l)}|)$. □

3.3. Dimensionality reduction without pooling operation

How to use pooling layers in GCNs is still a matter of debate. Some authors [20,21,49] point out that using heuristic approaches [66,67] to redesign pooling operation for GCNs has a positive effect on improving performances. Other authors propose their GCNs without pooling operation [23,58,61]. In this study, we omit heuristic pooling operations [20,21,49] because when they are used in our model, the performance is low and the output results of these heuristic methods are unstable [68,69]. Moreover, the size of the feature map $|F^{(l)}|$ in our model is a hyper-parameter, which can be used to reduce the size of feature maps, such as the capacity of pooling operation on the CNN. Therefore, our separate encoding function has partly the function of pooling operation, which allows for dimensionality reduction. Nevertheless, whether to utilize pooling operation for graph data is an open issue and needs further study in the future.

4. Experimental setup

4.1. Datasets

To verify the feasibility of the two variants (C-CNN and C-GCN) of our separate encoding function, we selected the MNIST dataset [70] as a benchmark, which is used to recognize handwritten digits and to verify the feasibility of other GCNs [20,21,26,58,60,63]. A total of 70,000 handwritten digital 28 × 28 images, divided into 10 categories, were used, of which 60,000 were used as a training set and 10,000 as a test set. Furthermore, we used 20NEWS [71], Reuters8 [52], and Reuters52 [52] datasets to demonstrate the availability of our models in text classification tasks. 20NEWS contains 11,314 training documents and 7532 testing documents, which are composed of 93,953 words and divided into 20 categories. Reuters8 and Reuters52 are two subsets of the Reuters 21,578 dataset. Reuters8 has 5,485 training and 2,189 test documents, which are divided into 8 categories. Reuters52 has 6532 training and 2568 test documents, which are divided

Table 1
Statistics of datasets for text classification.

Dataset	Docs	Training	Test	Words	Nodes	Classes
20NEWS	18,846	11,314	7532	93,953	10,000	20
Reuters8	7674	5485	2189	7688	7688	8
Reuters52	9100	6532	2568	8892	8892	52

Table 2
Settings of hyper-parameters.

Hyper-Parameter	Setting
Learning rate	0.03
Mini-batch size	128
Epochs	20
Dropout probability	0.5
Learning rate decay	0.95

into 52 categories. We selected top 10,000 high-frequency words from 20NEWS, 7688 words from Reuters8, and 8892 words from Reuters52 as nodes. Table 1 shows the statistics of datasets for text classification.

4.2. Baseline models

We compare our model with ten state-of-the-art GCN models for the MNIST dataset: the classical CNN, a Generalization of CNN [58], LRF [20], ChebyNet [21], MoNet [26], dynamic filters [60], GCN [23], motif-GCN [51], GAT [32], and PageRank-GCNs (PPNP and APPNP) [28]. For text classification, we compare our model with traditional models (i.e., linear SVM, multi-Naïve Bayes, softmax, and fully connected networks) and GCNs (i.e., ChebyNet [21], MoNet [26], GCN [23], DCNN [55], DSGC [62], text-GCN [52], motif-GCN [51], GAT [32], and PageRank-GCNs (PPNP and APPNP) [28]).

4.3. Hyper-Parameters

In the experiments, the k nearest neighbor (k-NN) method was used to construct graphs. For MNIST, each pixel was connected to its k-NN pixels to build the regular graph. The input was a vector composed of 784 pixels. For 20NEWS, Reuters8, and Reuters52, the following three steps were involved. First, words were embedded in a word-vector space through the Word2Vec algorithm [72]. Second, stop words were removed. Finally, according to the cosine distance of two word-vectors in the word-vector space, every word was connected as a node to its closest k words to construct the irregular graph. Table 1 shows the number of nodes for the three datasets. The input vector represented by Bag-of-Words is a $|N|$-dimensional vector in which elements are arranged in the order of nodesâ ID, where $|N|$ is the number of nodes in the irregular graph. We use the following notations to describe the frameworks: GC*c*:*s* denotes a convolutional layer in which c feature maps are contained, and each feature map consists of s units, and FC*s* denotes a fully connected layer with s units. All models are designed to use ReLU as the activation function in the graph centrality convolution. Our experimental program is based on TensorFlow. The hyper-parameter settings are shown in Table 2, where the learning rate decay is based on exponential decay, and the weights of models are learned using the Adam algorithm [73]. We chose these hyper-parameters according to paper [21], which are borrowed from the TensorFlow tutorial and experience.

The other crucial hyper-parameter is the graph centrality metric. In our experiments, four metrics of graph centrality (namely, degree centrality (DC), betweenness centrality (BC), closeness centrality (CC), and eigenvector centrality (EC)) describe the importance of nodes from different perspectives. DC only evaluates the number of neighbors of each node to define its importance, which is the simplest way to measure graph centrality. However, DC is not sufficient to fully describe the nodesâ importance. Some other factors play an essential role in graph centrality. BC emphasizes the path control of nodes like hubs of communication in cities—if a node in the shortest paths between two nodes has the higher proportion of all shortest paths between these two nodes, its significance is more notable [36,37]. CC estimates the closeness of each node to other nodes through the shortest path length [38]. EC mathematically deduces the convergence of the Markov chain to approximate the maximum eigenvector of the adjacency matrix to estimate graph centrality [39]. In short, every metric defines graph centrality from a different perspective. Only DC is locality in graph, whereas BC, CC, and EC are global metrics.

5. Results and discussion

5.1. Overall accuracies

Table 3 presents the results of the experiment conducted on the MNIST dataset; it shows that the C-CNN with DC has the highest accuracy compared with the classical CNN and other GCN models, and the C-GCN with BC shows state-of-the-art results. However, the regular graphs obtained using MNIST only verifies the feasibility of our models, because the accuracies of both our models and other GCNs are close. Hence, to demonstrate the availability of our models, we obtained irregular graphs using 20NEWS, Reuters8, and Reuters52 datasets, and the results are listed in Table 4. Our C-CNN and C-GCN models show impressive performances compared to the traditional models and GCN models in the same settings.

The regular graph designed by MNIST is similar to the image data obtained by CNNs; hence, the two variants of the separate encoding function aggregate the neighboring features to encode feature maps similar to the standard convolution of CNNs on images. In this experiment, the C-CNN is like a CNN without pooling layers;

Table 3
Comparison of our models, C-CNN and C-GCN, with other models using the MNIST dataset.

Model	Framework	Accuracy
Classical CNN	C32-P4-C64-P4-FC512	0.9933 ± 0.0015
A Generalization of CNN [58]	C20-FC512	0.9855 ± 0.0021
LRF [20]	400-LRF3200-MP800-LRF800-MP400-10	0.9870 ± 0.0036
ChebyNet [21]	GC32-P4-GC64-P4-FC512	0.9913 ± 0.0008
MoNet [26]	No details	0.9920 ± 0.0011
Dynamic Filters [60]	C32-C64-FC1024	0.9840 ± 0.0029
GCN [23]	GC32-GC64-FC512	0.9921 ± 0.0017
Motif-GCN [51]	GC32-GC64-FC512	0.9925 ± 0.0008
GAT [32]	No details	0.9931 ± 0.0004
PPNP [28]	GC32-GC64-FC512	0.9924 ± 0.0010
APPNP [28]	GC32-GC64-FC512	0.9929 ± 0.0012
Our C-CNN	**GC32:676-GC64:576-FC512**	**0.9934 ± 0.0006**
Our C-GCN	GC32:676-GC64:576-FC512	0.9928 ± 0.0010

Table 4
Comparison of our models with traditional models and GCNs using 20NEWS, Reuters8, and Reuters52 datasets.

Model	20NEWS	Reuters8	Reuters52
Linear SVM	0.6590 ± 0.0013	0.8121 ± 0.0002	0.7580 ± 0.0004
Multi-Naïve Bayes	0.6851 ± 0.0021	0.8356 ± 0.0005	0.7720 ± 0.0008
Softmax	0.6628 ± 0.0019	0.8026 ± 0.0007	0.7700 ± 0.0004
FC2500	0.6464 ± 0.0008	0.7990 ± 0.0012	0.7412 ± 0.0015
FC2500-FC500	0.6576 ± 0.0010	0.8041 ± 0.0010	0.7620 ± 0.0012
ChebyNet [21]	0.6826 ± 0.0015	0.8423 ± 0.0006	0.7812 ± 0.0008
MoNet [26]	0.7060 ± 0.0006	0.8687 ± 0.0003	0.8018 ± 0.0011
GCN [23]	0.7101 ± 0.0012	0.8821 ± 0.0005	0.8125 ± 0.0007
DCNN [55]	0.7035 ± 0.0022	0.8746 ± 0.0015	0.8242 ± 0.0012
DSGC [62]	0.7188 ± 0.0032	0.9032 ± 0.0010	0.8619 ± 0.0009
Text-GCN [52]	0.8012 ± 0.0015	0.9604 ± 0.0012	0.9302 ± 0.0012
Motif-GCN [51]	0.7314 ± 0.0027	0.9350 ± 0.0003	0.8810 ± 0.0005
GAT [32]	0.7383 ± 0.0016	0.9426 ± 0.0006	0.9143 ± 0.0008
PPNP [28]	0.7510 ± 0.0036	0.9556 ± 0.0010	0.9203 ± 0.0014
APPNP [28]	0.7462 ± 0.0031	0.9316 ± 0.0012	0.9080 ± 0.0010
Our C-CNN	**0.8259 ± 0.0017**	**0.9686 ± 0.0014**	**0.9342 ± 0.0008**
Our C-GCN	0.7946 ± 0.0028	0.9538 ± 0.0006	0.9224 ± 0.0005

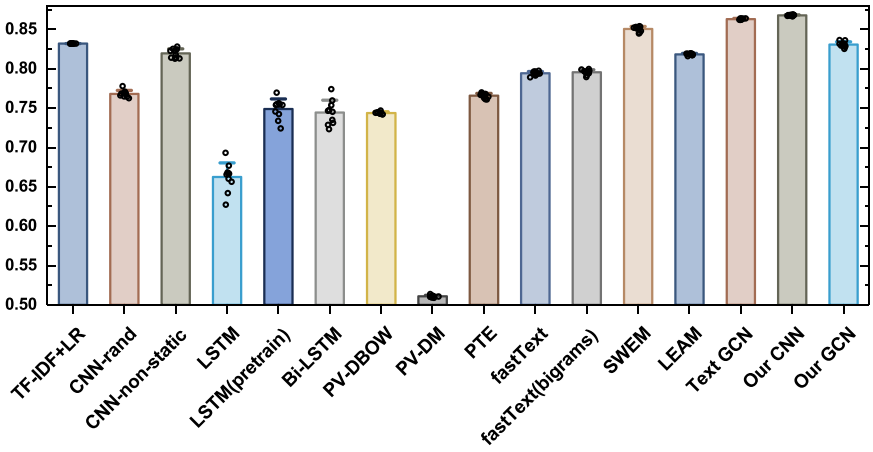


Fig. 4. Comparison of our C-CNN and C-GCN with other popular text classification models using 20NEWS. The input features of all models are based on 60000 high-frequency words.

that is, it is more like a generalization of a CNN on graphs. Furthermore, in the C-GCN, not only the separate encoding of the C-CNN is present but also the normalized graph Laplacian is integrated. In the experiments using 20NEWS, Reuters8, and Reuters52, the C-CNN yields state-of-the-art results. We deduce that the weight sharing in the separate encoding function plays a particular role in promoting performance because it reduces weights and learns local correlation like the standard convolution on images. These experiments not only further illustrates that the separate encoding function is a general extension of the standard convolution in a CNN but also demonstrates that their frameworks are the similarity.

To show the usability of our models, we compare the C-CNN and C-GCN with other popular text classification models, which are baselines such as TF-IDF+LR, CNN-rand [74], CNN-non-static [74], LSTM [75], Bi-LSTM, PV-DBOW [76], PV-DM [76], PTE [77], fastText [78], SWEM [79], LEAM [80], Text-CNN [52]. Fig. 4 presents the results for the same experimental settings with 20NEWS; it is observed that the proposed C-CNN has the highest accuracy.

5.2. Graph centrality metrics

Table 5 shows the comparison of the accuracies of our models by using different graph centrality metrics on the regular graph (MNIST) and the irregular graph (20NEWS). To build the words graph, 5000 high-frequency words from 20NEWS corpus were taken. The following can be observed: (a) In the MNIST experiment, accuracies of the C-CNN and C-GCN are not greatly different, with the C-CNN performing only slightly better than the C-GCN. (b) In the 20NEWS experiment, the accuracy of the C-CNN is considerably higher than that of the C-GCN. (c) From the perspective of graph centrality metrics, both variants show higher performances in DC and BC than in CC and EC.

We observe that different graph centrality metrics influence the performance of the models differently. From Table 5, the two variants perform better in DC. This is because the definition of DC represents local correlation more compared with other graph centrality metrics. Specifically, the importance of a node in DC is only affected by the node's neighbors (local correlation), but the importance of a node in other metrics are based on the entire graph structure (global correlation). The separate encoding function based on DC is more in line with the standard convolution of CNNs from the perspective of interpretability, because both methods use weight sharing to learn the local correlation.

5.3. Effect of graph structure and scale

The hyper-parameter k-NN was chosen on the basis of paper [21], in which 8-NN was used to construct the regular graph with the MNIST dataset, and 16-NN was used to build the irregular graph with the 20NEWS dataset. Fig. 5 shows the effect of increasing doubly k on the accuracy of our models. Note that the accuracies of the two variants in DC increase with increasing k-NN, but their growth rate gradually decreases with further increase in k.

Table 5
Accuracies of C-CNN and C-GCN for different graph centrality metrics tested using MNIST and 20NEWS datasets.

Dataset	Model	***k*-NN**	DC	BC	CC	EC
MNIST	C-CNN	24	0.9934 ± 0.0006	0.9925 ± 0.0009	0.9928 ± 0.0013	0.9929 ± 0.0011
	C-GCN	24	0.9927 ± 0.0007	0.9928 ± 0.0010	0.9927 ± 0.0012	0.9924 ± 0.0013
20NEWS	C-CNN	16	0.7965 ± 0.0010	0.7945 ± 0.0015	0.7713 ± 0.0013	0.7737 ± 0.0009
	C-GCN	16	0.7625 ± 0.0026	0.7671 ± 0.0021	0.7439 ± 0.0019	0.7262 ± 0.0032

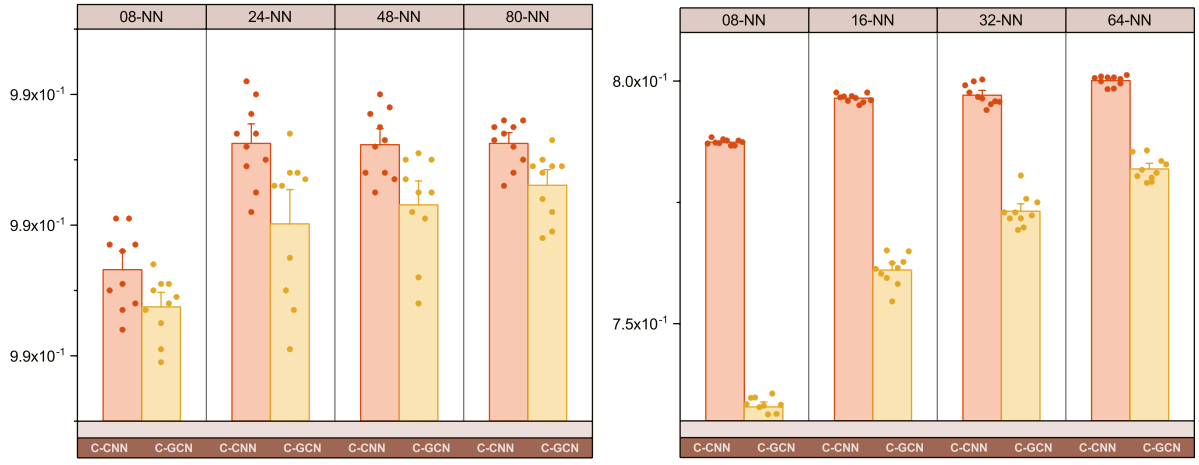


Fig. 5. Accuracies of both C-CNN and C-GCN in DC with increasing *k*-NN using MNIST (left) and 20NEWS (right).

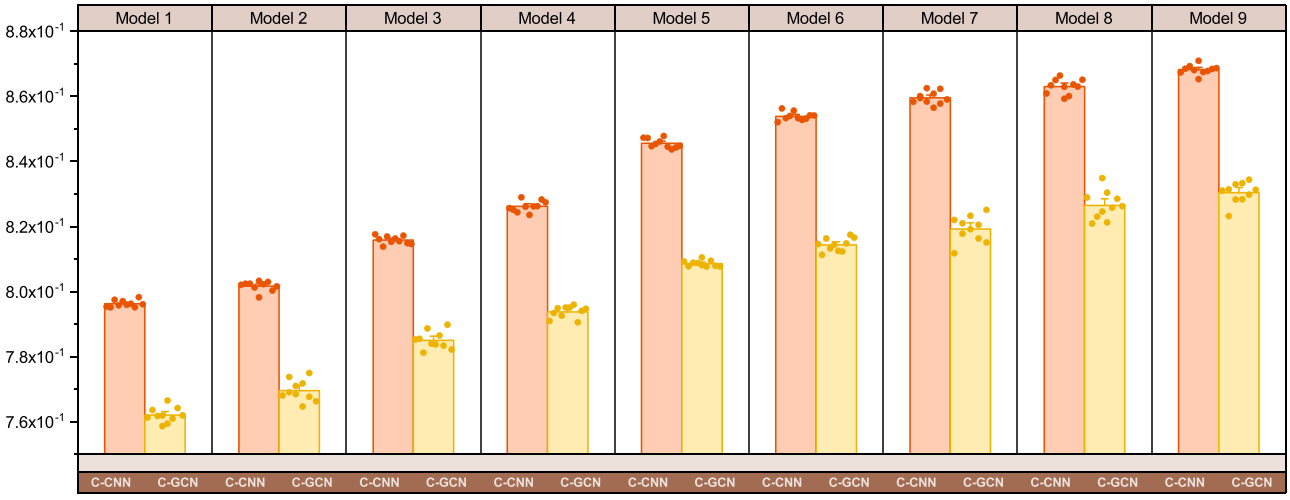


Fig. 6. Growth trends of accuracy and distribution details from Table 6.

This result implies that a larger k can promote the separate encoding of a broader range of neighborhood features, but an excessively large k limits this effect.

The effect of the graph scale on 20NEWS is presented in Table 6. As the number of high-frequency words and the size $|F^{(l)}|$ of feature map increase, the accuracies of our models improve gradually. However, the performance of the C-GCN is still considerably lower than the accuracy of the C-CNN. Fig. 6 shows the growth trends of accuracy and distribution details from Table 6; we observe that the performance of the C-CNN is consistently better than that of the C-GCN. The growth of the graph structure and scale nonlinearly increase the performance of our models. As Table 6 shows, the performances of the two variants gradually increase nonlinearly with increasing number of nodes (words) and changing structure of the word graph. Changes in the graph structure and scale significantly impact the performance of the two variants, and this relationship deserves further study in the future.

Table 6
Accuracies of the C-CNN and C-GCN using 20NEWS with increasing number of top high-frequency words and size of the feature map.

Model ID	Framework	Words	C-CNN	C-GCN
Model 1	GC32:1024-FC512	5000	0.7965 ± 0.0010	0.7625 ± 0.0026
Model 2	GC32:2048-FC512	5000	0.8023 ± 0.0017	0.7697 ± 0.0029
Model 3	GC32:2048-FC512	10000	0.8158 ± 0.0016	0.7840 ± 0.0025
Model 4	GC32:4096-FC512	10000	0.8259 ± 0.0022	0.7946 ± 0.0020
Model 5	GC32:10240-FC512	20000	0.8457 ± 0.0018	0.8085 ± 0.0019
Model 6	GC32:10240-FC512	30000	0.8543 ± 0.0020	0.8151 ± 0.0029
Model 7	GC32:20480-FC512	40000	0.8596 ± 0.0021	0.8204 ± 0.0031
Model 8	GC32:20480-FC512	50000	0.8631 ± 0.0019	0.8268 ± 0.0037
Model 9	GC32:25600-FC512	60000	0.8679 ± 0.0011	0.8306 ± 0.0033

5.4. Resource consumption

In addition to the accuracy experiments, we performed resource consumption experiments. We compared our two variants with

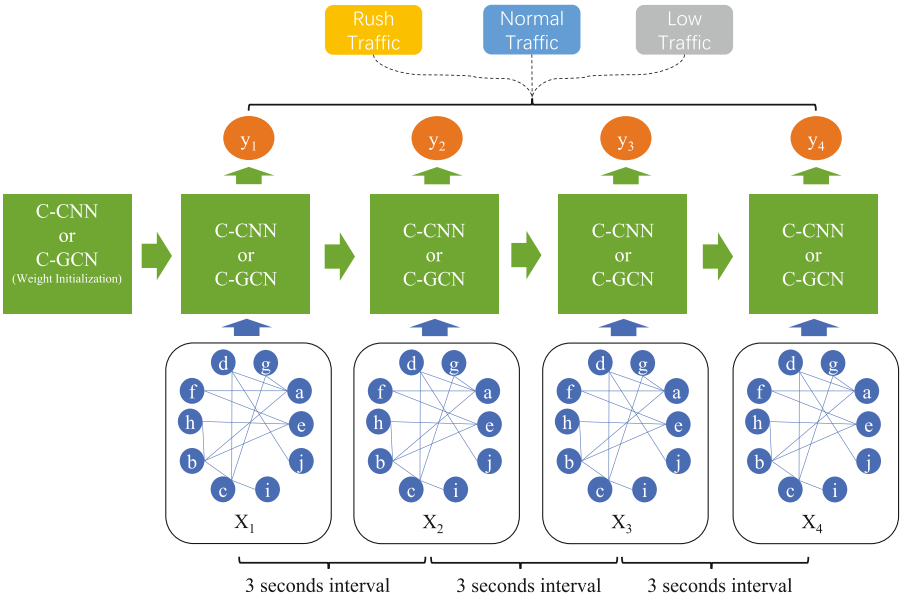


Fig. 7. Two variants combined with the RNN to predict the city-wide traffic congestion—Rush traffic (traffic jam), low traffic (little traffic, usually at night), and normal traffic (regular traffic with no congestion).

Table 7
Comparison of resource consumption performances of our model with other models based on Tensorflow using the MNIST dataset.

Model	Time(sec)	GPU Memory(GB)
Classical CNN	22	0.8
ChebyNet [21]	82	1.0
MoNet [26]	116	0.9
GCN [23]	35	1.2
DCNN [55]	81	1.2
DSGC [62]	104	1.1
Motif-GCN [51]	37	1.3
GAT [32]	130	0.9
PPNP [28]	36	1.1
APPNP [28]	31	1.0
Our C-CNN	78	1.1
Our C-GCN	96	1.2

only baseline models based on Tensorflow, which helps in managing and scheduling CPU, GPU, and memory in a unified way. Other baseline models discussed in Section 4 that are not currently based on Tensorflow were not used in the comparison with our models. Based on Table 7, we can conclude the following: (a) The classical CNN performs the best because its framework is optimized by Tensorflow. (b) The GCN [23], motif-GCN [51], PPNP [28], and APPNP [28] show similar performance in terms of resource consumption, because their frameworks are similar, all derived from the GCN [23]. (c) Because the MoNet [26] and GAT [32] have homologous architecture, their resource consumption is very close. (d) Our two variants show average-level performance. This is due to the resource-intensive function "tf.nn.embedding_lookup" of Tensorflow, which is used in the code for the separate encoding function. Although the classical CNN also uses this function, Tensorflow has encapsulated it at the lowest code-level, therefore imparting more efficiency. Other models do not use this function, so they are more efficient than our two variants.

5.5. Traffic congestion recognition

To further verify the practicability of our models, the C-CNN and C-GCN combined with the RNN were applied to predict city-wide traffic congestion. The many-to-many RNN was introduced because it can adequately handle trajectory data that have temporality-sequence correlation. Fig. 7 shows the combination of our models and many-to-many RNN, which has the same length of input and output sequences, and the time intervals correspond to each other. In our work, a road network of a city is abstracted to a road graph, with each node being a road and each edge being an intersection of two roads. The node feature is the number of GPS points that appear in this node (road) within a time interval; here the time interval was three seconds. The GPS points were collected per three seconds in the trajectory data of Xi'an and Chengdu, which have been opened to public by the Didi Chuxing GAIA Initiative. Fig. 8 shows trajectory data, and Table 8 shows the data fields of trajectory data. The input of our model was a road graph containing node features; our dataset comprised 1,756,800 chronological road graph data in a city within a span of two consecutive months. The road graph data generated in the first week were selected as the training set (approximately 10% of the total trajectory data), whereas the remaining were used as the test set. The road graph shows three states, which are the output of our model: rush traffic (traffic jam), low traffic (with little traffic, usually at night), and normal traffic (regular traffic with no congestion). These outputs are used as labels in the road graph classification task. Table 9 presents the results of the experiment conducted with the neural network (NN) and the GCN [23] as the baseline methods for comparison with our two variants combined with the RNN. Note that the C-CNN shows the best performance. The experimental settings were follows: the learning rate was 0.001, decay of learning rate was 0.95 per epoch, mini-batch size was 300, number of epochs was 30, framework was GC32:8096-FC:512, graph

Table 8
Fields of trajectory data.

Field	Type	Sample	Comment
Driver ID	String	glox.jrr	Anonymous
Order ID	String	jkkt8kxn	Anonymous
Time Stamp	String	1,501,584,540	Unix timestamp, per 3 s
Longitude of GPS	String	104.04392	GCJ-02 Coordinate System
Latitude of GPS	String	104.04392	GCJ-02 Coordinate System

Table 9
The test accuracies of neural network (NN), GCN [23], and our two variants with RNN structure.

City	Items	Labeled Rate	NN	**GCN** [23]	C-CNN	C-GCN
Xi'an	1756800	10.36%	0.8432 ± 0.0014	0.9296 ± 0.0031	**0.9557** ± **0.0022**	0.9521 ± 0.0031
Chengdu	1756800	10.42%	0.8369 ± 0.0023	0.9265 ± 0.0027	**0.9521** ± **0.0024**	0.9479 ± 0.0026

Fig. 8. Left: Trajectory data for Xi'an. Right: Trajectory data for Chengdu.

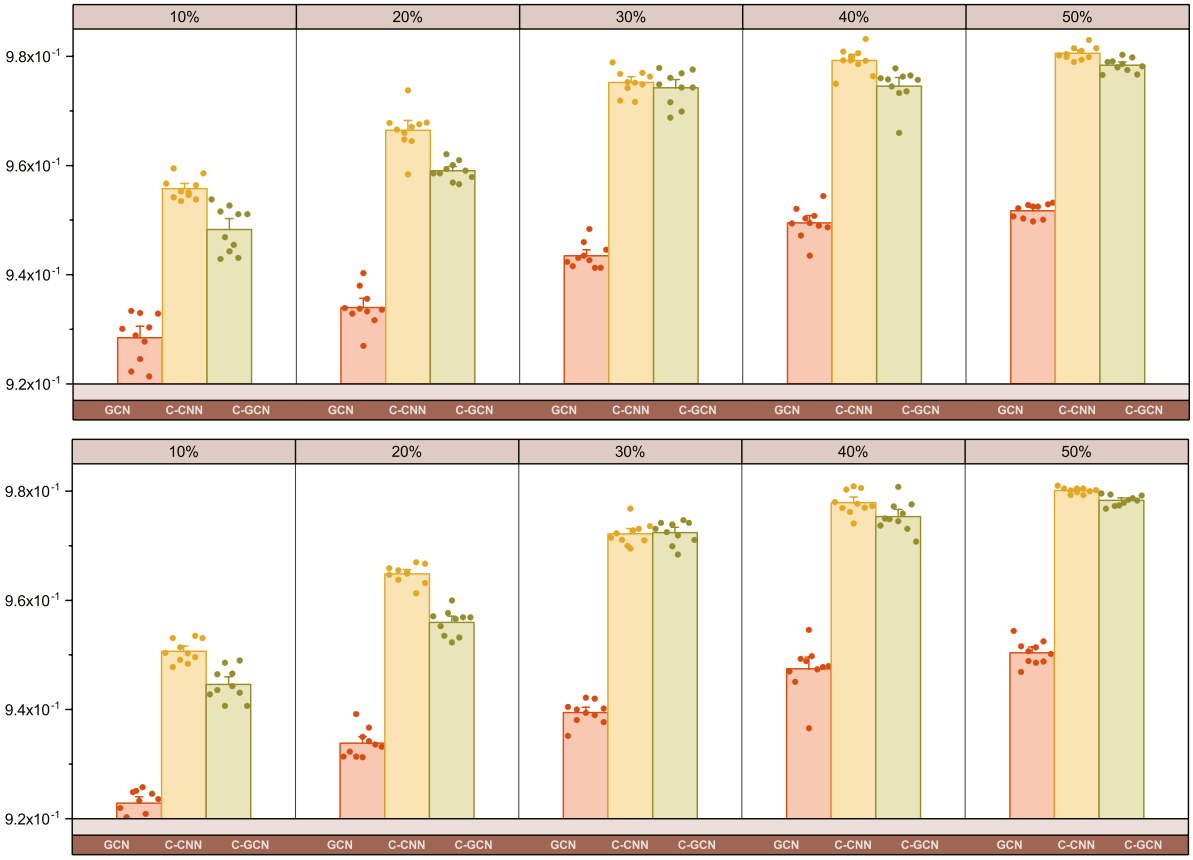


Fig. 9. Accuracies of the GCN [23], C-CNN, and C-GCN with increasing proportion of the training set for Xiâ an (top) and Chengdu (bottom).

centrality was DC, dropout for FC was 0.5, and RNN sequence length was 50. Fig. 9 shows that the accuracies of the GCN [23], C-CNN, and C-GCN increase when the proportion of the training set increases, and our C-CNN has the highest accuracy.

GCNs can replace other models for traffic congestion recognition because GCNs, as well as the two variants, have two advantages over traditional models: (a) GCNs integrate prior spatial information from a road graph, which is not possible in traditional models. The road graph is designed to structure the spatial information that is embedded in the uniform encoding function as well as our separate encoding function. Moreover, the RNN combines with GCNs to process this spatiotemporal information of the GPS. (b) GCNs can compress a large number of GPS points into the road graph data to reduce the structure size of the network, whereas classical CNNs cannot. Moreover, our two variants based on the separate encoding function further reduces model cost compared with GCN [23] for traffic congestion recognition. Hence, they are more advantageous to process massive road graphs such as those of Beijing and Shanghai than GCN [23] in future work.

6. Conclusion and future works

We presented two variants, C-CNN, and C-GCN, of our proposed separate encoding function, in which graph centrality is introduced for weight sharing and enhancing performance. The proposed models exhibited impressive performances on regular and irregular graphs in comparison with previous GCN methods. We also applied the two variants on online car-hailing service trajectory data, and the performance of the C-CNN was the state-of-the-art compared to that of the GCN [23]. In future studies, we will explore several potential improvements and extensions to our separate encoding function. The following states the scope for improvement of our model.

Directed graph: Our model is limited to undirected graphs. Although directed graphs can be transformed to undirected graphs to fit our separate encoding function, the directed information may be lost in this process, which will result in building of the model on incomplete prior information. Application of our separate encoding function for directed graphs is a focus of improvement.

Limited memory: The GCN [23] and the two variants are fed to the entire graph and built as a vast framework unlike traditional CNNs. The GPU memory cannot store the framework and graph data, while the CPU memory is not limited by this problem. However, the CPU performance is far less than the GPU in large-scale floating-point operations. Fortunately, our separate encoding function is an extension of a standard convolution of CNNs; therefore, to tailor the framework of the separate encoding function and to make the two variants less memory-intensive, we can import the depthwise separable convolution (DSC) of MobileNet [42] and MobileNetV2 [43] in our future work.

Separate encoding visualization: For the application of the C-CNN and C-GCN in graph classification tasks, in future, we will consider a visualization method for the separate encoding function based on the visualization method for CNNs proposed by MD Zeiler and R Fergus [81].

Declaration of Competing Interest

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

Acknowledgments

First, we thank the reviewers for their feedback and helpful suggestions. Secondly, thanks to the online car-hailing service trajectory data of Xi'an and Chengdu in China, which are provided by Didi Chuxing GAIA Initiative from Didi company. Finally, we are grateful for the support of following projects: (1) Grant No. 2017YFC0804103, Chinese 13th Five-Year Key Research and Development Program. (2) Grant No. 61761042, National Natural Science Foundation of China. (3) Grant No. 61941112, National Natural Science Foundation of China. (4) Grant No. 2019GY-020, Key R & D projects of Shaanxi Province. (5) Grant No. 300102238103, Fundamental Research Funds for the Central Universities.

References

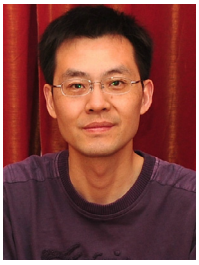
[1] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al., Relational inductive biases, deep learning, and graph networks, arXiv:1806.01261 (2018).
[2] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, M. Sun, Graph neural networks: A review of methods and applications, arXiv:1812.08434 (2018).
[3] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P.S. Yu, A comprehensive survey on graph neural networks, arXiv:1901.00596 (2019).
[4] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Proceedings of the Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[5] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: International Conference on Learning Representations, 2015.
[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.
[7] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[8] H. Geoffrey, D. Li, Y. Dong, E.D. George, A.-r. Mohamed, Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, IEEE Signal Process. Mag. 29 (6) (2012) 82–97.
[9] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, et al., Deep speech 2: End-to-end speech recognition in english and mandarin, in: Proceedings of the International Conference on Machine Learning, 2016, pp. 173–182.
[10] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, G. Zweig, Achieving human parity in conversational speech recognition, Tech. Rep. MSR-TR-2016-71 (revised) (February 2017). URL https://www.microsoft.com/en-us/research/publication/achieving-human-parity-conversational-speech-recognition-2/.
[11] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019, pp. 4171–4186.
[12] S. Yan, Y. Xiong, D. Lin, Spatial temporal graph convolutional networks for skeleton-based action recognition, in: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
[13] Z. Zhang, P. Cui, W. Zhu, Deep learning on graphs: A survey, arXiv:1812.04202 (2018).
[14] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, Network Flows (1988).
[15] M.S. Daskin, Network and Discrete Location: Models, Algorithms, And Applications, John Wiley & Sons, 2011.
[16] M. McPherson, L. Smith-Lovin, J.M. Cook, Birds of a feather: Homophily in social networks, Ann. Rev. Sociol. 27 (1) (2001) 415–444.
[17] M. Girvan, M.E. Newman, Community structure in social and biological networks, Proc. Natl. Acad. Sci. 99 (12) (2002) 7821–7826.
[18] M. Kanehisa, S. Goto, M. Furumichi, M. Tanabe, M. Hirakawa, KEGG for representation and analysis of molecular networks involving diseases and drugs, Nucleic Acids Res. 38 (suppl_1) (2009) D355–D360.
[19] V. Spirin, L.A. Mirny, Protein complexes and functional modules in molecular networks, Proc. Natl. Acad. Sci. 100 (21) (2003) 12123–12128.
[20] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, in: International Conference on Learning Representations (ICLR2014), CBLS, 2014.
[21] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Proceedings of the Advances in Neural Information Processing Systems, 2016, pp. 3844–3852.
[22] S. Kearnes, K. McCloskey, M. Berndl, V. Pande, P. Riley, Molecular graph convolutions: moving beyond fingerprints, J. Comput. Aided Molecular Des. 30 (8) (2016) 595–608.
[23] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: International Conference on Learning Representations, 2017.
[24] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Proceedings of the Advances in Neural Information Processing Systems, 2017, pp. 1025–1035.
[25] T. Pham, T. Tran, D. Phung, S. Venkatesh, Column networks for collective classification, in: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, 2017.
[26] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, M.M. Bronstein, Geometric deep learning on graphs and manifolds using mixture model CNNS, in: Proceedings of the CVPR, 1, 2017, p. 3.
[27] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 1263–1272.
[28] J. Klicpera, A. Bojchevski, S. Günnemann, Predict then propagate: Graph neural networks meet personalized pagerank (2018).
[29] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Netw. 20 (1) (2008) 61–80.
[30] Y. Li, D. Tarlow, M. Brockschmidt, R. Zemel, Gated graph sequence neural networks, in: International Conference on Learning Representations, 2016.
[31] H. Dai, Z. Kozareva, B. Dai, A. Smola, L. Song, Learning steady-states of iterative algorithms over graphs, in: Proceedings of the International Conference on Machine Learning, 2018, pp. 1114–1122.
[32] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph attention networks, in: International Conference on Learning Representations, 2018.
[33] Q. Li, Z. Cao, J. Zhong, Q. Li, Graph representation learning with encoding edges, Neurocomputing 361 (2019) 29–39.
[34] Q. Li, J. Zhong, Z. Cao, X. Li, Optimizing streaming graph partitioning via a heuristic greedy method and caching strategy, Optim. Methods Softw. (2019) 1–16.
[35] Q. Li, J. Zhong, Q. Li, C. Wang, Z. Cao, A community merger of optimization algorithm to extract overlapping communities in networks, IEEE Access 7 (2018) 3994–4005.

[36] L.C. Freeman, A set of measures of centrality based on betweenness, Sociometry (1977) 35–41.
[37] R. Carston, Herbert h. clark, using language. cambridge: Cambridge university press, 1996. pp. xi+ 432.-, J. Linguist. 35 (1) (1999) 167–222.
[38] D. Krackhardt, Assessing the political landscape: Structure, cognition, and power in organizations, Administ. Sci. Quarterly (1990) 342–369.
[39] M. Kitsak, L. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H. Stanley, H. Makse, Identifying influential spreaders in complex networks, Nat. Phys. 6 (2010) 888–893.
[40] D. Das, C.G. Lee, Unsupervised domain adaptation using regularized hyper-graph matching, in: Proceedings of the 25th IEEE International Conference on Image Processing (ICIP), IEEE, 2018, pp. 3758–3762.
[41] M. Cao, H. Zhang, J. Park, N.M. Daniels, M.E. Crovella, L.J. Cowen, B. Hescott, Going the distance for protein function prediction: a new distance metric for protein interaction networks, PloS One 8 (10) (2013) e76339.
[42] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017.
[43] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.
[44] P. Frasconi, M. Gori, A. Sperduti, A general framework for adaptive processing of data structures, IEEE Trans. Neural Netw. 9 (5) (1998) 768–786.
[45] A. Sperduti, A. Starita, Supervised neural networks for the classification of structures, IEEE Trans. Neural Netw. 8 (3) (1997) 714–735.
[46] M. Gori, G. Monfardini, F. Scarselli, A new model for learning in graph domains, in: Proceedings of the IEEE International Joint Conference on Neural Networks IJCNN'05, 2, IEEE, 2005, pp. 729–734.
[47] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Netw. 20 (1) (2009) 61–80.
[48] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, in: Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014.
[49] M. Henaff, J. Bruna, Y. LeCun, Deep convolutional networks on graph-structured data, in: Conference and Workshop on Neural Information Processing Systems, 2015.
[50] Q. Li, Z. Han, X.-M. Wu, Deeper insights into graph convolutional networks for semi-supervised learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2018.
[51] J.B. Lee, R.A. Rossi, X. Kong, S. Kim, E. Koh, A. Rao, Higher-order graph convolutional networks, arXiv:1809.07697 (2018).
[52] L. Yao, C. Mao, Y. Luo, Graph convolutional networks for text classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 7370–7377.
[53] Z. Wang, Z. Kong, S. Changra, H. Tao, L. Khan, Robust high dimensional stream classification with novel class detection, in: Proceedings of the IEEE 35th International Conference on Data Engineering (ICDE), IEEE, 2019, pp. 1418–1429.
[54] D.K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, R.P. Adams, Convolutional networks on graphs for learning molecular fingerprints, in: Proceedings of the Advances in Neural Information Processing Systems, 2015, pp. 2224–2232.
[55] J. Atwood, D. Towsley, Diffusion-convolutional neural networks, in: Proceedings of the Advances in Neural Information Processing Systems, 2016, pp. 1993–2001.
[56] M. Niepert, M. Ahmed, K. Kutzkov, Learning convolutional neural networks for graphs, in: Proceedings of the International Conference on Machine Learning, 2016, pp. 2014–2023.
[57] J. Chen, J. Zhu, L. Song, Stochastic training of graph convolutional networks with variance reduction, in: International Conference on Machine Learning, 2018, pp. 941–949.
[58] Y. Hechtlinger, P. Chakravarti, J. Qin, A generalization of convolutional neural networks to graph-structured data, Stat 1050 (2017) 26.
[59] Z. Zhou, X. Li, Graph convolution: a high-order and adaptive approach, arXiv preprint arXiv:1706.09916.
[60] N. Verma, E. Boyer, J. Verbeek, Dynamic filters in graph convolutional networks, arXiv:1706.05206 (2017).
[61] M. Chen, Z. Lin, K. Cho, Graph convolutional networks for classification with a structured label space, arXiv:1710.04908 (2017).
[62] G. Lai, H. Liu, Y. Yang, Learning graph convolution filters from data manifold, arXiv:1710.11577 (2017).
[63] B. Wu, Y. Liu, B. Lang, L. Huang, Dgcnn: Disordered graph convolutional neural network based on the gaussian mixture model, Neurocomputing 321 (2018) 346–356.
[64] U.S. Shanthamallu, J.J. Thiagarajan, A. Spanias, Improving robustness of attention models on graphs, arXiv:1811.00181 (2018).
[65] F.B. Zhan, C.E. Noon, Shortest path algorithms: an evaluation using real road networks, Transp. Sci. 32 (1) (1998) 65–73.
[66] D. Kushnir, M. Galun, A. Brandt, Fast multiscale clustering and manifold identification, Pattern Recogn. 39 (10) (2006) 1876–1891.
[67] I.S. Dhillon, Y. Guan, B. Kulis, Weighted graph cuts without eigenvectors a multilevel approach, IEEE Trans. Pattern Anal. Mach. Intell. 29 (11) (2007).
[68] G. Karypis, V. Kumar, Metis–unstructured graph partitioning and sparse matrix ordering system, version 2.0 (1995).
[69] U. Von Luxburg, A tutorial on spectral clustering, Statist. Comput. 17 (4) (2007) 395–416.
[70] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[71] A. Sokolov, S. Riezler, Task-driven greedy learning of feature hashing functions, in: Proceedings of the NIPS, 13, 2013, pp. 1–5.
[72] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, in: International Conference on Learning Representations, 2013.
[73] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: International Conference on Learning Representations, 2015.
[74] Y. Kim, Convolutional neural networks for sentence classification, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1746–1751.
[75] P. Liu, X. Qiu, X. Huang, Recurrent Neural Network for Text Classification with Multi- Task Learning, AAAI Press, 2016, pp. 2873–2879.
[76] Q. Le, T. Mikolov, Distributed representations of sentences and documents, in: Proceedings of the International Conference on Machine Learning, 2014, pp. 1188–1196.
[77] J. Tang, M. Qu, Q. Mei, Pte: Predictive text embedding through large-scale heterogeneous text networks, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2015, pp. 1165–1174.
[78] A. Joulin, E. Grave, P.B.T. Mikolov, Bag of tricks for efficient text classification, EACL 2017 (2017) 427.
[79] D. Shen, G. Wang, W. Wang, M. R. Min, Q. Su, Y. Zhang, C. Li, R. Henao, L. Carin, Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2018, pp. 440–450.
[80] G. Wang, C. Li, W. Wang, Y. Zhang, D. Shen, X. Zhang, R. Henao, L. Carin, Joint embedding of words and labels for text classification, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2018, pp. 2321–2331.
[81] M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: Proceedings of the European Conference on Computer Vision, Springer, 2014, pp. 818–833.

Wei Dong is a Ph.D. student with the School of Computer Science and Engineering. The University of Northwestern Polytechnical University. He received the MS degree in Xi'an University of Architecture and Technology 2014. He is currently pursuing the Ph.D. degree with the School of Computer Science, Northwestern Polytechnical University, Xi'an. His research interests cover computer vision, graph neural network and deep learning.

Junsheng Wu is currently a doctoral supervisor in the School of Computer Science and Engineering of Northwest Polytechnical University and a professor in the School of Software of Northwest Polytechnical University. His research interests cover domain-oriented software engineering technology and application. He is engaged in research and development, testing and validation of manufacturing enterprise application software system, airborne data acquisition and integrated processing system, computing visualization technology and its application system.

Zongwen Bai is with the Shaanxi Provincial Key Lab of bigdata of energy and intelligence processing, School of physics and electronic information, Yanan University, Yanan 716000, China. He received the MS degree in Yanan university 2008. He is currently pursuing the Ph.D. degree with the School of Computer Science, Northwestern Polytechnical University, Xi'an. He is a joint PhD student with the School of Computer Science and Engineering, Northwestern Polytechnical University. His research interests cover computer vision, nature language processing and deep learning. This work was supported by the National Natural Science Foundation of China (Grant No.61761042), the Key Research and Development Program of Yanan (Grant No. 2017KG-01,2017WZZ-04-01).



Weigang Li received the Ph.D. degrees in Manufacturing Engineering from Northwestern Polytechnical University, Xi'an, China in 2003. In 2007, he joined the faculty of Software Engineering of the Northwestern Polytechnical University (NPU). He is working as an associate professor of software engineering in the School of Software at the NPU. His research interests include cloud computing, big data processing for enterprise applications. He has successfully delivered many IT projects in China since 2004 and published over 30 peer-reviewed papers.

Wei Qiao works in Xi'an Research Institute Co., Ltd., China Coal Technology and Engineering Group, Shaanxi Key Laboratory of Coal Mine Water Hazard Prevention and Control Technology, Shaanxi Xi'an 710077. He is a joint Ph.D. student with the School of Computer Science and Engineering, Northwestern Polytechnical University. He received the MS degree in Xi'an University of Architecture and Technology 2014. He is currently pursuing the Ph.D. degree with the School of Computer Science, Northwestern Polytechnical University, Xi'an. His research interests cover big data processing, cloud computing and deep learning.