Applied Intelligence (2022) 52:4776–4791
https://doi.org/10.1007/s10489-021-02617-7

Simplified multilayer graph convolutional networks with dropout

Fei Yang[1] · Huyin Zhang[1,2] · Shiming Tao[1]

Accepted: 2 June 2021 / Published online: 28 July 2021

Abstract
Graph convolutional networks (GCNs) and their variants are excellent deep learning methods for graph-structured data. Moreover, multilayer GCNs can perform feature smoothing repeatedly, which creates considerable performance improvements. However, they may inherit unnecessary complexity and redundant computation; to make matters worse, they introduce overfitting as the number of layers increases. In this paper, we present simplified multilayer graph convolutional networks with dropout (DGCs), novel neural network architectures that successively perform nonlinearity removal and weight matrix merging between graph conventional layers, leveraging a dropout layer to achieve feature augmentation and effectively reduce overfitting. Under such circumstances, first, we extend a shallow GCN to a multilayer GCN. Then, we reduce the complexity and redundant calculations of the multilayer GCN, while improving its classification performance. Finally, we make DGCs readily applicable to inductive and transductive tasks. Extensive experiments on citation networks and social networks offer evidence that the proposed model matches or outperforms state-of-the-art methods.

Keywords Graph convolutional networks · Multilayer · Dropout · Feature augmentation

1 Introduction

Graph neural networks (GNNs) that directly deal with graphs were previously introduced in [1, 2] as a generalization of recursive neural networks. In [3], the eigendecomposition of the graph Laplacian was calculated by spectral graph convolutions in the Fourier domain, leading to non-spatial local filtering effects. This idea was adopted and improved by [4], which introduces fast localized convolutions to make them spatially localized. Recently, Defferrard et al. [5] applied Chebyshev polynomials to approximate convolution kernel filters for dealing with graph-structured data. Then, Kipf and Welling [6] proposed GCNs, which utilize the first-order Chebyshev approximation to construct a hierarchical linear model for fast and accurate semi-supervised node classification. To learn the graph representations, GCNs apply the same linear transformation to the neighborhood of a node followed by a nonlinear activation function. In recent years, GCNs and subsequent variants have obtained excellent results in many fields, including but not limited to citation networks [6, 7], social networks [8, 9], traffic forecasting [10–12], natural language processing [13–15] and computer vision [16].

However, GCNs are mainly extended from recent deep learning methods, so they may inherit unnecessary complexity and redundant calculations. Previous work used sampling to reduce the number of calculations. For instance, Hamilton et al. [17] introduced GraphSAGE, a method for calculating node representations through inductive sampling and aggregation. Chen et al. [18] presented FastGCN based on importance sampling to reduce the number of calculations. Recently, Thekumparampil et al. [19] empirically showed that graph linear network (GLN) can perform competitively. Wu and Zhang [7] proposed simple graph convolution (SGC), which successively performs nonlinear removal and weight matrix merging between graph conventional layers to reduce redundant computations. However,

✉ Huyin Zhang
zhy2536@whu.edu.cn

Fei Yang
young_fly@whu.edu.cn

Shiming Tao
tsm2783@whu.edu.cn

1 School of Computer Science, Wuhan University, Wuhan, China

2 Key Laboratory of Urban Land Resources Monitoring and Simulation, Ministry of Natural Resources, Shenzhen, China

in all experiments of [7], two-layer SGC was used. Similarly, the models mentioned above are all shallow models (usually 2 layers), which limits their ability to extract information from higher-order neighbors. Accordingly, our goal is to explore multilayer GCN, reduce its complexity and redundant calculations, and improve its classification performance.

In recent work, Li et al. [20] theoretically demonstrated that the graph convolution of GCN is a special form of Laplacian smoothing, which calculates the local average of features of a vertex and its nearby neighbors. The features of vertices in the same cluster become closer through a smoothing operation; thus, the classification task is greatly simplified. However, the smoothing operation also introduces potential oversmoothing problems. For a GCN model with multiple convolutional layers, the output features may become too similar, making it difficult to classify vertices in different categories. Moreover, this indistinguishable phenomenon soon appears on small data sets, even if the GCN model has very few convolutional layers. Generally, the more convolutional layers of the GCN, the more redundant calculations, and the more likely overfitting will occur. Therefore, we need to alleviate the overfitting problem introduced by the multilayer graph convolution.

To reduce the overfitting of deep neural networks, [21, 22] applied dropout to prevent complex coadaptations of training data. After using dropout regularization, each hidden unit in the network randomly stops working with a probability of 0.5. Therefore, the hidden units that work normally cannot be interdependent. Moreover, Bouthillier et al. [23] interpreted the use of dropout in the network layer as data augmentation and used it to learn better models. With the development of deep learning, previous work introduced dropout and its variants into deep learning, including but not limited to recurrent neural networks, convolutional neural networks, and graph convolutional networks. For instance, Zolna et al. [24] proposed fraternal dropout that takes advantage of dropout to optimize recurrent neural networks. Khan et al. [25] introduced spectral dropout to improve the generalization ability of deep convolutional neural networks (CNNs). Rong et al. [26] proposed DropEdge, a novel and flexible technique to alleviate overfitting and oversmoothing of graph conventional networks.

In this paper, we not only extend the shallow GCN model to the multilayer GCN model to extract information from higher-order neighbors, but also reduce redundant calculation and overfitting of the multilayer GCN to make it simple and efficient. We derive inspiration primarily from both simplified graph convolution and dropout regularization in deep learning. Therefore, we alleviate redundant computation by successively removing nonlinearities and merging weight matrices between graph conventional layers. Then, we add a simple and efficient dropout layer to achieve feature augmentation and effectively reduce overfitting. In the simplified multilayer model, the superimposed convolutional layer repeatedly performs Laplacian smoothing on the input features, and the dropout layer implements data augmentation for new features that have been Laplacian smoothed. We empirically show that the simplified multilayer model exhibits comparable or even better performance to GCN and its variants on various tasks, with higher computational efficiency and fewer fitting parameters. The simplified multilayer model is treated as DGC.

Many experiments on benchmark datasets for citation and social networks demonstrate that DGC compares favorably in classification accuracy against state-of-the-art methods. Simultaneously, it yields up to several orders of magnitude speedup over state-of-the-art methods, e.g., DGC outperforms FastGCN [18] by up to two orders of magnitude on our inductive tasks. In addition, on the downstream tasks, the performance of DGC also outperforms the latest methods. This demonstrates that applying the dropout layer to the simplified multilayer GCN model can significantly alleviate overfitting and improve classification accuracy.

Briefly, contributions of this paper are as follows:

- We propose a DGC model that first extends a shallow GCN to multilayer GCN and then eliminates the redundant calculations of the multilayer GCN. Under the premise of improving the classification ability of the model, the calculation speed of the model also increases by several orders of magnitude.
- We add a simple and effective dropout layer after feature extraction for feature augmentation since the dropout layer can effectively alleviate overfitting.
- Extensive experiments on citation networks and social networks demonstrate that our method can match or outperform state-of-the-art methods.

The rest of this paper is organized as follows. Section 2 reviews the related literature, and Section 3 gives a formal and detailed introduction to DGC. As presented in Section 4, extensive experiments conducted on citation networks and social networks show that DGC matches or outperforms several state-of-the-art methods. Section 5 summarizes the paper and outlines the direction of future work.

2 Related work

2.1 Graph neural networks

Recently, considerable efforts have been devoted to generalizing powerful CNNs to deal with graph-structured data. Bruna et al. [3] first proposed a spectral graph-based extension of convolutional networks to graphs. In follow-up work, ChebyNets [5] defined graph convolutions, which use Chebyshev polynomials to approximate convolution kernel filters. GCN [6] further simplified graph convolutions by applying the first-order approximation to develop a hierarchical linear model. Chen et al. [18] proposed an efficient GCN variant based on importance sampling. Hamilton et al. [17] introduced GraphSAGE, a method for calculating node representations through inductive sampling and aggregation. Wu and Zhang [7] presented a simple graph convolution (SGC), which successively removes nonlinearities and merges weight matrices between graph conventional layers to reduce redundant computation.

With the development of deep learning, an increasing number of GCN variants have been widely used in deep learning fields. Thekumparampil et al. [19] replaced the propagation layers with self-attentional layers to address the graph-structured data. Furthermore, Veličković et al. [27] presented a graph attention network (GAT), which utilizes masked self-attentional mechanisms to assign different weights to different nodes in a neighborhood. Abu-El-Haija et al. [28] proposed a new model MixHop that repeatedly mixes feature representations of neighbors at various distances. Qu et al. [29] proposed the graph Markov neural network (GMNN), which combines the advantages of both GCN and Markov. You et al. [30] utilized position-aware graph neural networks (PGNNs) to compute position-aware node embeddings. Later, Veli¸ cković et al. [31] presented deep graph infomax (DGI), which learns graph embeddings in an unsupervised manner. Similarly, new progress has been made in graph reduction [32, 33] and graph matching [34, 35]. Rong et al. [26] proposed DropEdge, a novel and flexible technique to alleviate overfitting and oversmoothing of deep graph conventional networks. Hong et al. [36] proposed a novel heterogeneous graph structural attention neural network (HetSANN) to directly encode structural information of a heterogeneous information network (HIN). More recently, GCNs and subsequent variants have obtained excellent results in many fields, including but not limited to social networks [8, 9], traffic forecasting [10–12], natural language processing [13–15], reinforcement learning [37, 38] and electronic health records (EHRs) [39].

2.2 Neural networks with dropout

Overfitting is a serious problem in deep neural networks with a large number of parameters. Dropout regularization is a common technique to alleviate overfitting. Hinton et al. [21] and Srivastava et al. [22] applied dropout to prevent complex coadapations of the training data to reduce overfitting. Bouthillier et al. [23] defined the application of dropout in the network layer as data augmentation and used it to learn better models. Furthermore, Zhao et al. [40] discussed the exact conditions for maintaining equivalence between dropout and dataaugmentation.

Due to the success of dropout, many variants have been proposed [41–43]. More recently, an increasing number of works have obtained new definitions of dropout regularization through in-depth analysis of its principles, so dropout regularization has been widely used in the field of deep learning. For instance, Dao et al. [44] proposed a general model of augmentation as a Markov process to establish a theoretical framework for understanding data augmentation. Nalisnick et al. [45] proposed a novel framework for understanding Bernoulli noise (i.e., dropout) in neural networks. Wang et al. [46] used a novel jumpout method, which samples the dropout rate from a monotonically decreasing distribution. Zunino et al. [47] presented a guided dropout regularizer for deep neural networks. With the development of dropout theory, dropout and its subsequent variants have achieved state-of-the-art results in various deep learning fields, including but not limited to recurrent neural networks [24, 48, 49], convolutional neural networks [25, 50], and graph convolutional networks [26, 27].

Our method draws inspiration from graph conventional networks, which perform convolutions directly on the graph. In contrast to these works, the proposed DGC model uses a simple and efficient dropout layer to improve the feature extraction performance of the multilayer simplified graph convolutional network model. In this case, we demonstrate that DGC can not only achieve a calculation speed increase of several orders of magnitude but can also match or outperform the state-of-the-art methods in classification accuracy.

3 Proposed model

First, some GCN notations used in this paper are defined. An undirected graph is represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents the vertex set consisting of nodes $\{v_1, v_2, \cdots, v_n\}$, and $\mathcal{E}$ is the edge set. $A = \left[a_{ij}\right] \in \mathbb{R}^{n\times n}$ is a symmetric (typically sparse) adjacency matrix, where a_{ij} denotes the edge weight between nodes v_i and v_j. $D = diag(d_1, \cdots, d_n)$ denotes the degree matrix of A, where the row-sum of the adjacency matrix $d_i = \sum_j a_{ij}$ is the degree of vertex v_i. The unnormalized graph Laplacian [51] is defined as $L = D - A$, and the two versions of normalized graph Laplacians are defined as symmetric normalized Laplacian, $L_{\text{sys}} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$, and random walk normalized Laplacian, $L_{\text{rw}} = D^{-1} L$.

The entire feature matrix $X = [\mathrm{x}_1, \cdots, \mathrm{x}_n]^{\intercal} \in \mathbb{R}^{n\times c}$ stacks n feature vectors on top of one another, and $\mathrm{x}_i \in \mathbb{R}^c$ is the c-dimensional feature vector of vertex v_i. Each vertex belongs to one out of C classes, which is labeled with a C-dimensional one-hot vector $y_i \in \{0, 1\}^C$. Only the labels of a subset are given, and the goal is to predict the labels of the remaining vertices $\mathcal{V}_u$. As shown in Fig. 1, DGC is composed of feature extraction, feature augmentation and standard logistic regression. For clarity, we describe each step in detail.

3.1 Graph convolutional networks

Graph convolutional neural networks (GCNNs) extend traditional convolutional neural networks to the graph domain. Particularly, spectral GCNNs define the convolution as the multiplication of a signal $x \in \mathbb{R}^n$ (a scalar for each vertex) with a filter $g_\theta = \mathrm{diag}(\theta)$ (a function of eigenvalues of L_{sym}) in the Fourier domain [3]. However, this model is computationally expensive. Furthermore, computing the eigendecomposition of L_{sym} might be prohibitively expensive for large graphs. Defferrard et al. [5] approximates the spectral filter g_θ with Chebyshev polynomials up to the K-th order to build a K-localized convolution, where it is defined as:

$$g_\theta \star x \approx \sum_{k=0}^{K} \theta'_k T_k(L_{\mathrm{sym}})x, \tag{1}$$

where $x \in \mathbb{R}^n$ is the signal on the graph, g_θ is the spectral filter, $\star$ denotes the convolution operator, T_k is the Chebyshev polynomial, and $\theta'_k \in \mathbb{R}^K$ is a vector of Chebyshev coefficients.

GCN [6] limits the layerwise convolution operation to $K = 1$ and further approximates the largest eigenvalue $\lambda_{\max} \approx 2$. Under these approximations, the convolution simplifies to:

$$g_\theta \star x \approx \theta \left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) x, \tag{2}$$

with a single Chebyshev coefficient parameter θ. Note that $I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ now has eigenvalues in the range $[0, 2]$. The following renormalization trick is introduced:

$$I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \to \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}, \tag{3}$$

where $\tilde{A} = A + I_N$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, $\tilde{D} = D + I_N$.

The above convolution definition is generalized to a graph signal $X \in \mathbb{R}^{n \times c}$ with c input channels (i.e., each vertex is associated with a c-dimensional feature vector) and f spectral filters as follows:

$$H^{(l+1)} = \sigma \left(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} \Theta^{(l)} \right), \tag{4}$$

where $H^{(l)}$ is the matrix of feature representation in the l-th layer, $H^{(0)} = X$, $\Theta^{(l)} \in \mathbb{R}^{c \times f}$ is the learned weight matrix of layer l, and σ is the activation function, e.g., $ReLU(\cdot) = max(0, \cdot)$.

For node classification, the last layer of a GCN utilizes a softmax classifier to predict the labels . The class prediction $\hat{Y}$ of a K-layer GCN can be defined as:

$$\hat{Y}_{\mathrm{GCN}} = \mathrm{softmax} \left(\hat{A} H^{(K-1)} \Theta^{(K)} \right), \tag{5}$$

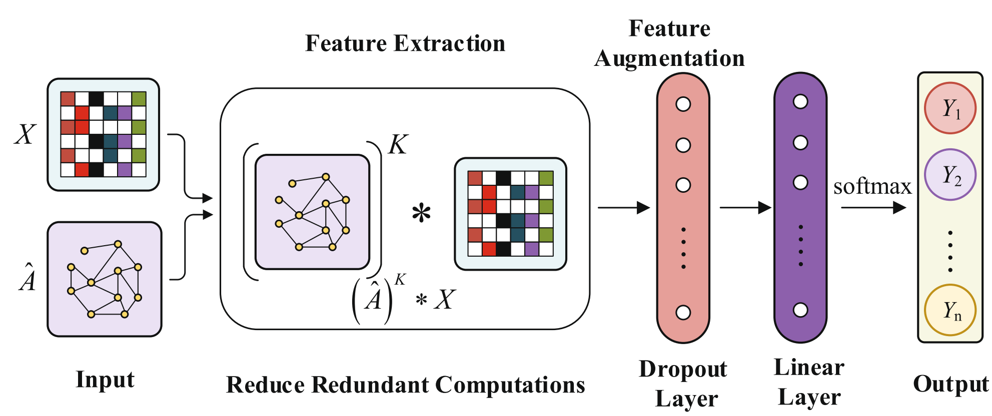


Fig. 1 Schematic layout of DGC. The input to the DGC model consists of $X \in \mathbb{R}^{n \times c}$ and $\hat{A} \in \mathbb{R}^{n \times n}$, which are the feature matrix and renormalization adjacency matrix, respectively. In addition, $\hat{A} = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$, where $\tilde{A} = A + I_N$ and $\tilde{D} = D + I_N$. [7] proved that the nonlinearity between GCN layers is not critical. What matters is the local averaging, which is the source of the majority of the benefit. Here, we redefine the process of reducing redundant computations as the process of feature extraction and perform nonlinear removal and weight matrix merging in this process. Further, we briefly explain the process. The l-th layer of the GCN [6] is defined as $H^{(l+1)} = \sigma\left(\hat{A} H^{(l)} \Theta^{(l)}\right)$. After removing the nonlinearities, the entire GCN calculation process is simplified to $Y_{\mathrm{GCN}} = \mathrm{softmax}\left(\hat{A}\hat{A} \ldots \hat{A} X \Theta^{(1)} \Theta^{(2)} \ldots \Theta^{(K)}\right)$. Then, we collapse the repeated multiplication with the normalized adjacency matrix $\hat{A}$ to a single matrix by raising $\hat{A}$ to the K-th power, $\hat{A}^{(K)}$. Furthermore, we merge weight matrices between graph conventional layers into a single matrix $\Theta = \Theta^{(1)} \Theta^{(2)} \ldots \Theta^{(K)}$, which is defined in the linear layer. Moreover, a simple and efficient dropout layer is used for feature augmentation. The linear layer is the output layer, parameterized by a hidden-to-output weight matrix, $\Theta \in \mathbb{R}^{c \times n}$. Finally, the softmax activation function is applied rowwise. In this case, the DGC model is simplified to $\hat{Y}_{\mathrm{DGC}} = \mathrm{softmax}\left(\mathrm{dropout}\left(\hat{A}^{(K)} X\right) \Theta\right)$

where $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ denotes the "normalized" adjacency matrix with added self-loops and $\mathrm{softmax}(x_i) = \exp(x_i)/\sum_{c=1}^{C}\exp(x_i)$ is the softmax classifier. The class predictions for n nodes are denoted as $\hat{Y}_{\mathrm{GCN}} \in \mathbb{R}^{n\times c}$.

A one-layer GCN actually contains two operations. First, a new feature matrix is generated Y from X by applying graph convolution:

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

Here, $\tilde{A} = A + I$. Then, the new feature matrix Y is fed to a fully connected layer. Obviously, graph convolution is the key to performance improvement.

The Laplacian smoothing [52] on each channel of the input features is defined as:

$$\hat{\mathrm{y}}_i = (1-\gamma)\mathrm{x}_i + \gamma\sum_j \frac{\tilde{a}_{ij}}{d_i}\mathrm{x}_j \qquad (\text{for } 1 \le i \le n)\,, \tag{7}$$

where $0 < \gamma \le 1$ is a weight parameter that is related to the features of the current vertex and the features of its neighbors. The Laplacian smoothing can be rewritten as:

$$\hat{Y} = X - \gamma\tilde{D}^{-1}\tilde{L}X, \tag{8}$$

where $\tilde{L} = \tilde{D} - \tilde{A}$, By letting $\gamma = 1$, i.e., only the neighbors' features is considered. Here, $\hat{Y} = \tilde{D}^{-1}\tilde{A}X$, which is the standard Laplacian smoothing.

Specifically, when the normalized Laplacian $\tilde{D}^{-1}\tilde{L}$ is replaced with the symmetrically normalized Laplacian $\tilde{D}^{-\frac{1}{2}}\tilde{L}\tilde{D}^{-\frac{1}{2}}$ and $\gamma = 1$, the feature propagation formula becomes $\hat{Y} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}X$, which is exactly the graph convolution. It is demonstrated that graph convolution is a special form of Laplacian smoothing (symmetric Laplacian smoothing). Note that the smoothing still includes the current vertex's features.

The new features of a vertex are calculated as the weighted average of itself and its neighbors through Laplacian smoothing. Since vertices in the same cluster tend to be densely connected, the smoothing makes their features similar, which makes the subsequent classification task much easier.

For a GCN with two propagation layers, applying smoothing again makes the output features of vertices in the same cluster more similar. Furthermore, the classification task becomes simple. However, a GCN with many layers is computationally complex. Additionally, as the number of convolutional layers increases, the complexity and redundant calculations of the multilayer GCN model also increase, which introduces the problem of overfitting. Therefore, in the next section, we introduce the DGC model, which can alleviate the abovementioned problems of the multilayer GCN model.

3.2 Graph convolutional network with dropout

Typically, the propagation layer respects the adjacency pattern in A, performing a variation of local averaging (current vertex and its neighbors) in the shadow GCN, which is the definition of a graph convolution operation. Next, a single perceptron layer is applied to each vertex separately. Simultaneously, the effective context size for each vertex increases by the size of its K^{th}-order neighborhood (for a model with K layers) with each additional layer. However, GCN benefits little from increased depth, when the model depth exceeds 3. Representatively, GCN [6] obtains the best results with a 2- or 3-layer model for citation networks. Furthermore, overfitting becomes an issue as the number of parameters increases with model depth.

Note that the critical aspect of the graph convolutional layer is feature propagation (local averaging), rather than the nonlinearity of the graph convolutional layer. Therefore, the nonlinear activation functions between each layer are removed and only the final softmax is retained. The resulting model is linear, but still has the same increased receptive field of a layer GCN [7],

$$Y_{\mathrm{GCN}} = \mathrm{softmax}\left(\hat{A}\hat{A}\ldots\hat{A}X\Theta^{(1)}\Theta^{(2)}\ldots\Theta^{(K)}\right). \tag{9}$$

Here, the repeated multiplication with the normalized adjacency matrix $\hat{A}$ can be collapsed to a single matrix by raising $\hat{A}$ to the K-th power, $\hat{A}^{(K)}$. Furthermore, weight matrices between graph conventional layers can be merged into a single matrix $\Theta = \Theta^{(1)}\Theta^{(2)}\ldots\Theta^{(K)}$. Under these tricks, the formula simplifies to:

$$Y_{\mathrm{GCN}} = \mathrm{softmax}\left(\hat{A}^{(K)}X\Theta\right). \tag{10}$$

New feature propagation $\hat{A}^{(K)}X$ may mix the vertices features from different clusters as K increases. It has been demonstrated that dropout is an effective technique for feature augmentation and preventing the coadaptation of neurons [21–23]. During training, dropout randomly zeroes some of the elements of the input tensor with probability η by sampling from a Bernoulli distribution. Each channel is zeroed out independently on every forward call,

$$\begin{aligned} &r_j \sim \mathrm{Bernoulli}(\eta),\\ &\mathrm{dropout}(x) = r * x. \end{aligned} \tag{11}$$

Here, $*$ denotes an elementwise product. r is a vector of independent Bernoulli random variables, and the probability of each random variable being 1 is η. After using dropout, the resulting classifier becomes

$$\hat{Y}_{\mathrm{DGC}} = \mathrm{softmax}\left(\mathrm{dropout}\left(\hat{A}^{(K)}X\right)\Theta\right), \tag{12}$$

which refers to simplified multilayer graph convolutional networks with dropout (DGC).

4 Experiments and analysis

We test DGC in many experiments: semi-supervised and full-supervised document classification in citation networks, inductive prediction of the community structure on social networks (Reddit), and a range of downstream tasks. This section summarizes our experimental setup, the results, and a brief qualitative analysis of a DGC model's feature extraction and feature augmentation.

4.1 Citation networks

Datasets Three commonly used citation network datasets are chosen: Cora, Citeseer, Pubmed [53]. Each dataset contains sparse bag-of-words feature vectors for a document and a list of citation links between documents, and the citation links are described as a binary, symmetric adjacency matrix A. Although the networks are directed, undirected versions of the graphs are considered for all experiments. In addition, each category uses 20 nodes for training. We define the fixed label rate as Ra. Dataset statistics are summarized in Table 1.

Experimental setup We closely follow the experimental setup in [7]. On the citation networks, DGC is trained for 100 epochs using Adam [54] with a learning rate of 0.2. In addition, weight decay is applied and tuned on each dataset using hyperopt [55]. Experiments on citation networks are conducted transductively.

Baselines For citation networks, we compare DGC to the reported performance of graph conventional network (GCN) [6], graph attention network (GAT) [27], attention-based graph neural network (AGNN) [19], FastGCN [18], (LNet, AdaLNet) [56], deep graph infomax (DGI) [31] and SGC [7].

Hyperparameters tuning From the experimental settings and the definition of $\hat{Y}_{\mathrm{DGC}}$ (12), we can see that the parameters that can be adjusted are the dropout rate η and the model depth K. We discuss the impact of η on model performance under different model depths, where the model depth gradually changes from 1 to 7, and η gradually increases from 0 to 0.7 at intervals of 0.05. Figure 2 shows the process of hyperparameter adjustment. Simultaneously, we can conclude that when η is set to 0.4, 0.45, and 0.5, DGC can obtain the best performance on Cora, Citeseer and Pubmed, respectively. In addition, we also notice that DGC usually achieves better performance when the model depth is relatively shallow. Finally, these two hyperparameters are also used in subsequent experiments.

Table 2 Test accuracy (%) on citation networks, where the dropout rates of DGC on Cora, Citeseer and Pubmed are 0.4, 0.45 and 0.5, respectively

Method	Cora	Citeseer	Pubmed
GCN	81.5	70.3	79.0
GAT	83.0±0.7	72.5±0.7	79.0±0.3
GLN	81.2±0.1	70.9±0.1	78.9±0.1
AGNN	83.1±0.1	71.7±0.1	79.9±0.1
FastGCN	79.8±0.3	68.8±0.6	77.4±0.3
LNet	79.5±1.8	66.2±1.9	78.3±0.3
AdaLNet	80.4±1.1	68.7±1.0	78.1±0.4
DeepWalk	70.7±0.6	51.4±0.5	76.8±0.6
DGI	82.3±0.6	71.8±0.7	76.8±0.6
SGC	81.0±0.0	71.9±0.1	78.9±0.0
DGC (2)	81.9±0.0	71.7±0.0	79.9±0.0
DGC (3)	82.3±0.0	71.8±0.0	79.1±0.0

DGC (K) indicates that DGC theoretically has K convolutional layers

Results in terms of test accuracy The results are summarized in Table 2, and the results for all other baseline methods are taken from [7]. Table 2 reflects that the performance of DGC with different convolutional layers can match or outperform the performance of SGC and state-of-the-art graph networks on citation networks. When $K = 2$, DGC is approximately 1% better than SGC on Cora and Pubmed, and approximately 1% better than GCN on Citeseer. Specifically, on Pubmed, DGC is compared to GAT and AGNN; however, DGC has a faster running speed. Note that different datasets require different K (convolutional layers) and different η (dropout rate) for better performance. Actually, DGC obtains higher accuracy with reasonable K and η, e.g., test accuracy up to 82.5% on Cora when $K = 3$

Table 1 Dataset statistics of the citation networks and Reddit

Dataset	Nodes	Edges	Classes	Features	Train/Dev/Test Nodes
Cora	2708	5429	7	1433	140/500/1000
Citeseer	3327	4732	6	3703	120/500/1000
Pubmed	19717	44338	3	500	60/500/1000
Reddit	233K	11.6M	41	602	152K/24K/55K

Table 3 Summary of results in terms of p-values

Method	GCN	GAT	GLN	AGN.	Fas.	LNe.	Ada.	Dee.	DGI	SGC	DGC (2)	DGC (3)
DGC (2)	0.109	0.593	0.108	0.317	0.108	0.108	0.108	0.108	1	0.285	–	1
DGC (3)	0.109	0.276	0.108	0.285	0.109	0.109	0.109	0.108	0.317	0.285	1	–

Here, we use the classification accuracy of the model on the three datasets as input for its significance test, and compare DGC with all other baselines. We set the threshold of the significance test to $\alpha = 0.05$

and $\eta = 0.6$. The reason for this performance improvement is due to the reduction in parameters and the increase in dropout layers, which can alleviate overfitting and achieve feature enhancement, respectively. For all citation networks, DGC is most stable with different K and η. Remarkably, both LNet and AdaLNet perform slightly worse and are unstable on citation networks.

Significance tests of the Results To explore the differences among the model results, we leveraged the Wilcoxon signed-rank test (WSRT) for significance tests. The WSRT is a type of nonparametric statistical hypothesis test and is usually used to compare the predictive power of models [57]. The advantage of this test is that it does not need the assumption of data normality. In addition, the subsequent experiments of this paper will uniformly use WART for significance tests. In Table 3, since each model has only three accuracy values, the quantity of data is too small, resulting in all p-values greater than α. The conclusion is that DGC is not significantly different from all models. However, this is not very convincing. In Table 4, except for DGI, there are significant differences between DGC and other models. In Table 5, in addition to AGNN and DGI, there are significant differences between DGC and other models. In Table 6, except for AGNN, there are significant differences between DGC and other models. Overall, the results suggest that on these three datasets, the classification performance of DGC is significantly different from the classification performance of most baselines.

Influence of model depth and dropout rate on classification performance In Fig. 2(a), DGC with different K almost all obtain the highest test accuracy when $\eta = 0.4$. In the whole process, DGC with $K = 3$ or $K = 4$ outperforms DGC with $K = 2$ as η increases. Figure 2(b) shows that the performance of DGC is attenuated as K increases. However, the performance of DGC improves as η increases while keeping K fixed. In particular, DGC with different K-values obtains almost all the highest test accuracies when $\eta = 0.45$. In Fig. 2(c), DGC with different K almost all obtain the highest test accuracies when $\eta = 0.6$. For citation network datasets, it is demonstrated that the best results are obtained with model depth (number of layers) $K = 2$, $K = 3$ or $K = 4$. Simultaneously, the most reasonable and commonly used η is approximately 0.5, which was utilized in [21, 22]. Moreover, in Fig. 3, we fixed the dropout rate to show the effect of the dropout layer on the DGC model with different convolutional layers. The results further highlight the effect of the dropout layer on feature augmentation.

Influence of model depth and label rate on classification performance In Fig. 4, we fixed the dropout rate to show the effect of the dropout layer on the DGC model with different convolutional layers and label rates. The dropout layer can achieve feature augmentation at low label rates. In particular, DGC with different network layers can still achieve similar feature augmentation effects. Moreover, in Fig. 5, we further show the influence of the dropout layer on the DGC model through the task of full-supervised node classification. The dropout layer can also achieve feature augmentation at high label rates (full-supervised). This demonstrates that adding a dropout layer can further enhance the robustness of DGC.

Influence of label rate on classification performance We fixed the dropout rate η and model depth K to show the effect of the label rate on the DGC model. Figure 6 demonstrates that DGC can match or outperform GCN on citation networks at low label rates. In addition, for Fig. 6(b), the accuracy of GCN with label rates between

Table 4 Summary of results in terms of p-values (e-5) on Cora

Method	GCN	GAT	GLN	AGN.	Fas.	LNe.	Ada.	Dee.	DGI	SGC	DGC (2)	DGC (3)
DGC (2)	8.02	8.52	6.39	6.81	6.92	5.79	5.75	6.93	9.49	7.5	–	6.18
DGC (3)	6.82	11.4	6.90	5.66	6.90	6.40	6.91	6.93	0.76e5	7.21	6.18	–

We run the models 20 times on Cora. Therefore, each model obtains 20 corresponding accuracy values, which are passed to a statistical test to see if the observed difference is statistically significant

Table 5 Summary of results in terms of p-values (e-5) on Citeseer

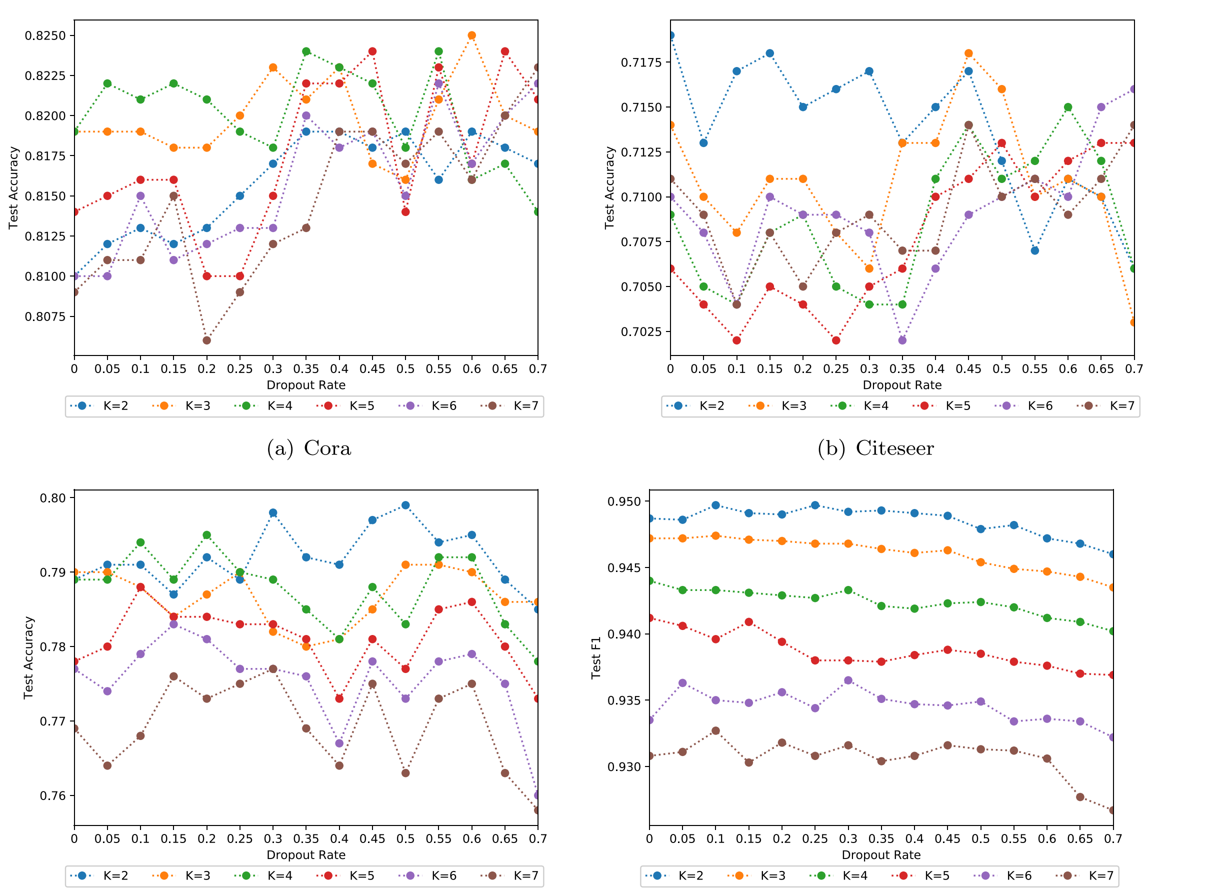
Method	GCN	GAT	GLN	AGN.	Fas.	LNe.	Ada.	Dee.	DGI	SGC	DGC (2)	DGC (3)
DGC (2)	5.70	7.62	7.24	0.76e5	6.85	7.68	7.62	7.80	0.5e5	6.56	–	7.3e1
DGC (3)	6.21	7.38	6.12	2.16e1	7.39	6.90	7.41	6.94	0.6e5	5.3e1	7.3e1	–

We run the models 20 times on Citeseer. Therefore, each model obtains 20 corresponding accuracy values, which are passed to a statistical test to see if the observed difference is statistically significant

Table 6 Summary of results in terms of p-values (e-5) on Pubmed

Method	GCN	GAT	GLN	AGN.	Fas.	LNe.	Ada.	Dee.	DGI	SGC	DGC (2)	DGC (3)
DGC (2)	1.23e2	8.64e1	1.2e1	0.439e5	8.46	9.73	9.02	7.42	5.66	1.27e1	–	1.11e2
DGC (3)	1.57e2	4.94e3	1.22e2	6.66	7.99	8.23	7.72	7.39	6.12	6.66	1.11e2	–

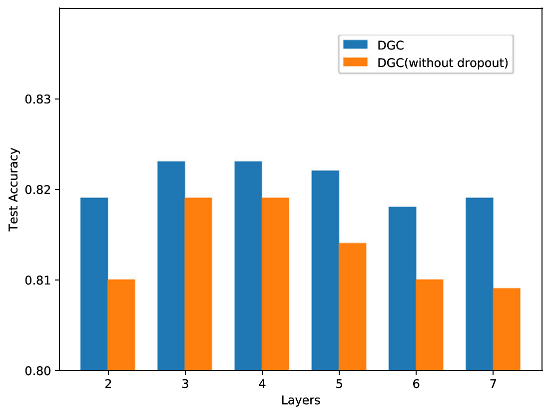
We run the model 20 times on Pubmed. Therefore, each model obtains 20 corresponding accuracy values, which are passed to a statistical test to see if the observed difference is statistically significant



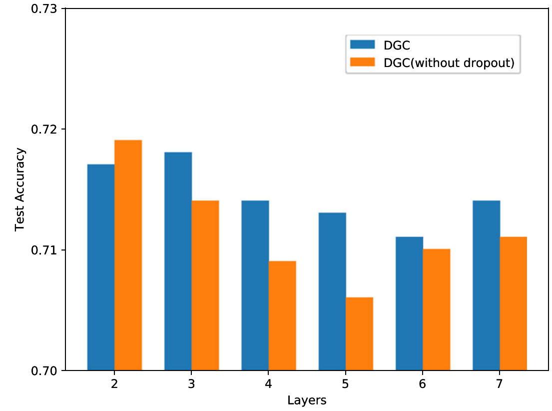
(a) Cora (b) Citeseer

(c) Pubmed (d) Reddit

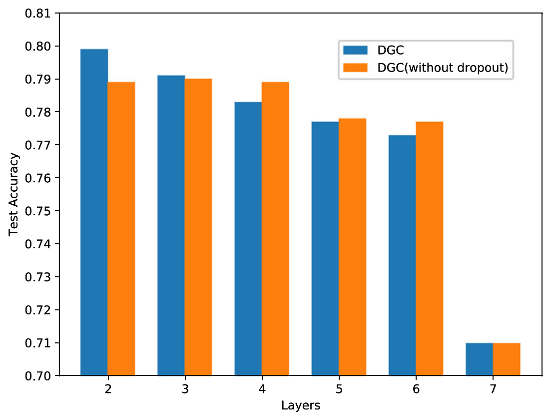
Fig. 2 Influence of model depth (number of layers) and dropout rate on classification performance. Here, DGC performs transductive tasks on Cora, Citeseer and Pubmed, and performs inductive tasks on Reddit. In addition, we test accuracy on Cora, Citeseer and Pubmed, and test the F1 score on Reddit. K is the number of convolutional layers of DGC



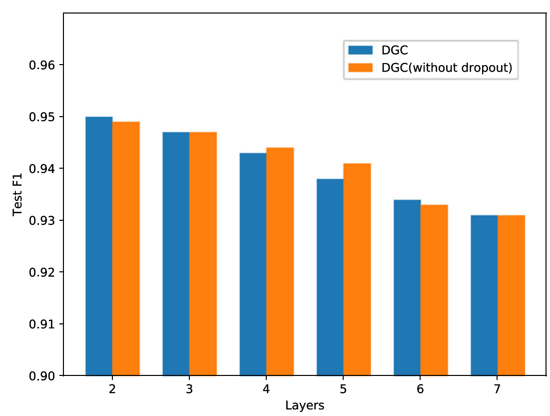
(a) Cora



(b) Citeseer



(c) Pubmed

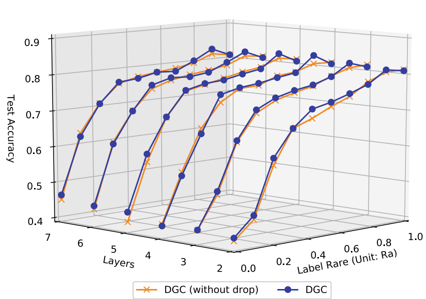


(d) Reddit

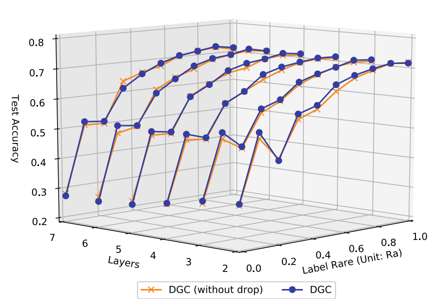
Fig. 3 Influence of the dropout layer on the classification performance. Here, we fixed the dropout rates of Cora, Citeseer, Pubmed and Reddit at 0.4, 0.45, 0.5, and 0.2, respectively. In addition, DGC (without dropout) indicates that no dropout layer is used, that is, the dropout rate is 0

0.3*Ra* and 0.4*Ra* is greatly improved, which is caused by the early stopping of GCN with label rates between 0.1*Ra* and 0.3*Ra*.

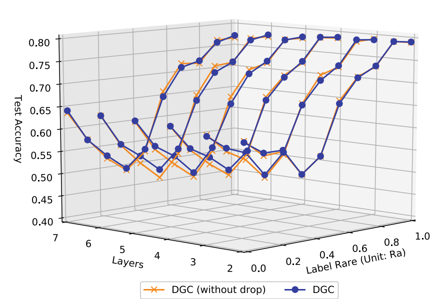
Run time analysis Table 7 shows the specific running time on citation networks. The training time was measured on an NVIDIA TITAN Xp GPU. Particularly, $\hat{A}^{(K)}X$ is precomputed and the training time of DGC considers this precomputational time. Note that DGC can match the training time of SGC, and even its training time is several orders of magnitude faster than GCN, FastGCN and AGNN. In addition, adding an additional propagation layer



(a) Cora

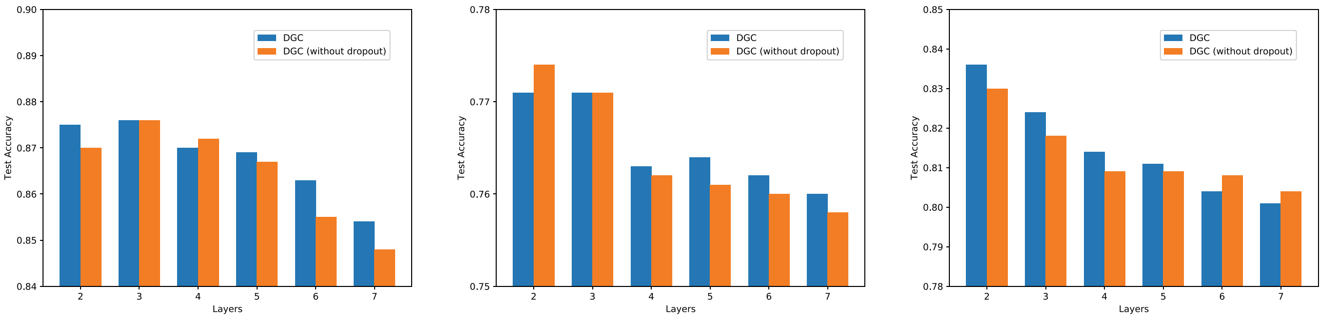


(b) Citeseer



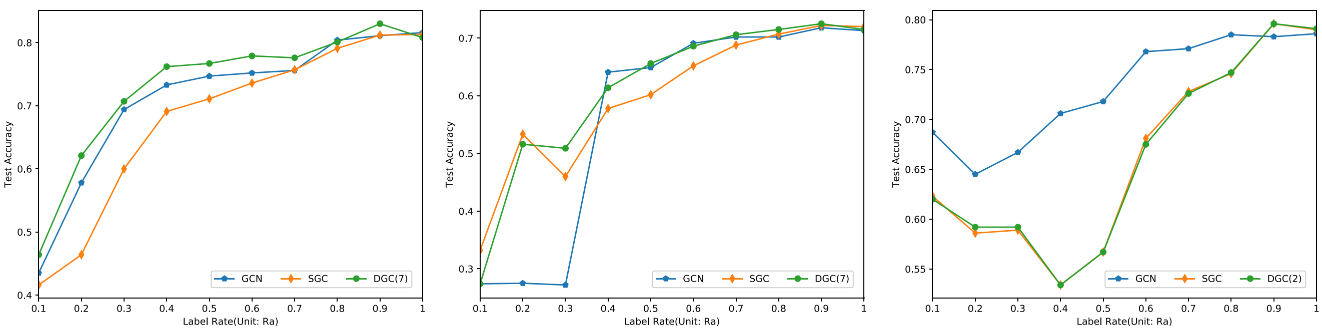
(c) Pubmed

Fig. 4 Influence of model depth and label rate on classification performance. Here, we fixed the dropout rates of Cora, Citeseer and Pubmed at 0.4, 0.1 and 0.1, respectively. Similarly, DGC (without dropout) indicates that no dropout layer is used, that is, the dropout rate is 0



(a) Cora (b) Citeseer (c) Pubmed

Fig. 5 Influence of the dropout layer on the full-supervised node classification performance. DGC (without dropout) indicates that no dropout layer is used, that is, the dropout rate is 0



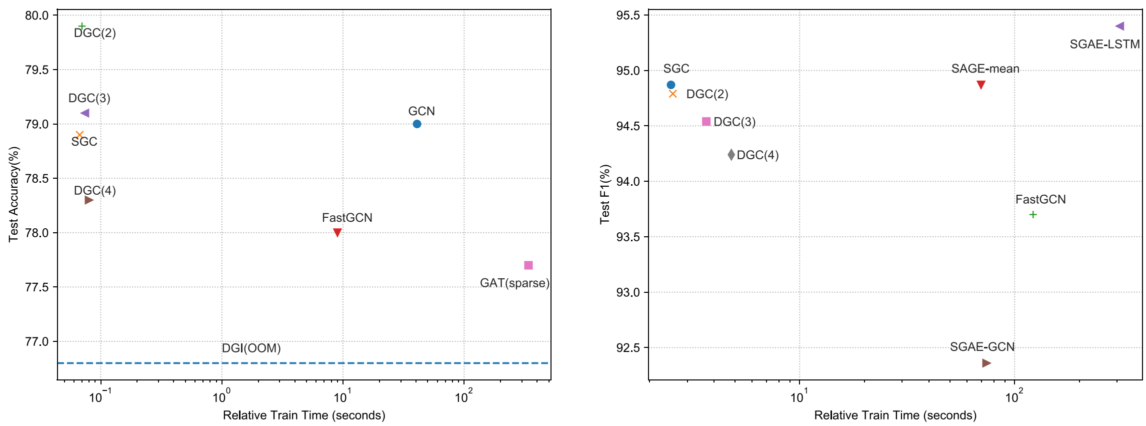
(a) Cora (b) Citeseer (c) Pubmed

Fig. 6 Influence of label rate on the classification performance. Here, we test the accuracy at different low label rates on the citation networks, and fix the dropout rates of Cora, Citeseer and Pubmed at 0.4, 0.1 and 0.1, respectively

Table 7 Training time (seconds) on citation networks

Method	Cora	Citeseer	Pubmed
GCN	7.9	11.9	40.86
SGC	0.059	0.094	0.067
FastGCN	6.528	13.287	9.0
GAT	142.91	187.66	OOM
AGNN	10.17	15.11	OOM
DGC (2)	0.062	0.098	0.07
DGC (3)	0.064	0.1	0.074
DGC (4)	0.065	0.103	0.082

To maintain consistency, the dropout rate of DGC is uniformly set to 0.5. DGC (K) means that DGC theoretically has K convolutional layers. OOM: GPU out of memory



(a) Pubmed (b) Reddit

Fig. 7 Performance over training time (seconds) on Pubmed and Reddit. Here, we show the running time on two relatively large datasets. OOM: GPU out of memory

to DGC only increases the training time by a small amount. Moreover, Fig. 7(a) shows the performance of the state-of-the-art graph networks over their training time relative to that of DGC on the Pubmed datasets. DGC can be trained up to two orders of magnitude faster than fast sampling-based methods while having little or no drop in performance.

Significance tests of the run times In Table 8, since each model has only three run time values, the quantity of data is too small, resulting in all p-values greater than α. The conclusion is that the run time of DGC is not significantly different from the run time of all models. However, this is not very convincing. In Table 9, all p-values are far less than α, which indicates that there are significant differences between DGC and all other models. Besides, since we use more data for the significance test, the experimental results are more accurate. Overall, the results suggest that on these three datasets, the run time of DGC is significantly different from the run time of all baselines.

4.2 Social networks

Datasets and experimental setup For social networks, we followed [18] to perform inductive evaluation of the DGC model on Reddit. The parameters of DGC are optimized through L-BFGS [58]. In addition, the experimental environment of the social networks is the same as the experimental environment of the citation networks.

Baselines For Reddit, we compare against GaAN [59], supervised and unsupervised variants of GraphSAGE [17], FastGCN [18], GCN [6] and SGC [7]. The setting of the feature extraction for each method is exhibited in Table 10. Notably, DGC involves no learning and no parameter feature extraction step $\hat{A}^{(K)}X$. Finally, both no-learning and unsupervised methods train logistic regression models with labels.

Results in terms of micro F1 score On Reddit, Table 10 shows that DGC has comparable performance to SGC. Moreover, DGC outperforms the previous sampling-based GCN variants, SAGE-GCN and FastGCN by more than 1%. In addition, GCN cannot be trained on large graphs (e.g. Reddit) due to out-of-GPU memory problems. Previous methods solve this limitation by either sampling to shrink the neighborhood [17, 18] or limiting their model sizes [31]. By precomputing $\hat{A}^{(K)}X$ and the dropout layer, DGC can not only minimize memory usage (only learns a single weight matrix during training), but also enhance feature representation.

Table 8 Summary of results in terms of p-values

Method	GCN	SGC	FastGCN	GAT	AGNN	DGC (2)	DGC (3)	DGC (4)
DGC (2)	0.109	0.102	0.109	–	–	–	0.102	0.109
DGC (3)	0.109	0.109	0.109	–	–	0.102	–	0.109
DGC (4)	0.109	0.109	0.109	–	–	0.109	0.109	–

Here, we use the training time (seconds) of the model on the three datasets as input for its significance test. Since GAT and AGNN has out-of-GPU memory problems on Pubmed, we did not compare DGC with them. We set the threshold of the significance test to $\alpha = 0.05$

Table 9 Summary of results in terms of p-values (e-5) on the three citation networks

Dataset	Method	GCN	SGC	FastGCN	GAT	AGNN	DGC (2)	DGC (3)	DGC (4)
Cora	DGC (2)	8.41	5.72e1	8.11	8.43	8.16	–	8.12	6.84
	DGC (3)	8.57	1.45e1	7.84	8.41	8.16	8.12	–	9.60
	DGC (4)	8.38	6.38	7.82	8.43	8.16	6.84	9.60	–
Citeseer	DGC (2)	8.72	8.07	7.79	8.82	8.73	–	1.1e3	1.48e1
	DGC (3)	8.60	7.38	8.0	8.83	8.43	1.1e3	–	7.91e1
	DGC (4)	8.72	6.40	7.83	8.84	8.18	1.48e1	7.91e1	–
Pubmed	DGC (2)	7.85	1.52e1	7.85	–	–	–	1.73e1	8.11
	DGC (3)	8.14	8.15	8.16	–	–	1.73e1	–	1.15e1
	DGC (4)	8.41	7.78	8.41	–	–	8.11	1.15e1	–

We run the models 20 times on each dataset. Therefore, each model obtains 20 corresponding run time values, which are passed to a statistical test to see if the observed difference is statistically significant. Since GAT and AGNN have out-of-GPU memory problems on Pubmed, we did not compare DGC with them on Pubmed

Significance tests of the Results In Table 11, all p-values are less than α, which indicates that there are significant differences between DGC and all other models. Particularly, as shown in Table 10, the F1 values of SGC and SAGE-mean (superimposed) are very close to the F1 value of DGC, with a difference of 0.1. However, through the significance test results in Table 11, we know that there are significant differences between the F1 value of DGC and the F1 values of SGC and SAGE-mean (superimposed). This also illustrates the importance of significance tests in analyzing experimental results.

Influence of model depth and dropout rate on classification performance For social networks Reddit, Fig. 2(d) clearly reflects that the performance of DGC worsens as K increases. Obviously, overfitting becomes the main cause of performance degradation. However, adding a dropout layer can still slightly improve the performance for a fixed K. Moreover, in Fig. 3(d), we fixed the dropout rate to show the effect of the dropout layer on the DGC model with different convolutional layers. The results further demonstrate the feature augmentation effect of the dropout layer on a large graph.

Run time analysis Figure 7(b) shows the performance of the state-of-the-art graph networks over their training time relative to that of DGC on the Reddit datasets. Here, we also consider the precomputation time. DGC can be

Table 10 Test Micro F1 Score (%) on Reddit, where the dropout rate of DGC is 0.25

Setting	Model	Test F1
Supervised	GaAN	96.4
	SAGE-mean	94.9
	SAGE-LSTM	95.4
	SAGE-GCN	92.4
	FastGCN	93.7
	GCN	OOM
Unsupervised	SAGE-mean	89.7
	SAGE-LSTM	90.7
	SAGE-GCN	90.8
No Learning	SGC	94.9
	DGC (2)	95.0

DGC (K) means that DGC theoretically has K convolutional layers. OOM: GPU out of memory

Table 11 Summary of results in terms of p-values (e-5) on Reddit

Setting	Model	DGC (2)
Supervised	GaAN	8.46
	SAGE-mean	2.0e3
	SAGE-LSTM	6.25e1
	SAGE-GCN	6.95
	FastGCN	8.09
	GCN	–
Unsupervised	SAGE-mean	8.27
	SAGE-LSTM	6.93
	SAGE-GCN	8.35
No Learning	SGC	1.8e3

We run the models 20 times on Reddit. Therefore, each model obtains 20 corresponding F1 values, which are passed to a statistical test to see if the observed difference is statistically significant. Since GCN has out-of-GPU memory problems on Reddit, we did not compare DGC with GCN on Pubmed

Table 12 Test accuracy (%) on text classification datasets

Dataset	Model	Test ACC. ↑	Time (seconds)
20NG	GCN	87.9±0.2	2022.4
	SGC	88.5±0.1	32.19
	DGC	88.7±0.0	33.21
R8	GCN	97.0±0.2	167.2
	SGC	97.2±0.1	2.45
	DGC	97.6±0.0	2.64
R52	GCN	93.8±0.2	280.2
	SGC	94.0±0.2	3.5
	DGC	94.5±0.0	3.67
Ohsumed	GCN	68.2±0.4	370.6
	SGC	68.5±0.3	4.53
	DGC	68.7±0.0	4.69
MR	GCN	76.3±0.3	14.6
	SGC	75.9±0.3	2.89
	DGC	76.4±0.0	3.63

Except on the Ohsumed dataset, we set the dropout rate of DGC to 0.4. On the other four datasets, we set the dropout rate of DGC to 0.6. In addition, the following models all include 2 convolutional layers

trained up to two orders of magnitude faster than sampling-based methods [17, 18] while having little or no decrease in performance. For DGC, precomputation is a nonlearning process, and the training parameter of the model is only Θ, which is the core reason for the reduction in training time.

4.3 Downstream tasks

Datasets and experimental setup We extend our empirical evaluation to text classification which assigns labels to documents. Yao et al. [13] use a shallow GCN to achieve the latest results by creating a single text graph at the corpus-level, which treats both documents and words as nodes in a graph. Weights between two word nodes are calculated through pointwise mutual information (PMI), and weights between document nodes and word nodes are normalized term frequency-inverse document frequency (TF-IDF) scores. We followed [13] to perform a comparative evaluation of the DGC model on 5 benchmark datasets.

Results in terms of test accuracy and run time Table 12 shows that DGC can not only match the running speed of SGC but also has better performance than GCN and SGC on the 5 downstream tasks. It is worth noting that GCN, SGC, and DGC all use a two-layer convolutional layer model. Once again, it demonstrates that the dropout layer can play a role in feature augmentation.

Significance tests of the Results As shown in Table 13, DGC is significantly different from other models in terms of classification accuracy and run time. It is worth noting that when the number of records used for the significance test reaches 5, the result of the significance test becomes relatively accurate. Similarly, as shown in Table 14, on each downstream task, DGC is significantly different from other models in terms of classification accuracy and running time. Overall, the results suggest that on all the five downstream tasks, classification performance and run time of DGC are significantly different from those of all baselines.

Table 13 Summary of results in terms of p-values

Category	Method	GCN	SGC
Test ACC.	DGC	0.043	0.042
Time	DGC	0.043	0.043

Here, we use the classification accuracy of the model on the five datasets as input for its significance test. Similarly, we also use the run time of the model on the five datasets as input for its another significance test. We set the threshold of the significance test to $\alpha = 0.05$

Table 14 Summary of results in terms of p-values (e-5) on the five downstream tasks

Dataset	Model	Test ACC.		Time	
		GCN	SGC	GCN	SGC
20NG	DGC	7.38	1.80e1	8.73	8.30
R8	DGC	7.71	7.74	8.43	8.05
R52	DGC	8.13	7.88	8.71	8.52
Ohsumed	DGC	8.14	1.23e1	8.43	8.27
MR	DGC	1.3e3	8.13	8.56	7.87

We run the models 20 times on each dataset. Therefore, each model obtains 20 corresponding accuracy values and run time values, which are passed to statistical tests to see if the observed difference is statistically significant

5 Conclusion

In this paper, we present a simplified multilayer graph convolutional network with dropout for semi-supervised classification on graph-structured data. In the DGC model, a dropout layer is added to the fixed feature extractor $\hat{A}^{(K)}$ for feature augmentation, which can not only greatly reduce overfitting, but also improve the classification ability of DGC. Many experiments on various tasks demonstrate that the proposed DGC model is competent for extracting both graph structure and node features efficiently for semi-supervised classification. Simultaneously, the experimental results also show that the most reasonable number of convolutional layers is 2 to 4 layers, and the most favorable dropout rate η for data augmentation is approximately 0.5. In this setting, DGC matches or outperforms state-of-the-art methods, while being computationally efficient.

The proposed DGC model has some potential improvements and extensions that can be solved in future work, such as exploring deeper convolutional layer networks and extending DGC to more downstream tasks. In the future, we will also explore different methods for optimizing the deeper simplified convolutional network model. Furthermore, from an application point of view, it is of great practical significance to extend this method to perform graph classification and link prediction and even extend it to other research fields, such as natural language processing and computer vision.

Acknowledgements This work is supported by the National Natural Science Foundation of China (Grant No. 61772386), and the Open Fund of Key Laboratory of Urban Land Resources Monitoring and Simulation, Ministry of Natural Resources (Grant No. KF-2020-05-014).

References

1. Gori M, Monfardini G, Scarselli F (2005) A new model for learning in graph domains. In: Proceedings. 2005 IEEE international joint conference on neural networks, 2005, vol 2. IEEE, Montreal, pp 729–734. https://doi.org/10.1109/IJCNN.2005.1555942
2. Scarselli F, Gori M, Chung Tsoi A, Hagenbuchner M, Monfardini G (2009) The graph neural network model. IEEE Trans Neural Netw 20(1):61–80. https://doi.org/10.1109/TNN.2008.2005605
3. Bruna J, Zaremba W, Szlam A, LeCun Y (2014) Spectral networks and locally connected networks on graphs. In: 2th International conference on learning representations
4. Henaff M, Bruna J, LeCun Y (2015) Deep convolutional networks on graph-structured data. arXiv:1506.05163
5. Defferrard M, Bresson X, Vandergheynst P (2016) Convolutional neural networks on graphs with fast localized spectral filtering. In: Advances in neural information processing systems. pp 3844–3852
6. Kipf TN, Welling M (2017) Semi-supervised classification with graph convolutional networks. In: 5th international conference on learning representations
7. Wu F, Zhang T (2019) Simplifying graph convolutional networks. In: Proceedings of the 36th international conference on machine learning, vol 97, pp 6861–6871
8. Bian T, Xiao X, Xu T, Zhao P, Huang W, Rong Y, Huang J (2020) Rumor detection on social media with bi-directional graph convolutional networks. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 549–556. https://doi.org/10.1609/aaai.v34i01.5393
9. Wu Y, Lian D, Xu Y, Wu L, Chen E (2020) Graph convolutional networks with markov random field reasoning for social spammer detection. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 1054–1061. https://doi.org/10.1609/aaai.v34i01.5455
10. Zheng C, Fan X, Wang C, Qi J (2020) GMAN: a graph multi-attention network for traffic prediction. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 1234–1241. https://doi.org/10.1609/aaai.v34i01.5477
11. Song C, Lin Y, Guo S, Wan H (2020) Spatial-temporal synchronous graph convolutional networks: a new framework for spatial-temporal network data forecasting. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 914–921. https://doi.org/10.1609/aaai.v34i01.5438
12. Zhang Q, Chang J, Meng G, Xiang S, Pan C (2020) Spatio-temporal graph structure learning for traffic forecasting. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 1177–1185. https://doi.org/10.1609/aaai.v34i01.5470
13. Yao L, Mao C, Luo Y (2019) Graph convolutional networks for text classification. In: Proceedings of the AAAI conference on artificial intelligence
14. Xu X, Wang T, Yang Y, Hanjalic A, Shen HT (2021) Radial graph convolutional network for visual question generation. IEEE Trans Neural Netw Learn Syst 32(4):1654–1667

15. Tang Z, Wan B, Yang L (2020) Word-character graph convolution network for chinese named entity recognition. IEEE/ACM Trans Audio Speech Lang Process 28:1520–1532
16. Lin J, Yuan Y, Shao T, Zhou K (2020) Towards high-fidelity 3D face reconstruction from in-the-wild images using graph convolutional networks. In: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR). IEEE, Seattle, pp 5890–5899. https://doi.org/10.1109/CVPR42600.2020.00593
17. Hamilton WL, Ying R, Leskovec J (2017) Inductive representation learning on large graphs. In: Advances in neural information processing systems. pp 1024–1034
18. Chen J, Ma T, Xiao C (2018) FastGCN: fast learning with graph convolutional networks via importance sampling. In: 6th international conference on learning representations
19. Thekumparampil KK, Wang C, Oh S, Li LJ (2018) Attention-based graph neural network for semi-supervised learning. arXiv:1803.03735
20. Li Q, Han Z, Wu XM (2018) Deeper insights into graph convolutional networks for semi-supervised learning. In: Proceedings of the AAAI conference on artificial intelligence, pp 3538–3545
21. Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR (2012) Improving neural networks by preventing co-adaptation of feature detectors. arXiv:1207.0580, https://doi.org/10.9774/GLEAF.978\-1\-909493\-38\-4_2
22. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting. J Mach Learn Res 15(1):1929–1958
23. Bouthillier X, Konda K, Vincent P, Memisevic R (2016) Dropout as data augmentation. arXiv:1506.08700
24. Zolna K, Arpit D, Suhubdy D, Bengio Y (2018) Fraternal dropout. In: 6th International conference on learning representations
25. Khan SH, Hayat M, Porikli F (2019) Regularization of deep neural networks with spectral dropout. Neural Netw 110:82–90. https://doi.org/10.1016/j.neunet.2018.09.009
26. Rong Y, Huang W, Xu T, Huang J (2020) Dropedge: towards deep graph convolutional network on node classification. In: Proceedings of the 37th International conference on machine learning
27. Veličković P, Cucurull G, Casanova A, Romero A, Liò P, Bengio Y (2018) Graph attention networks. In: 6th International conference on learning representations
28. Abu-El-Haija S, Perozzi B, Kapoor A, Alipourfard N, Lerman K, Harutyunyan H, Steeg GV, Galstyan A (2019) MixHop: higher-order graph convolutional architectures via sparsified neighborhood mixing. In: Proceedings of the 36th International conference on machine learning. pp 21–29
29. Qu M, Bengio Y, Tang J (2019) GMNN: graph Markov neural networks. In: Proceedings of the 36th international conference on machine learning, vol 97, pp 5214–5250
30. You J, Ying R, Leskovec J (2019) Position-aware graph neural networks. In: Proceedings of the 36th international conference on machine learning, vol 97, pp 7134–7143
31. Veliˇ cković P, Fedus W, Hamilton WL, Liò P, Bengio Y, Hjelm RD (2019) Deep graph Infomax. In: 7th international conference on learning representations
32. Loukas A (2019) Graph reduction with spectral and cut guarantees. J Mach Learn Res 20(116):1–42
33. Sun Y, Li X (2021) Using statistical measures and machine learning for graph reduction to solve maximum weight clique problems. IEEE Trans Pattern Anal Mach Intell 43(5):1746–1760
34. Yang X, Liu ZY, Qiao H (2020) A continuation method for graph matching based feature correspondence. IEEE Trans Pattern Anal Mach Intell 42(8):1809–1822. https://doi.org/10.1109/TPAMI.2019.2903483
35. Hu W, Shi X, Zhou Z, Xing J, Ling H, Maybank S (2020) Dual L1-normalized context aware tensor power iteration and its applications to multi-object tracking and multi-graph matching. Int J Comput Vis 128(2):360–392. https://doi.org/10.1007/s11263-019-01231-y
36. Hong H, Guo H, Lin Y, Yang X, Li Z, Ye J (2020) An attention-based graph neural network for heterogeneous structural learning. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 4132–4139. https://doi.org/10.1609/aaai.v34i04.5833
37. Adel T, Weller A (2019) TibGM: a transferable and information-based graphical model approach for reinforcement learning. In: Proceedings of the 36th international conference on machine learning
38. Chen L, Chen Z, Tan B, Long S, Gasic M, Yu K (2019) Agent-Graph: toward universal dialogue management with structured deep reinforcement learning. IEEE/ACM Trans Audio Speech Lang Process 27(9):1378–1391. https://doi.org/10.1109/TASLP.2019.2919872
39. Choi E, Xu Z, Li Y, Dusenberry M, Flores G, Xue E, Dai A (2020) Learning the graphical structure of electronic health records with graph convolutional transformer. In: Proceedings of the AAAI Conference on artificial intelligence, vol 34, pp 606–613. https://doi.org/10.1609/aaai.v34i01.5400
40. Zhao D, Yu G, Xu P, Luo M (2019) Equivalence between dropout and data augmentation: A mathematical check. Neural Netw 115:82–89
41. Huang G, Sun Y, Liu Z, Sedra D, Weinberger K (2016) Deep networks with stochastic depth. In: European conference on computer vision. pp 646–661
42. Singh S, Hoiem D, Forsyth D (2016) Swapout: learning an ensemble of deep architectures. In: Advances in neural information processing systems. pp 28–36
43. Achille A, Soatto S (2018) Information dropout: learning optimal representations through noisy computation. IEEE Trans Pattern Anal Mach Intell 40(12):2897–2905. https://doi.org/10.1109/TPAMI.2017.2784440
44. Dao T, Gu A, Ratner AJ, Smith V, Sa CD, Re C (2019) A kernel theory of modern data augmentation. In: Proceedings of the 36th international conference on machine learning, vol 97, pp 1528–1537
45. Nalisnick E, Hernández-Lobato JM, Smyth P (2019) Dropout as a structured shrinkage prior. In: Proceedings of the 36th international conference on machine learning, vol 97, pp 4712–4722
46. Wang S, Zhou T, Bilmes JA (2019) Jumpout : improved dropout for deep neural networks with ReLUs. In: Proceedings of the 36th international conference on machine learning, pp 6668–6676
47. Zunino A, Bargal SA, Morerio P, Zhang J, Sclaroff S, Murino V (2021) Excitation dropout: encouraging plasticity in deep neural networks. Int J Comput Vis 129(4):1139–1152. https://doi.org/10.1007/s11263-020-01422-y
48. Krueger D, Maharaj T, Kramár J, Pezeshki M, Ballas N, Ke NR, Goyal A, Bengio Y, Courville A, Pal C (2017) Zoneout: regularizing RNNs by randomly preserving hidden activations. In: 5th International conference on learning representations
49. Bacciu D, Crecchi F (2020) Augmenting recurrent neural networks resilience by dropout. IEEE Trans Neural Netw Learn Syst 31(1):345–351. https://doi.org/10.1109/TNNLS.2019.2899744
50. Tompson J, Goroshin R, Jain A, LeCun Y, Bregler C (2015) Efficient object localization using convolutional networks. In: 2015 IEEE conference on computer vision and pattern recognition (CVPR), pp 648–656
51. Chung FR, Graham FC (1997) Spectral graph theory. American Mathematical Soc
52. Taubin G (1995) A signal processing approach to fair surface design. In: Proceedings of the 22nd annual conference on computer graphics and interactive techniques - SIGGRAPH '95. ACM Press, Not Known, pp 351–358. https://doi.org/10.1145/218380.218473
53. Sen P, Namata G, Bilgic M, Getoor L, Galligher B, Eliassi-Rad T (2008) Collective classification in network data. AI Mag 29(3):93. https://doi.org/10.1609/aimag.v29i3.2157

54. Kingma DP, Ba J (2015) Adam: a method for stochastic optimization. In: 3th international conference on learning representations
55. Bergstra J, Komer B, Eliasmith C, Yamins D, Cox DD (2015) Hyperopt: a Python library for model selection and hyperparameter optimization. Comput Sci Discov 8(1). https://doi.org/10.1088/1749-4699/8/1/014008
56. Liao R, Zhao Z, Urtasun R, Zemel RS (2019) LanczosNet: multi-scale deep graph convolutional networks. In: 7th International conference on learning representations
57. Hoang ND, Nguyen QL (2019) A novel method for asphalt pavement crack classification based on image processing and machine learning. Eng Comput 35(2):487–498. https://doi.org/10.1007/s00366-018-0611-9
58. Liu DC, Nocedal J (1989) On the limited memory BFGS method for large scale optimization. Math Program 45(1-3):503–528. https://doi.org/10.1007/BF01589116
59. Zhang J, Shi X, Xie J, Ma H, King I, Yeung DY (2018) GaAN: gated attention networks for learning on large and spatiotemporal graphs. In: Proceedings of the 34th conference on uncertainty in artificial intelligence. pp 339–349

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Fei Yang is a Ph.D. candidate in School of Computer Science at Wuhan University. He received the B.S. degree in Electronic Information Engineering from Heilongjiang Institute of Technology. His main research interests include spatio-temporal big data mining, social networks and deep learning.

Huyin Zhang. He is currently a Professor of Computer Science with Wuhan University, where he directs the National Experimental Teaching Center of virtual simulation and has trained over eighty graduate students. He has authored 10 monographs and textbooks, over 70 papers, of which more than 50 were SCI/EI/ISTP indexed. His main research interests focus on computer networks and communications, network QoS, data center software energy-saving, high-performance computing, the next generation Internet, network management, e-Learning, and multimedia distance education.

Shiming Tao is a Ph.D. candidate in School of Computer Science at Wuhan University. He received his B.S. degrees in Mathematics and Applied Mathematics from College of Mathematics and Computer Science, Hunan Normal University and the M.S. degree in Operational Research and Cybernetics in Department of Mathematics, Tongji University. His research interests include wireless network, network virtualization, mobile edge computing and optimization algorithm design, etc.