GLCN

Semi-supervised Learning with Graph Learning-Convolutional Networks

Bo Jiang, Ziyan Zhang, Doudou Lin, Jin Tang*and Bin Luo
School of Computer Science and Technology, Anhui University, Hefei, 230601, China
jiangbo@ahu.edu.cn,{zhangziyanahu,ahu_lindd}@163.com,ahhftang@gmail.com,luobin@ahu.edu.cn

Abstract

*Graph Convolutional Neural Networks (graph CNNs) have been widely used for graph data representation and semi-supervised learning tasks. However, existing graph CNNs generally use a fixed graph which may not be optimal for semi-supervised learning tasks. In this paper, we propose a novel Graph Learning-Convolutional Network (GLCN) for graph data representation and semi-supervised learning. The aim of GLCN is to learn an optimal graph structure that best serves graph CNNs for semi-supervised learning by integrating both **graph learning** and **graph convolution** in a unified network architecture. The main advantage is that in GLCN both given labels and the estimated labels are incorporated and thus can provide useful 'weakly' supervised information to refine (or learn) the graph construction and also to facilitate the graph convolution operation for unknown label estimation. Experimental results on seven benchmarks demonstrate that GLCN significantly outperforms the state-of-the-art traditional fixed structure based graph CNNs.*

1. Introduction

Deep neural networks have been widely used in many computer vision and pattern recognition tasks. Recently, many methods have been proposed to generalize the convolution operation on arbitrary graphs to address graph structure data [5, 1, 15, 11, 19, 21]. Overall, these methods can be categorized into spatial convolution and spectral convolution methods [22]. For spatial methods, they generally define graph convolution operation directly by defining an operation on node groups of neighbors. For example, Duvenaud et al. [5] propose a convolutional neural network that operates directly on graphs and provide an end-to-end feature learning for graph data. Atwood and Towsley [1] propose Diffusion-Convolutional Neural Networks (DCNNs) by employing a graph diffusion process to incorporate the contextual information of node in graph node classification. Monti et al. [15] present mixture model CNNs (MoNet) and provide a unified generalization of CNN architectures on graphs. By designing an attention layer, Veličković et al. [21] present Graph Attention Networks (GAT) for semi-supervised learning. For spectral methods, they generally define graph convolution operation based on spectral representation of graphs. For example, Bruna et al. [3] propose to define graph convolution in the Fourier domain based on eigen-decomposition of graph Laplacian matrix. Defferrard et al. [4] propose to approximate the spectral filters based on Chebyshev expansion of graph Laplacian to avoid the high computational complexity of eigen-decomposition. Kipf et al. [11] propose a more simple Graph Convolutional Network (GCN) for semi-supervised learning.

The above graph CNNs have been widely used for supervised or semi-supervised learning tasks. In this paper, we focus on semi-supervised learning. One important aspect of graph CNNs is the graph structure representation of data. In general, the data we provide to graph CNNs either has a known intrinsic graph structure, such as social networks, or we construct a human established graph for it, such as k-nearest neighbor graph with Gaussian kernel. However, it is difficult to evaluate whether the graphs obtained from domain knowledge (e.g., social network) or established by human are optimal for semi-supervised learning in graph CNNs. Henaff et al. [7] propose to learn a supervised graph with a fully connected network. However, the learned graph is obtained from a separate network which is also not guaranteed to best serve the graph CNNs. Li et al. [19] propose optimal graph CNNs, in which the graph is learned adaptively by using a distance metric learning. However, it use an approximate algorithm to estimate graph Laplacian which may lead to weak local optimal solution.

In this paper, we propose a novel Graph Learning-Convolutional Network (GLCN) for semi-supervised learning problem. The main idea of GLCN is to learn an optimal graph representation that best serves graph CNNs for semi-supervised learning by integrating both *graph learning* and *graph convolution* simultaneously in a unified network architecture. The main advantages of the proposed GLCN for semi-supervised learning are summarized as follows.

*Corresponding author

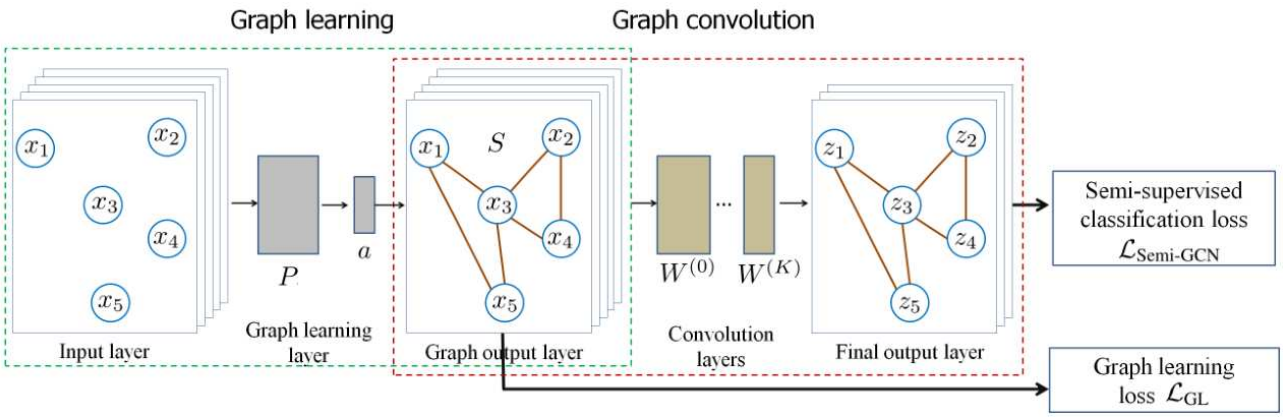


Figure 1. Architecture of the proposed GLCN network for semi-supervised learning.

- In GLCN, both given labels and the estimated labels are incorporated and thus can provide useful 'weakly' supervised information to refine (or learn) the graph construction and to facilitate the graph convolution operation in graph CNN for unknown label estimation.
- GLCN can be trained via a single optimization manner, which can thus be implemented simply.

To the best of our knowledge, this is the first attempt to build a unified graph *learning-convolutional network* architecture for semi-supervised learning. Experimental results demonstrate that GLCN outperforms state-of-the-art graph CNNs on semi-supervised learning tasks.

2. Related Work

Here, we briefly review GCN based semi-supervised learning proposed in work [11]. Let $X = (x_1, x_2, \cdots x_n) \in \mathbb{R}^{n\times p}$ be the collection of n data vectors in p dimension. Let $G(X, A)$ be the graph representation of X with $A \in \mathbb{R}^{n\times n}$ encoding the pairwise relationship (such as similarities, neighbors) among data X. GCN contains one input layer, several propagation (hidden) layers and one final perceptron layer [11]. Given an input $X^{(0)} = X$ and graph A, GCN [11] conducts the following layer-wise propagation in hidden layers as,

$$X^{(k+1)} = \sigma(D^{-1/2}AD^{-1/2}X^{(k)}W^{(k)}) \tag{1}$$

where $k = 0, 1, \cdots K-1$ and $D = \text{diag}(d_1, d_2 \cdots d_n)$ is a diagonal matrix with $d_i = \sum_{j=1}^{n} A_{ij}$. $W^{(k)} \in \mathbb{R}^{d_k \times d_{k+1}}, d_0 = p$ is a layer-specific weight matrix needing to be trained. $\sigma(\cdot)$ denotes an activation function, such as $\text{ReLU}(\cdot) = \max(0, \cdot)$, and $X^{(k+1)} \in \mathbb{R}^{n\times d_{k+1}}$ denotes the output of activations in the k-th layer. For semi-supervised classification, GCN [11] defines the final perceptron layer as

$$Z = \text{softmax}(D^{-1/2}AD^{-1/2}X^{(K)}W^{(K)}) \tag{2}$$

where $W^{(K)} \in \mathbb{R}^{d_K \times c}$ and c denotes the number of classes. The final output $Z \in \mathbb{R}^{n\times c}$ denotes the label prediction for all data X in which each row Z_i denotes the label prediction for the i-th node. The optimal weight matrices $\{W^{(0)}, W^{(1)}, \cdots W^{(K)}\}$ are trained by minimizing the cross-entropy loss function as,

$$\mathcal{L}_{\text{Semi-GCN}} = -\sum_{i\in L}\sum_{j=1}^{c} Y_{ij}\ln Z_{ij} \tag{3}$$

where L indicates the set of labeled nodes.

3. Graph Learning-Convolutional Network

One core aspect of GCN is the graph representation $G(X, A)$ of data X. In some applications, the graph structure of data are available from domain knowledge, such as chemical molecules, social networks etc. In this case, one can use the existing graph directly for GCN based semi-supervised learning. In many other applications, the graph data are not available. One popular way is to construct a human established graph (e.g., k-nearest neighbor graph) [8] for GCN. However, the graphs obtained from domain knowledge or estimated by human are generally independent of GCN (semi-supervised) learning process and thus are not guaranteed to best serve GCN learning. Also, the human established graphs are usually sensitive to the local noise and outliers. To overcome these problems, we propose a novel Graph Learning-Convolution Network (GLCN) which integrates graph learning and graph convolution simultaneously in a unified network architecture and thus can learn an adaptive (or optimal) graph representation for GCN learning. In particular, as shown in Figure 1, GLCN contains one graph learning layer, several convolution layers and one final perceptron layer. In the following, we explain them in detail.

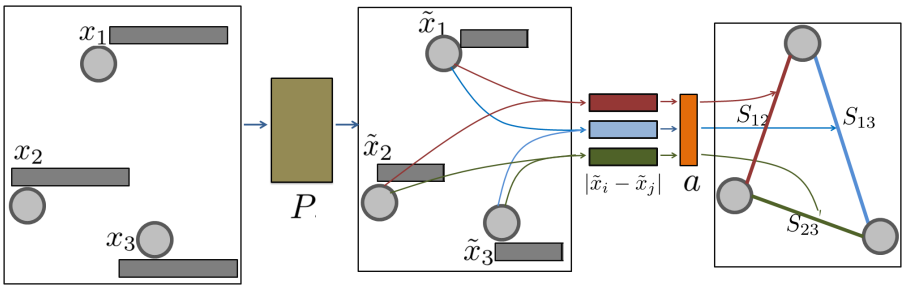


Figure 2. Architecture of the proposed graph learning architecture in GLCN.

3.1. Graph learning architecture

Given an input $X = (x_1, x_2 \cdots x_n) \in \mathbb{R}^{n \times p}$, we aim to seek a nonnegative function $S_{ij} = g(x_i, x_j)$ that represents the pairwise relationship between data x_i and x_j. We implement $g(x_i, x_j)$ via a single-layer neural network, which is parameterized by a weight vector $a = (a_1, a_2, \cdots a_p)^T \in \mathbb{R}^{p \times 1}$. Formally, we learn a graph S as

$$S_{ij} = g(x_i, x_j) = \frac{\exp(\text{ReLU}(a^T|x_i - x_j|))}{\sum_{j=1}^{n} \exp(\text{ReLU}(a^T|x_i - x_j|))} \quad (4)$$

where $\text{ReLU}(\cdot) = \max(0, \cdot)$ is an activation function, which guarantees the nonnegativity of S_{ij}. The role of the above softmax operation on each row of S is to guarantee that the learned graph S can satisfy the following property,

$$\sum_{j=1}^{n} S_{ij} = 1, S_{ij} \geq 0 \quad (5)$$

We optimize the optimal weight vector a by minimizing the following loss function,

$$\mathcal{L}_{\text{GL}} = \sum_{i,j=1}^{n} \|x_i - x_j\|_2^2 S_{ij} + \gamma \|S\|_F^2 \quad (6)$$

That is, larger distance $\|x_i - x_j\|_2$ between data point x_i and x_j encourages a smaller value S_{ij}. The second term is used to control the sparsity of learned graph S because of simplex property of S (Eq.(5)), as discussed in [17].

Remark. Minimizing the above loss $\mathcal{L}_{\text{GL}}$ independently may lead to trivial solution, i.e., $a = (0, 0 \cdots 0)$. We use it as a regularized term in our final loss function, as shown in Eq.(15) in §3.2.

For some problems, when an initial graph A is available, we can incorporate it in our graph learning as

$$S_{ij} = g(x_i, x_j) = \frac{A_{ij} \exp(\text{ReLU}(a^T|x_i - x_j|))}{\sum_{j=1}^{n} A_{ij} \exp(\text{ReLU}(a^T|x_i - x_j|))} \quad (7)$$

We can also incorporate the information of A by considering a regularized term in the learning loss function as

$$\mathcal{L}_{\text{GL}} = \sum_{i,j=1}^{n} \|x_i - x_j\|_2^2 S_{ij} + \gamma \|S\|_F^2 + \beta \|S - A\|_F^2 \quad (8)$$

On the other hand, when the dimension p of the input data X is large, the above computation of $g(x_i, x_j)$ may be less effective due to the long weight vector a needing to be trained. Also, the computation of Euclidean distances $\|x_i - x_j\|_2$ between data pairs in loss function $\mathcal{L}_{\text{GL}}$ is complex for large dimension p. To solve this problem, we propose to conduct our graph learning in a low-dimensional subspace. We implement this via a single-layer low-dimensional embedding network, parameterized by a projection matrix $P \in \mathbb{R}^{p \times d}, d < p$. In particular, we conduct our final graph learning as follows,

$$\tilde{x}_i = x_i P, \text{ for } i = 1, 2 \cdots n \quad (9)$$

$$S_{ij} = g(\tilde{x}_i, \tilde{x}_j) = \frac{A_{ij} \exp(\text{ReLU}(a^T|\tilde{x}_i - \tilde{x}_j|))}{\sum_{j=1}^{n} A_{ij} \exp(\text{ReLU}(a^T|\tilde{x}_i - \tilde{x}_j|))} \quad (10)$$

where A denotes an initial graph. If it is unavailable, we can set $A_{ij} = 1$ in the above update rule. The loss function becomes

$$\mathcal{L}_{\text{GL}} = \sum_{i,j=1}^{n} \|\tilde{x}_i - \tilde{x}_j\|_2^2 S_{ij} + \gamma \|S\|_F^2 \quad (11)$$

The whole architecture of the proposed graph learning network is shown in Figure 2.

Remark. The proposed learned graph S has a desired probability property (Eq.(5)), i.e., the optimal S_{ij} can be regarded a probability that data x_j is connected to x_i as a neighboring node. That is, the proposed graph learning (GL) architecture can establish the neighborhood structure of data automatically either based on data feature X only or by further incorporating the prior initial graph A with X. The GL architecture indeed provides a kind of nonlinear function $S = \mathcal{G}_{\text{GL}}(X, A; P, a)$ to predict/compute the neighborhood probabilities between node pairs.

3.2. GLCN architecture

The proposed graph learning architecture is general and can be incorporated in any graph CNNs. In this paper, we incorporate it into GCN [11] and propose a unified

Graph Learning-Convolutional Network (GLCN) for semi-supervised learning problem. Figure 1 shows the overview of GLCN architecture. The aim of GLCN is to learn an optimal graph representation for GCN network and integrate graph learning and convolution simultaneously to boost their respectively performance.

As shown in Figure 1, GLCN contains one graph learning layer, several graph convolution layers and one final perceptron layer. The graph learning layer aims to provide an optimal adaptive graph representation S for graph convolutional layers. That is, in the convolutional layers, it conducts the layer-wise propagation rule based on the adaptive neighbor graph S returned by graph learning layer, i.e.,

$$X^{(k+1)} = \sigma(D_s^{-1/2} S D_s^{-1/2} X^{(k)} W^{(k)}) \tag{12}$$

where $k = 0, 1 \cdots K-1$. $D_s = \text{diag}(d_1, d_2, \cdots d_n)$ is a diagonal matrix with diagonal element $d_i = \sum_{j=1}^{n} S_{ij}$. $W^{(k)} \in \mathbb{R}^{d_k \times d_{k+1}}$ is a layer-specific trainable weight matrix for each convolution layer. $\sigma(\cdot)$ denotes an activation function, such as $\text{ReLU}(\cdot) = \max(0, \cdot)$, and $X^{(k+1)} \in \mathbb{R}^{n \times d_{k+1}}$ denotes the output of activations in the k-th layer. Since the learned graph S satisfies $\sum_j S_{ij} = 1, S_{ij} \geq 0$, thus Eq.(12) can be simplified as

$$X^{(k+1)} = \sigma(S X^{(k)} W^{(k)}) \tag{13}$$

For semi-supervised classification task, we define the final perceptron layer as

$$Z = \text{softmax}(S X^{(K)} W^{(K)}) \tag{14}$$

where $W^{(K)} \in \mathbb{R}^{d_K \times c}$ and c denotes the number of classes. The final output $Z \in \mathbb{R}^{n \times c}$ denotes the label prediction of GLCN network, in which each row Z_i denotes the label prediction for the i-th node. The whole network parameters $\Theta = \{P, a, W^{(0)}, \cdots W^{(K)}\}$ are jointly trained by minimizing the following loss function as

$$\mathcal{L}_{\text{Semi-GLCN}} = \mathcal{L}_{\text{Semi-GCN}} + \lambda \mathcal{L}_{\text{GL}} \tag{15}$$

where $\mathcal{L}_{\text{GL}}$ and $\mathcal{L}_{\text{Semi-GCN}}$ are defined in Eq.(11) and Eq.(3), respectively. Parameter $\lambda \geq 0$ is a tradeoff parameter. It is noted that, when $\lambda = 0$, the optimal graph S is learned based on labeled data (i.e., cross-entropy loss) only which is also feasible in our GLCN.

Demonstration and analysis. There are two main benefits of the proposed GLCN network:

- In GLCN, both given labels Y and the estimated labels Z are incorporated and thus can provide useful 'weakly' supervised information to refine the graph construction S and thus to facilitate the graph convolution operation in GCN for unknown label estimation. That is, the graph learning and semi-supervised learning are conducted jointly in GLCN and thus can boost their respectively performance.
- GLCN is a unified network which can be trained via a single optimization manner and thus can be implemented simply.

Figure 3 shows the cross-entropy loss values over labeled node L across different epochs. One can note that, GLCN obtains obviously lower cross-entropy value than GCN at convergence, which clearly demonstrates the higher predictive accuracy of GLCN model. Also, the convergence speed of GLCN is just slightly slower than GCN, indicating the efficiency of GLCN. Figure 4 demonstrates 2D t-SNE [14] visualizations of the feature map output by the first convolutional layer of GCN [11] and GLCN, respectively. Different classes are marked by different colors. One can note that, the data of different classes are distributed more clearly and compactly in our GLCN representation, which demonstrates the desired discriminative ability of GLCN on conducting graph node representation and thus semi-supervised classification tasks.

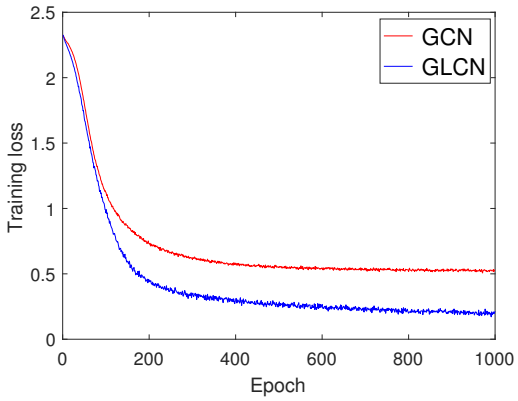


Figure 3. Demonstration of cross-entropy loss values across different epochs on MNIST dataset.

4. Experiments

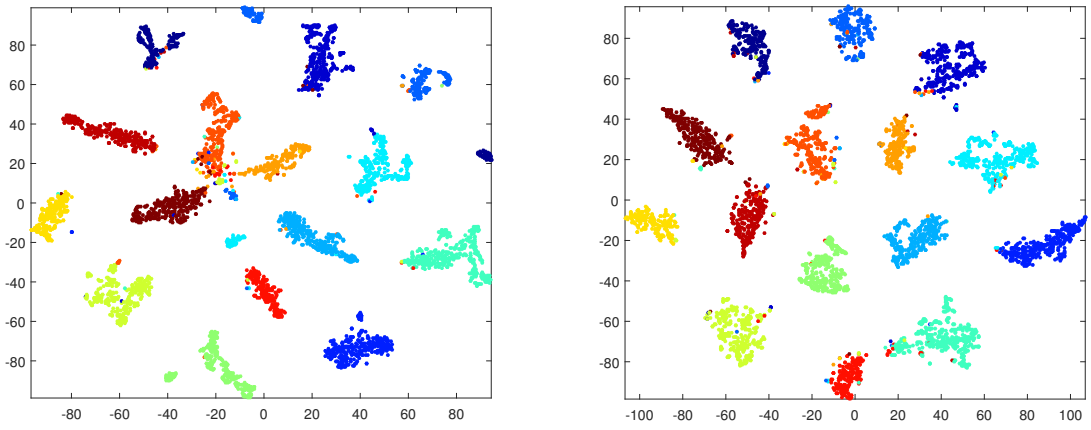
4.1. Datasets

To verify the effectiveness and benefit of the proposed GLCN on semi-supervised learning tasks, we test it on seven benchmark datasets, including three standard citation network benchmark datasets (Citeseer, Cora and Pubmed [20]) and four image datasets (CIFAR10 [12], SVHN [16], MNIST and Scene 15 [9]). The details of these datasets and their usages in our experiments are introduced below.

Citeseer. This network data contains 3327 nodes and 4732 edges. The nodes are falling into six classes and each node is represented by a 3703 dimension feature descriptor.

Cora. This data contains 2708 nodes and 5429 edges. Each node has a 1433 dimension feature descriptor and all the nodes are falling into six classes.

Pubmed. This data contains 19717 nodes and 44338 edges.



(a) GCN result (b) GLCN result

Figure 4. 2D t-SNE [14] visualizations of the feature map output by the first convolutional layer of GCN [11] and GLCN respectively on Scene15 dataset. Different classes are marked by different colors. One can note that, the data of different classes are distributed more clearly and compactly in our GLCN convolutional layer feature representation.

Each node is represented by a 500 dimension feature descriptor and all the nodes are classified into 3 classes.

CIFAR10. This dataset contains 50000 natural images which are falling into 10 classes [12]. Each image in this dataset is a 32×32 RGB color image. In our experiments, we select 1000 images for each class and use 10000 images in all for our evaluation. For each image, we extract a CNN feature descriptor for it.

SVHN. It contains 73257 training and 26032 test images [16]. Each image is a 32×32 RGB image which contains multiple number of digits and the task is to recognize the digit in the image center. Similar to CIFAR 10 dataset, in our experiments, we select 1000 images for each class and obtain 10000 images in all for our evaluation. For each image, we extract a CNN feature descriptor for it.

MNIST.[1] This dataset consists of images of hand-written digits from '0' to '9'. We randomly select 1000 images from each digit class and obtain 10000 images in all for our evaluation. Similar to other related works, we use gray value directly and convert it to a 784 dimension vector.

Scene15. It consists of 4485 scene images with 15 different categories [13, 9]. For each image, we use the feature descriptor provided by work [9].

4.2. Experimental setting

For Cora, Citeseer and Pubmed datasets, we follow the experimental setup of previous works [11, 21]. For image dataset CIFAR10, SVHN and MNIST, we randomly select 1000, 2000 and 3000 images as labeled samples and the remaining data are used as unlabeled samples. For unlabeled samples, we select 1000 images for validation purpose and use the remaining 8000, 7000 and 6000 images as test samples, respectively. All the reported results are averaged over ten runs with different splits of training, validation and testing data. For image dataset Scene15 [9], we randomly select 500, 750 and 1000 images as label data and use 500 images for validation, respectively. The remaining samples are used as the unlabeled test samples. The reported results are averaged over ten runs with different data splits.

Table 1. Comparison results of semi-supervised learning on dataset Citeseer, Cora and Pubmed.

Methond	Citeseer	Cora	Pubmed
ManiReg [2]	60.1%	59.5%	70.7%
LP [23]	45.3%	68.0%	63.0%
DeepWalk [18]	43.2%	67.2%	65.3%
GCN [11]	70.9%	82.9%	77.9%
GAT [21]	71.0%	83.2%	78.0%
GLCN	**72.0%**	**85.5%**	**78.3%**

Similar to [11], in our experiments we set the number of convolution layers in our GLCN to 2. The number of units in graph learning layer is set to 70 and it is set to 30 in graph convolution hidden layer. We train GLCN for a maximum of 5000 epochs (training iterations) using an ADAM algorithm [10] with learning rate 0.005, and stop training if the validation loss does not decrease for 100 consecutive epochs, as suggested in work [11]. All the network weights Θ are initialized using Glorot initialization [6].

4.3. Comparison with state-of-the-art methods

Baselines. We first compare our GLCN model with baseline GCN [11] which is the most related model with our GLCN. We also compare our method against some oth-

[1] http://yann.lecun.com/exdb/mnist/

Table 2. Comparison results on dataset SVHN, CIFAR, MNIST and Scene15

Dataset	SVHN			CIFAR		
No. of label	1000	2000	3000	1000	2000	3000
ManiReg [2]	69.44±0.69%	72.73±0.44%	74.63±0.45%	52.30±0.66%	57.08±0.80%	59.69±0.71%
LP [23]	69.68±0.84%	70.35±1.73%	69.47±2.96%	57.52±0.67%	59.22±0.67%	60.38±0.51%
DeepWalk [18]	74.64±0.23%	76.21±0.23%	77.04±0.42%	56.16±0.54%	59.73±0.35%	61.26±0.32%
GCN [11]	71.33±1.48%	73.43±0.46%	73.63±0.52%	60.43±0.56%	60.91±0.50%	60.99±0.49%
GAT [21]	73.87±0.32%	74.85±0.55%	75.17±0.43%	63.25±0.50%	65.55±0.58%	66.56±0.58%
GLCN	**79.14±0.38%**	**80.68±0.22%**	**81.43±0.34%**	**66.67±0.24%**	**69.33±0.54%**	**70.39±0.54%**
Dataset	MNIST			Scene15		
No. of label	1000	2000	3000	500	750	1000
ManiReg [2]	92.74±0.33%	93.96±0.23%	94.62±0.22%	81.29±3.35%	86.45±1.91%	89.86±0.71%
LP [23]	79.28±0.91%	81.91±0.82%	83.45±0.53%	89.40±4.74%	92.12±2.87%	92.98±2.45%
DeepWalk [18]	**94.55±0.27%**	95.04±0.28%	95.34±0.26%	95.64±0.24%	96.01±0.24%	96.53±0.37%
GCN [11]	90.59±0.26%	90.91±0.19%	91.01±0.23%	91.42±2.07%	94.41±0.92%	95.44±0.89%
GAT [21]	92.11±0.35%	92.64±0.28%	92.81±0.29%	93.98±0.75%	94.64±0.41%	95.03±0.46%
GLCN	94.28±0.28%	**95.09±0.17%**	**95.46±0.20%**	**96.19±0.38%**	**96.71±0.40%**	**96.67±0.37%**

er graph neural network based semi-supervised learning approaches which contain i) two traditional graph based semi-supervised learning methods including Label Propagation (LP) [23], Manifold Regularization (ManiReg) [2], and ii) three graph neural network methods including DeepWalk [18], Graph Convolutional Network (GCN) [11] and Graph Attention Networks (GAT) [21]. The codes of GCN and GAT have been provided by authors and we use them in our experiments. For fair comparison, the parameter settings of GCN are the same as our GLCN to obtain the average better performance on all datasets.

Results. Table 1 summarizes the comparison results on three citation network benchmark datasets (Citeseer, Cora and Pubmed [20])[2]. Table 2 summarizes the comparison results on four widely used image datasets. The best results are marked as bold in Table 1 and 2. Overall, we can note that (1) GLCN outperforms the baseline method GCN [11] on all datasets, especially on the four image datasets. This clearly demonstrates the higher predictive accuracy on semi-supervised classification of GLCN by incorporating graph learning architecture. Comparing with GCN, the hidden layer presentations of graph nodes in GLCN become more discriminatively (as shown in Figure 4), which thus facilitates to semi-supervised learning results. (2) GLCN performs better than recent graph network GAT [21], which indicates the benefit of GLCN on graph data representation and learning. (3) GLCN performs better than other graph based semi-supervised learning methods, such as LP [23], ManiReg [2] and DeepWalk [18], which further demonstrates the effectiveness of GLCN on conducting graph semi-supervised classification tasks.

[2]Note that, GCN results in Table 1 are slightly different from that reported in work [11], because we use a different hidden layer setting to make it consistent with GLCN and also obtain average better results on all the network and image datasets.

4.4. Parameter analysis

In this section, we evaluate the performance of GLCN model with different settings of network parameters. We first investigate the influence of model depth of GLCN (number of convolutional layers) on semi-supervised classification results. Figure 5 shows the performance of our GLCN method across different number of convolutional layers on MNIST dataset. As a baseline, we also list the results of GCN model with the same setting. One can note that GLCN can obtain better performance with different number of layers, which indicates the insensitivity of the GLCN w.r.t. model depth. Also, GLCN always performs better than GCN under different model depths, which further demonstrates the benefit and better performance of GLCN comparing with the baseline method.

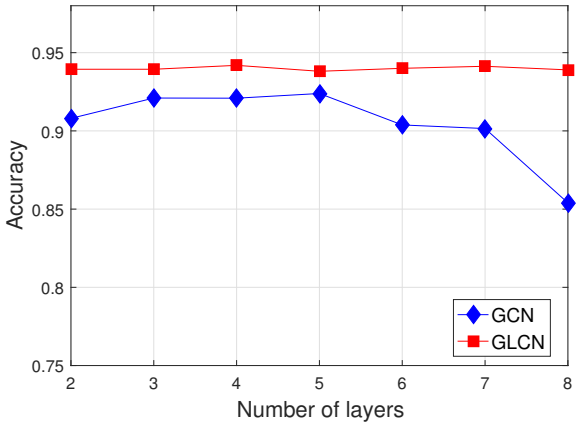


Figure 5. Results of GLCN across different convolutional layers on MNIST dataset.

Table 3. Results of two-layer GLCN across different number of units in convolutional-layer on MNIST dataset.

GCN-Layers	50	60	70	80	90
GCN	0.9041	0.9075	0.9080	0.9076	0.9070
GLCN	0.9410	0.9396	0.9394	0.9410	0.9389

Table 4. Results of GLCN with different settings of graph learning parameter λ in loss (Eq.(15)) on MNIST and CIFAR10 datasets

Parameter λ	0	1e-4	1e-3	1e-2	1e-1	1.0
CIFAR10	0.67	0.69	0.69	0.70	0.69	0.69
MNIST	0.92	0.93	0.93	0.94	0.94	0.93

Then, we evaluate the performance of two-layer GLCN with different number of hidden units in convolutional layer. Table 3 summarizes the performance of GLCN with different number of hidden units on MNIST dataset. We can note that Both GCN and GLCN are generally insensitive w.r.t. number of units in the hidden layer.

Finally, we investigate the influence of graph learning parameter λ in our GLCN. Table 4 shows the performance of GLCN with different parameter settings. When λ is set to 0, GLCN can also return a reasonable result. Also, one can note that the graph learning regularization term in loss function improves the graph learning and thus semi-supervised classification results.

5. Conclusion and Future Works

In this paper, we propose a novel Graph Learning-Convolutional Network (GLCN) for graph based semi-supervised learning problem. GLCN integrates the proposed new graph learning operation and traditional graph convolution architecture together in a unified network, which can learn an optimal graph structure that best serves GCN for semi-supervised learning problem. Experimental results on seven benchmarks demonstrate that GLCN generally outperforms traditional fixed-graph CNNs on various semi-supervised learning tasks.

Note that, GLCN is not limited to deal with semi-supervised learning tasks. In the future, we will adapt GLCN on some more pattern recognition tasks, such as graph data clustering, graph link/edge prediction, etc. Also, we can explore GLCN method on some other computer vision tasks, such as visual object detection, image co-segmentation and visual saliency analysis.

Acknowledgment

This work is supported in part by NSFC Key Projects of International (Regional) Cooperation and Exchanges under Grant (61860206004); National Natural Science Foundation of China (61602001, 61872005, 61671018); Natural Science Foundation of Anhui Province (1708085QF139); Natural Science Foundation of Anhui Higher Education Institutions of China (KJ2016A020).

References

[1] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1993–2001, 2016.

[2] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7:2399–2434, 2006.

[3] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations*, 2014.

[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[5] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in Neural Information Processing Systems*, pages 2224–2232, 2015.

[6] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256, 2010.

[7] Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[8] Bo Jiang, Chris Ding, Bio Luo, and Jin Tang. Graph-laplacian pca: Closed-form solution and robustness. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3492–3498, 2013.

[9] Zhuolin Jiang, Zhe Lin, and Larry S Davis. Label consistent k-svd: Learning a discriminative dictionary for recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2651–2664, 2013.

[10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[11] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[12] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[13] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2169–2178, 2006.

[14] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9:2579–2605, 2008.

[15] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5423–5434, 2017.

[16] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, 2011.

[17] Feiping Nie, Xiaoqian Wang, and Heng Huang. Clustering and projected clustering with adaptive neighbors. In *ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pages 977–986, 2014.

[18] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[19] Feiyun Zhu Junzhou Huang Ruoyu Li, Sheng Wang. Adaptive graph convolutional neural networks. In *AAAI Conference on Artificial Intelligence*, pages 3546–3553, 2018.

[20] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[21] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[22] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: Algorithms, applications and open challenges. In *International Conference on Computational Social Networks*, pages 79–91, 2018.

[23] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.