MR-GCN: Multi-Relational Graph Convolutional Networks based on Generalized Tensor Product

Zhichao Huang[1,2*], **Xutao Li**[1,2*†], **Yunming Ye**[1,2†] and **Michael K. Ng**[3]
[1]Harbin Institute of Technology, Shenzhen
[2]Shenzhen Key Laboratory of Internet Information Collaboration
[3]Department of Mathematics, The University of Hong Kong
iceshzc@stu.hit.edu.cn, {lixutao, yeyunming}@hit.edu.cn, mng@maths.hku.hk

Abstract

Graph Convolutional Networks (GCNs) have been extensively studied in recent years. Most of existing GCN approaches are designed for the homogenous graphs with a single type of relation. However, heterogeneous graphs of multiple types of relations are also ubiquitous and there is a lack of methodologies to tackle such graphs. Some previous studies address the issue by performing conventional GCN on each single relation and then blending their results. However, as the convolutional kernels neglect the correlations across relations, the strategy is sub-optimal. In this paper, we propose the Multi-Relational Graph Convolutional Network (MR-GCN) framework by developing a novel convolution operator on multi-relational graphs. In particular, our multi-dimension convolution operator extends the graph spectral analysis into the eigen-decomposition of a Laplacian tensor. And the eigen-decomposition is formulated with a generalized tensor product, which can correspond to any unitary transform instead of limited merely to Fourier transform. We conduct comprehensive experiments on four real-world multi-relational graphs to solve the semi-supervised node classification task, and the results show the superiority of MR-GCN against the state-of-the-art competitors.

1 Introduction

In recent years, Graph Convolutional Networks (GCNs) have been extensively studied, which can be applied to many graph applications, such as node classification [Kipf and Welling, 2016; Hamilton *et al.*, 2017], link prediction [Schlichtkrull *et al.*, 2018] and personalized recommendation [Ying *et al.*, 2018]. In terms of convolution operation manners, there are two types of GCNs, namely *spatial* methods [Hamilton *et al.*, 2017; Jørgensen *et al.*, 2018] and *spectral* methods [Henaff *et al.*, 2015; Defferrard *et al.*, 2016; Kipf and Welling, 2016]. Despite of extensive literatures, most of them work merely on homogenous graphs with a single type of relation and cannot tackle multi-relational graphs.

Recently, some remedies are put forward to address the problem, which fall into two lines in terms of their strategies. The first line conducts GCN on each single relation and then integrates the results with multi-view learning [Schlichtkrull *et al.*, 2018; Ma *et al.*, 2019; Sun *et al.*, 2019]. In such methods, relation correlations are not effectively exploited as GCN basis is independently constructed on each graph. Another line is aggregating the multi-relational graph into a homogeneous graph [Khan and Blumenstock, 2019; Yun *et al.*, 2019]. For example, Multi-GCN [Khan and Blumenstock, 2019] adopts manifold ranking to achieve the aggregation, and then applies a standard GCN on the aggregated graph. The type of methods may result in information loss or noises. Hence, both strategies are sub-optimal. The reason roots at that the convolution operation on multi-relational graphs is not defined and existing studies bypass the issue.

In this paper, we propose to define the convolution operator in multi-relational graphs, upon which a neural network framework, termed as Multi-Relational Graph Convolutional Network (MR-GCN), can be established. Specifically, a Laplacian tensor is first constructed for a given multi-relational graph. The tensor based eigenvalue decomposition is formulated, which offers the basis to develop multi-relational graph convolution operators (MR-GCO). As the tensor eigen-decomposition can be defined with a generalized tensor product, the basis of MR-GCO can be any unitary transform, *e.g.,* Haar, Discrete Cosine transform (DCT), instead of limited merely to Fourier transform. To validate the effectiveness of the proposed MR-GCN framework, we empirically evaluate the performance on node classification task in multi-relational graphs. The main contributions of the paper can be summarized as follows:

- We define the convolution operator in multi-relational graphs based on the eigen-decomposition of Laplacian tensors. Upon the convolution operator, a neural network framework, namely MR-GCN, is established for node classification. To the best of our knowledge, we are the first to extend GCNs spectral graph theory by considering tensor eigen-decomposition.
- Different from conventional GCNs, which perform convolutions in the spectral domain with discrete Fourier

*Co-first authors with equal contributions.
†Corresponding authors.

transform (DFT), our MR-GCN is established on the eigen-decomposition defined by generalized tensor product. Hence, the spectral domain can be any unitary transform, *e.g.,* Haar, DCT, etc, instead of limited to DFT.

- Comprehensive experimental evaluations have been conducted on four real-world multi-relational graphs, and the results show that the developed MR-GCN framework outperforms state-of-the-art competitors.

2 Related Work

2.1 Graph Convolutional Networks

GCNs are increasingly developed for a wide range of tasks. According to convolution operation manners, they can be roughly categorized into two types, namely spectral methods [Bruna *et al.*, 2013; Henaff *et al.*, 2015; Defferrard *et al.*, 2016; Kipf and Welling, 2016] and spatial methods [Hamilton *et al.*, 2017; Monti *et al.*, 2017; Jørgensen *et al.*, 2018]. [Bruna *et al.*, 2013] introduce a graph based convolution operator in spectral domain based on Fourier basis. Then, [Kipf and Welling, 2016] simplify the convolution via a localized first-order approximation. In contrast, spatial approaches utilize spatially close neighbors to define convolution operations. For instance, [Hamilton *et al.*, 2017] introduce a learnable aggregating function to summarize neighbors' information for node representations.

However, most of previous GCN methods are designed for single relational graphs and cannot tackle multi-relational ones. Though several multi-relational version of GCNs appear in recent studies [Schlichtkrull *et al.*, 2018; Ma *et al.*, 2019; Sun *et al.*, 2019; Khan and Blumenstock, 2019; Yun *et al.*, 2019], their solutions are still conventional GCN in essence. The key obstacle stands in between GCN and multi-relational graphs is that a multi-graph convolution operator remains undefined, which is our focus in this paper.

2.2 Tensor Product

Tensor, also known as *n-way* or *n-mode* array, refers to a multi-dimensional array of numbers. It is a powerful tool to represent and analyze multi-dimensional data [Ng *et al.*, 2011; Li *et al.*, 2013; Li *et al.*, 2017]. Conventional tensor product means mode-n multiplication. However, in this definition, the eigen-decomposition of a tensor cannot be well defined. In [Kilmer and Martin, 2011], a novel tensor product, termed as $t\text{-}product$ is introduced, which is indeed a generalized matrix multiplication, where each element denotes a tube and Fourier convolution is adopted because each element is a vector rather than a scalar in conventional matrix. Based on the product, a tensor singular value decomposition (t-svd) is formulated. Very recently, [Song *et al.*, 2019] further generalize the t-product by relaxing the DFT into any unitary transform. In this paper, we aim to define convolution operator in multi-relational graphs with the generalized tensor product.

3 Proposed Framework

In this section, we introduce the proposed MR-GCN framework. First, we revisit the basic concepts of graph convolutions and some essential definitions of generalized tensor product (GTP) as preliminaries. Then we present how the MR-GCN operator is developed and finally discuss to build the semi-supervised node classifier upon the operator.

3.1 Preliminaries

Notations

In this paper, we define the fields of real and complex number with respect to $\mathbb{R}$ and $\mathbb{C}$. Tensors and matrices are symbolized by Euler and boldface capital letters, respectively. Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ be a third-order tensor, we represent its (*i, j, k*)-th entry as $\mathcal{A}_{ijk}$ and employ matlab notations $\mathcal{A}(i, :, :)$, $\mathcal{A}(:, i, :)$ and $\mathcal{A}(:, :, i)$ to denote the i-th horizontal, lateral and frontal slices, respectively. For simplicity, we also denote the frontal slice $\mathcal{A}(:, :, i)$ as $\mathcal{A}^{(i)}$.

Graph Convolution Operator

The core idea of GCN is to define convolution operator based on graph structures. To this end, the graph Fourier transform is first formulated by replacing the eigen basis of continuous Laplacian in Fourier transform with that of graph Laplacian. Then, the convolution operator on the graph can be easily defined following the convolution theorem. Given a graph G with an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$, we calculate its normalized graph Laplacian as $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}$, where $\boldsymbol{I}$ is the identity matrix and $\boldsymbol{D}$ represents a diagonal degree matrix with entries $\boldsymbol{D}_{ii} = \sum_j \boldsymbol{A}_{ij}$. Then the convolution of a input signal $x \in \mathbb{R}^N$ with a filter $g \in \mathbb{R}^N$ on G is defined as:

$$x \star_G g = \boldsymbol{U}^T(\boldsymbol{U}x \odot \boldsymbol{U}g), \tag{1}$$

where $\star_G$ denotes the graph convolution operator, $\odot$ indicates the Hadamard product which is element-wise multiplication and $\boldsymbol{U}$ refers to the matrix of eigenvectors of the normalized graph Laplacian, which is obtained by its eigenvalue decomposition $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$. $\boldsymbol{U}x$ and $\boldsymbol{U}g$ are the graph Fourier transforms of x and g.

Generalized Tensor Product

To well define the eigen-decomposition of tensors, $t\text{-}product$ is formulated in [Kilmer and Martin, 2011]. In the product, third-order tensors are considered as a matrix with each element to be a vector, instead of a scalar. By analogue with matrix multiplication, t-product is defined and vector-vector interaction is achieved by *circular convolution*, which corresponds to DFT. In [Song *et al.*, 2019], DFT is further extended to any unitary transform and a generalized tensor product (GTP), *i.e.,* $\boldsymbol{\Phi}\text{-}product$, is introduced. Next, we briefly introduce the GTP.

Let $\mathcal{A} \in \mathbb{C}^{n_1 \times n_2 \times n_3}$ be a third-order tensor, and $\boldsymbol{\Phi} \in \mathbb{C}^{n_3 \times n_3}$ be the unitary transform matrix, *i.e.,* $\boldsymbol{\Phi}\boldsymbol{\Phi}^H = \boldsymbol{\Phi}^H\boldsymbol{\Phi} = \boldsymbol{I}$. Here $\boldsymbol{\Phi}^H$ denotes the Hermitian transpose of $\boldsymbol{\Phi}$. We let $\boldsymbol{\Phi}[\mathcal{A}]$ represent the transformed tensor (also denoted as $\hat{\mathcal{A}}_{\boldsymbol{\Phi}}$), which is obtained by applying the transform $\boldsymbol{\Phi}$ to each tube along the third dimension. That is, we have $vec\left(\hat{\mathcal{A}}_{\boldsymbol{\Phi}}(i, j, :)\right) = \boldsymbol{\Phi}\left[vec\left(\mathcal{A}(i, j, :)\right)\right]$, where $vec(\cdot)$ refers to mapping the tensor tube to a vector. Thanks to the unitary property of $\boldsymbol{\Phi}$, we have $\mathcal{A} = \boldsymbol{\Phi}^H\boldsymbol{\Phi}[\mathcal{A}] = \boldsymbol{\Phi}^H[\hat{\mathcal{A}}_{\boldsymbol{\Phi}}]$. Let $blockdiag(\mathcal{A})$ be a $n_1 n_3$-by-$n_2 n_3$ block diagonal matrix, obtained by putting the

frontal slices of $\mathcal{A}$ into the diagonal, and $fold$ indicate its inverse operator, namely $\mathcal{A} = fold\left(blockdiag\left(\mathcal{A}\right)\right)$. With the notations, the conjugate transpose of $\mathcal{A}$ is defined as:

$$\mathcal{A}^H = \mathbf{\Phi}^H \left[fold\left(bolckdiag\left(\hat{\mathcal{A}}_{\mathbf{\Phi}}\right)^H \right) \right]. \quad (2)$$

Given two third-order tensors $\mathcal{A} \in \mathbb{C}^{n_1 \times n_2 \times n_3}$, $\mathcal{B} \in \mathbb{C}^{n_2 \times n_4 \times n_3}$, the $\mathbf{\Phi}$-product between them is defined as follow:

$$\mathcal{C} = \mathcal{A} \diamond_{\mathbf{\Phi}} \mathcal{B} = \mathbf{\Phi}^H \left[fold\left(blockdiag\left(\hat{\mathcal{A}}_{\mathbf{\Phi}}\right) \times blockdiag\left(\hat{\mathcal{B}}_{\mathbf{\Phi}}\right) \right) \right], \quad (3)$$

where $\diamond_{\mathbf{\Phi}}$ represents the generalized tensor product, $\times$ is the standard matrix product and its result is $\mathcal{C} \in \mathbb{C}^{n_1 \times n_4 \times n_3}$.

3.2 Multi-Relational Graph Convolution Operator

Let MG be an undirected multi-relational graph and its topology is characterized by an adjacency tensor $\mathcal{A} \in \mathbb{R}^{N \times N \times R}$, where N and R represent the numbers of nodes and relations, respectively. We construct a diagonal degree tensor $\mathcal{D} \in \mathbb{R}^{N \times N \times R}$, where $\mathcal{D}_{iik} = \sum_j \mathcal{A}_{ijk}$ and $\mathcal{D}_{ijk} = 0$ for $\forall i \neq j$. Then, a normalized Laplacian tensor $\mathcal{L} \in \mathbb{R}^{N \times N \times R}$ is constructed and each of its frontal slice $\mathcal{L}^{(i)}$ is computed as:

$$\mathcal{L}^{(i)} = \boldsymbol{I} - \mathcal{D}^{(i)^{-\frac{1}{2}}} \mathcal{A}^{(i)} \mathcal{D}^{(i)^{-\frac{1}{2}}}, \quad (4)$$

where $\boldsymbol{I} \in \mathbb{R}^{N \times N}$ is the identity matrix.

To extend the graph convolution theory into the multi-relational case, we need perform eigen-decomposition on the Laplacian tensor $\mathcal{L}$. Unlike the matrix, which has a standard eigen-decomposition, tensors require to know product definitions in advance. Here we adopt the $\mathbf{\Phi}$-product presented in preliminaries. The choice has two important advantages. First, it can nicely exploit the inherent correlations across relations. To understand this point, we revisit the GTP and reveal its essence. Figure 1 presents an example of GTP computation. We can see that the product of two tensors $\mathcal{A}$ and $\mathcal{B}$ is computed as the following three steps: (i) converting the tensors into $\mathbf{\Phi}$ space (*i.e.,* $\hat{\mathcal{A}}_{\mathbf{\Phi}}$ and $\hat{\mathcal{B}}_{\mathbf{\Phi}}$); (ii) performing slice-by-slice matrix multiplication; (iii) transforming the result back to the original space with $\mathbf{\Phi}^H$. The steps follow the definition in Eq. 3. By convolution theorem, the process suggests that the product can be written as:

$$\left(\mathcal{A} \diamond_{\mathbf{\Phi}} \mathcal{B}\right)(i,j,:) = \sum_{k=1}^{n_2} \mathcal{A}(i,k,:) \circ_{\mathbf{\Phi}} \mathcal{B}(k,j,:). \quad (5)$$

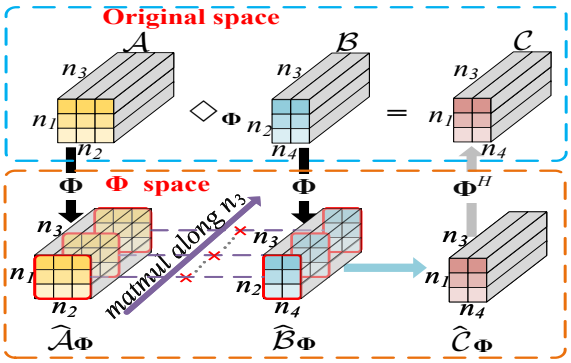


Figure 1: GTP computation process.

Algorithm 1 Tensor Based Eigenvalue Decomposition

Input: $\mathcal{L} \in \mathbb{C}^{N \times N \times R}$, $\mathbf{\Phi} \in \mathbb{C}^{R \times R}$.

1: $\hat{\mathcal{L}}_{\mathbf{\Phi}} = \mathbf{\Phi}[\mathcal{L}]$
2: **for** $i = 1, ..., R$ **do**
3: [**U**, **S**] = EVD($\hat{\mathcal{L}}^{(i)}_{\mathbf{\Phi}}$);
4: $\hat{\mathcal{U}}^{(i)}_{\mathbf{\Phi}} = \mathbf{U}$, $\hat{\mathcal{S}}^{(i)}_{\mathbf{\Phi}} = \mathbf{S}$;
5: **end for**
6: $\mathcal{U} = \mathbf{\Phi}^H[\hat{\mathcal{U}}_{\mathbf{\Phi}}]$, $\mathcal{S} = \mathbf{\Phi}^H[\hat{\mathcal{S}}_{\mathbf{\Phi}}]$

Output: $\mathcal{U} \in \mathbb{C}^{N \times N \times R}$, $\mathcal{S} \in \mathbb{C}^{N \times N \times R}$.

Here $\circ_{\mathbf{\Phi}}$ denotes a convolution defined by $\mathbf{\Phi}$. For example, when considering t-product, $\mathbf{\Phi}$ is the DFT and $\circ_{\mathbf{\Phi}}$ corresponds to the circular convolution. Thanks to the convolution along the relation dimension, the eigentensor of $\mathcal{L}$ defined on $\mathbf{\Phi}$-product can nicely model the relation correlations. Second, $\mathbf{\Phi}$-product can correspond to any unitary transform (*e.g.,* DCT, Haar, etc), not limited to DFT as t-product. Next, we introduce how to perform tensor based eigenvalue decomposition (TEVD) for the Laplacian tensor $\mathcal{L}$.

Theorem 1. *Let $\mathcal{L} \in \mathbb{C}^{N \times N \times R}$ be a third-order Laplacian tensor, where each frontal slice is symmetric and semi-definite. We define TEVD along the third dimension with respect to $\mathbf{\Phi}$-product and factorize $\mathcal{L}$ as follow:*

$$\mathcal{L} = \mathcal{U} \diamond_{\mathbf{\Phi}} \mathcal{S} \diamond_{\mathbf{\Phi}} \mathcal{U}^H. \quad (6)$$

Here $\mathcal{U} \in \mathbb{C}^{N \times N \times R}$ denotes the eigentensor and $\mathcal{U} \diamond_{\mathbf{\Phi}} \mathcal{U}^H = \mathcal{U}^H \diamond_{\mathbf{\Phi}} \mathcal{U} = \mathcal{I}_{\mathbf{\Phi}}$, where $\mathbf{\Phi}[\mathcal{I}_{\mathbf{\Phi}}]$ is a tensor with each frontal slice being the $N \times N$ identity matrix. $\mathcal{S} \in \mathbb{C}^{N \times N \times R}$ is a diagonal tensor with eigenvalues.

The TEVD computation is summarized in Algorithm 1, which is very simple. Its main idea is to convert each frontal slice of tensor $\mathcal{L}$ into $\mathbf{\Phi}$-space, perform eigen-decomposition, fold them into tensors $\hat{\mathcal{U}}_{\mathbf{\Phi}}$ and $\hat{\mathcal{S}}_{\mathbf{\Phi}}$ and then transform the two tensors back into the original space.

Definition 1. *Given a normalized Laplacian tensor $\mathcal{L} \in \mathbb{R}^{N \times N \times R}$ of an undirected multi-relational graph MG, a convolution operator of the input signal $x \in \mathbb{R}^{N \times R}$ with the filter $g \in \mathbb{R}^{N \times R}$ on MG can be defined as follow:*

$$x \star_{MG} g = \mathcal{U}^H \diamond_{\mathbf{\Phi}} \left[(\mathcal{U} \diamond_{\mathbf{\Phi}} x) \odot (\mathcal{U} \diamond_{\mathbf{\Phi}} g) \right], \quad (7)$$

where $\star_{MG}$ denotes the multi-relational graph convolution operator (MR-GCO), $\odot$ refers to the element-wise Hadamard product. And $\mathcal{U} \diamond_{\mathbf{\Phi}} x$ and $\mathcal{U} \diamond_{\mathbf{\Phi}} g$ represent the transformed

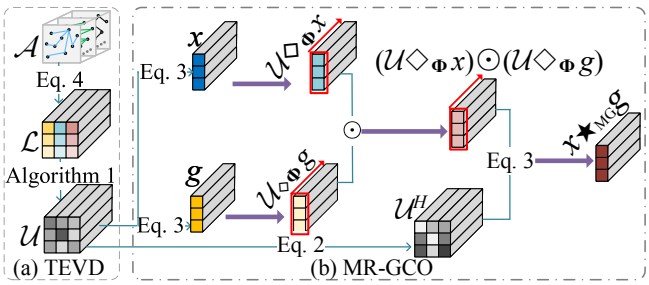


Figure 2: An illustration of MR-GCO with $\mathbf{\Phi}$.

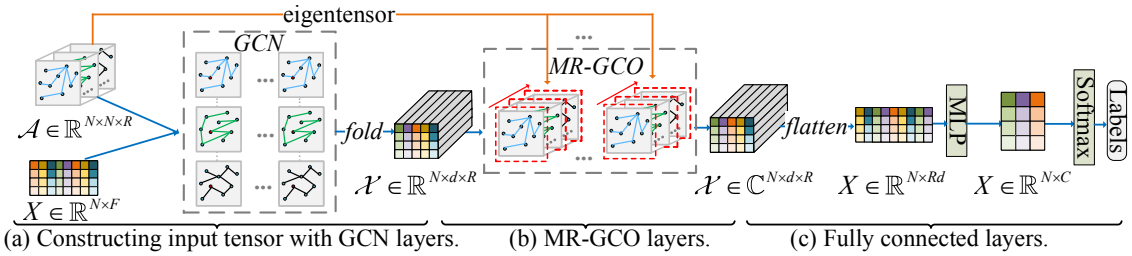


Figure 3: Semi-supervised learning with MR-GCN.

tensors based on $\mathcal{U}$, which is the eigentensor of Laplacian tensor $\mathcal{L}$ with $\mathbf{\Phi}$.

To better understand the MR-GCO definition, we illustrate its computation process in Figure 2. Given a graph with adjacency tensor $\mathcal{A}$, we construct its normalized Laplacian tensor $\mathcal{L}$ and compute the corresponding eigenvalue decomposition by Algorithm 1. As a result, we obtain the eigentensor $\mathcal{U}$ as the transform basis. According to the GTP definition in Eq. 3, the signal x and filter g are transformed into a new space by $\mathcal{U}$, and then their Hadamard product result is obtained. With Eq. 2, we compute $\mathcal{U}^H$, which is the conjugate transpose tensor of $\mathcal{U}$. Finally, the Hadamard product result is transformed back by $\mathcal{U}^H$. The MR-GCO can better model the relation correlations than multi-view GCNs, *e.g.,* R-GCN [Schlichtkrull *et al.*, 2018], because (i) the eigentensor basis is not computed independently for each view, but as a whole (See Eq. 6); (ii) the signal x is convolved with filter g based on the whole eigentensor basis (See Eq. 7), instead of on each view basis as multi-view GCNs.

3.3 Semi-supervised Learning with MR-GCN

The developed MR-GCO can be utilized in any multi-relational graph tasks. In this paper, we adopt it to build a semi-supervised node classifier. The architecture of the classifier is shown as in Figure 3. We can see that it consists of three key components, which are conventional GCN layers, MR-GCO layers and fully connected layers. Let us elaborate the three components, respectively. Given a multi-relational graph $\mathcal{A} \in \mathbb{R}^{N\times N\times R}$, we let a matrix $\boldsymbol{X} \in \mathbb{R}^{N\times F}$ denote its node features, where F is the feature size. As the input signal $\boldsymbol{X}$ has an inappropriate size for MR-GCO, we apply two-layer GCN [Kipf and Welling, 2016] in each relational graph with d-dimensional output as:

$$\boldsymbol{X}^{(l+1)} = \sigma(\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-\frac{1}{2}}\boldsymbol{X}^{(l)}\boldsymbol{W}^{(l)}), \tag{8}$$

where $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ is the adjacency matrix $\boldsymbol{A}$ of the graph G with added self-connections, $\tilde{\boldsymbol{D}}$ is the diagonal degree matrix of $\tilde{\boldsymbol{A}}$ and $\boldsymbol{W}^{(l)}$ is the matrix of filter parameters. We note the layers can be in any form, not necessary to be GCN. By folding the its outputs, we obtain a tensor signal $\mathcal{X} \in \mathbb{R}^{N\times d\times R}$ and feed it into the second component. After several layers of MR-GCO, the tensor signal is further transformed but its size does not change. Finally, in the third component, the tensor signal is flattened into a matrix and a simple multi-layer perceptron (MLP) as well as a softmax layer is appended. In the task, we adopt the cross-entropy loss.

4 Experiments

4.1 Experimental Setup

Datasets

In this paper, we adopt node classification to evaluate the performance. The following four real-world multi-relational graphs are utilized and Table 1 summarizes the statistics of them.

- **ACM**[1] is an academic multi-relational graph. In the graph, each node is a paper and two types of connections are considered, namely *paper coauthor edge* and *paper subject edge*. Bag-of-word representation of each paper is given as its node attribute, and its category is regarded as node label. We follow the settings of the raw dataset for training, validation and test separation.
- **IMDB**[1] is a multi-relational graph from movie dataset, which also includes two types of relations. Each movie is a node and its description is also changed into bag-of-word representation. The movie genre is the node label. Similarly, the raw dataset provides the training, validation and test sets.
- **Amazon**[2] is a multi-relational graph from Amazon purchase record. We employ the metadata of Electronics category and construct two types of connections between products, namely *co-viewed* or *co-purchased*. Node feature is composed one-hot encoding of its category list and its price information. The main category of each product is treated as node label. We randomly select 20 products from each category as our training set, and validation and test sets consist of 500 and 1,000 instances, respectively.
- **Reuters**[3] is a graph constructed from a multilingual dataset. The dataset contains 1,200 documents over six labels and is described by five views of 2,000 words. In the graph, each document is a node. We treat its English view as node attribute and construct four connections based on the other views. Specifically, two document nodes are connected in a view if their normalized text similarity is larger than 0.5. We follow the similar way on Amazon to construct the training, validation and test sets.

[1]https://drive.google.com/file/d/1qOZ3QjqWMIIvWjzrIdRe3EA4iKzPi6S5/view

[2]http://deepyeti.ucsd.edu/jianmo/amazon/index.html

[3]http://lig-membres.imag.fr/grimal/data.html (*Sample 1 is used*)

Dataset	#Nodes	#Relation 1	#Relation 2	#Relation 3	#Relation 4	#Classes	#Features	#Training	#Validation	#Test
ACM	8,994	9,936	3,025	-	-	3	1,902	600	300	2,125
IMDB	12,772	4,661	13,983	-	-	3	1,256	300	300	2,339
Amazon	14,810	29,808	13,675	-	-	5	850	100	500	1,000
Reuters	1,200	3,216	2,956	2,532	2,398	6	2000	120	100	500

Table 1: The statistics of four datasets in the experiments.

Baselines

To test the performance of the proposed MR-GCN, we compare it with graph embedding and state-of-the-art GCN approaches. Specifically, three graph embedding methods are adopted as baselines (*i.e.,* **LINE** [Tang *et al.*, 2015], **DeepWalk** [Perozzi *et al.*, 2014] and **Node2Vec** [Grover and Leskovec, 2016]). As the three methods are all unsupervised, we also consider a semi-supervised attributed graph embedding method, namely **Planetoid** [Yang *et al.*, 2016], and its inductive version is leveraged. **GCN** [Kipf and Welling, 2016] is a robust and popular approach, we include it as a baseline as well. We note that all the above graph embedding and GCN approaches are developed for single relational graphs. To compare with them, we report their performance on each single relational graph and a union graph whose adjacency matrix is an addition of all the adjacency matrices of relations. Very recently, two new graph neural network (GNN) approaches, heterogeneous graph attention network (**HAN**) [Wang *et al.*, 2019] and graph transformer network (**GTN**) [Yun *et al.*, 2019], are proposed for multi-relational graphs. They both generate new graph structures based on meta-paths, where HAN learns the importance of fixed meta-paths via an attentional graph neural network and GTN automatically learns meta-paths. We also adopt the two approaches as baseline methods.

Implementation Details

To make a fair comparison, we set the embedding dimension as 64 and adopt *Adam* optimizer for all approaches. The implementations of LINE, DeepWalk and Node2Vec are obtained from OpenNE[4] and the ones of all the other baselines are provided by their authors. For HAN and GTN, we adopt the meta-path setting as [Yun *et al.*, 2019] on ACM and IMDB, and initialize meta-paths for the other datasets with adjacency matrices. We tune all the baseline methods based on the validation sets.

In the proposed MR-GCN, we set the maximum of epochs to be 500 and employ the early stop strategy. The window size and dropout rate are set as 10 and 0.5, respectively. The other hyperparameters, learning rate from {0.01, 0.005, 0.0025, 0.001}, layer numbers from {1, 2, 3, 4, 5}, l2 loss weight decay parameter from {5e-2, 5e-4, 1e-8}, are tuned based on validation sets. Finally, we set the learning rate as 0.001 on ACM and Reuters, 0.0025 on IMDB and 0.005 on Amazon. We adopt 1 layer MR-GCO on Amazon and IMDB, 2 layer MR-GCO on Reuters and 4 layer MR-GCO on ACM. The best optimal l2 loss weight decay parameters are 5e-4 on ACM and Reuters, 1e-8 on IMDB and 5e-2 on Amazon.

[4]https://github.com/thunlp/OpenNE

Method	Type	Datasets		
		ACM	IMDB	Amazon
LINE	rel. 1	39.76	37.49	36.2
	rel. 2	61.13	42.45	31.0
	union	63.62	38.44	45.0
DeepWalk	rel. 1	66.26	40.87	43.7
	rel. 2	68.14	46.77	37.6
	union	79.91	48.95	54.1
Node2Vec	rel. 1	67.39	48.99	45.4
	rel. 2	68.75	47.24	35.8
	union	81.08	50.02	54.2
Planetoid	rel. 1	87.95	54.47	73.7
	rel. 2	86.54	54.47	73.8
	union	87.81	54.51	73.9
GCN	rel. 1	91.34	52.16	75.2
	rel. 2	80.94	54.55	74.4
	union	91.57	56.56	76.3
HAN	multi	89.93	56.82	62.3
GTN	multi	91.01	60.45	75.7
MR-GCN	multi	**93.22**	**61.65**	**77.8**

Table 2: Classification accuracies on two-relational datasets (in percent). Best results are in bold.

As our MR-GCN can be equipped with any unitary transform, we select six well-known transforms to validate its effectiveness, which are discrete Cosine, Walsh-Hadamard, Fourier, Hartely, Haar and Slant [Jain, 1979].

4.2 Results and Discussions

Main Results

Tables 2 and 3 report the classification performance. As if there are only two types of relations, the six unitary transform matrices are exactly the same. Hence, Table 2 shows the results on ACM, IMDB and Amazon which include two relations and Table 3 summarizes the results on Reuters that has four relations.

According to Table 2, we make following conclusions: (i) Our MR-GCN yields the best performance on the three two-relational graphs. This is attributed to our developed multi-relational convolution operator. In the operator, the tensor based eigenvalue decomposition is conducted to construct the transform basis, which nicely exploits the inherent correlations across relations. (ii) For all the single relational graph embedding methods, namely LINE, DeepWalk, Node2Vec, Planetoid and GCN, we find that their results on union graph structures are better than those on single relational graphs in general. The observation implies the usefulness of multi-relational topology, which validates our motivation. (iii) GCN and Planetoid always beat LINE, DeepWalk and Node2Vec, because they are semi-supervised approaches while the other methods are unsupervised. (iv) We observe

Type	Method							
	LINE	DeepWalk	Node2Vec	Planetoid	GCN	HAN	GTN	MR-GCN
rel. 1	26.5	27.3	29.3	58.9	64.8	64.8	63.4	**68.8** (Cosine)
rel. 2	27.9	29.1	27.7	58.8	65.4			**68.8** (Hadamard)
rel. 3	25.1	26.7	27.2	57.4	64.8			67.8 (Fourier)
rel. 4	23.8	26.7	27.0	57.7	64.8			68.6 (Hartely)
union	26.9	28.2	28.7	59.3	66.2			**68.8** (Haar)
multi	-	-	-	-	-			68.6 (Slant)

Table 3: Classification accuracies on Reuters.

that GNN based approaches, namely GCN (union), HAN, GTN and the proposed MR-GCN significantly outperforms Planetoid (union), except for the Amazon dataset where HAN is worse due to no-well-fixed meta-paths available. The fact implies the necessity to develop graph convolution operators to tackle graph related problems, which also validates our motivation to build convolution operators for multi-relational graphs.

In Table 3, similar observations and conclusions can be made. Moreover, among the six transforms, we find that Cosine, Hadamard and Haar deliver the best results, followed by Hartely and Slant. Fourier is less competitive than the other five transforms. The reason may be that the circular correlation (corresponding to Fourier), which makes periodic assumptions on signals, is not very suitable to model the graph relations.

Ablation Studies

Revisiting Figure 3, it can be seen that our framework contains two key components: (i) constructing input tensor with a fold of the GCN result on each single relation; (ii) multi-relational graph convolution operator. To examine the effectiveness of the two components, we conduct experiments on two variants, namely '*-w/o* GCN' which utilizes random initializations instead of GCNs, and '*-w/o* MR-GCO' which removes the multi-relational graph convolution operators. Table 4 shows the results, from which we can conclude that both components are very useful in our MR-GCN.

4.3 Parameter Sensitivity Study

Impact of layer numbers. Here we investigate how the number of MR-GCO layers affects the performance. Figure 4 shows the results on the four datasets, where the layer number is increased from 1 to 5. It can be observed that the parameter is more sensitive on IMDB and Reuters than on ACM and Amazon. The reason may be that the distributions of node representations learned by different MR-GCO layers sensitively affect the margins of classifiers on the two datasets.

Convergence and impact of training set sizes. In this part, we test the convergence and the impact of training set sizes. Here we adopt the Cosine transform and show the results on Reuters (Results on other datasets are similar). We can see

Method (Cosine)	Datasets			
	ACM	IMDB	Amazon	Reuters
MR-GCN	93.22	61.65	77.80	68.60
- *w/o* GCN	89.27	53.23	77.10	66.80
- *w/o* MR-GCO	91.81	59.94	76.50	66.80

Table 4: Ablation test.

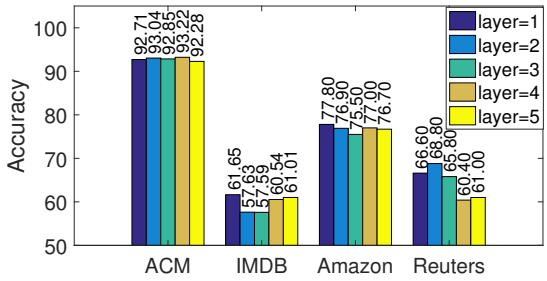
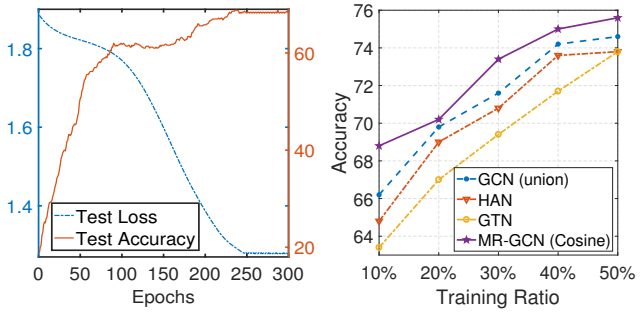


Figure 4: Impact of layer numbers of MR-GCO.



(a) Convergence. (b) Training set sizes impact.

Figure 5: Convergence and impact of training set sizes.

from Figure 5(a) that the test loss and accuracy tend to level off after 250 epochs, indicating a fast convergence. From Figure 5(b), we find that the performance of GCN (union), HAN, GTN and the proposed MR-GCN all improve as the training sample ratio is increased from 10% to 50%, which is not surprising as the methods obtain more supervision information. Moreover, we observe that MR-GCN consistently outperforms GCN (union), HAN and GTN, which again validates the superiority of the proposed method.

5 Conclusion

In this paper, we study the multi-relational graph learning problem and extend the notion of GCN by developing a novel multi-relational graph convolution operator. The operator is defined upon the eigenvalue decomposition of the Laplacian tensor. As we consider generalized tensor product, the decomposition can correspond to any unitary transform. Extensive experiments on four real-world datasets have been conducted, and the results show the effectiveness of the proposed method.

Acknowledgments

This research was supported in part by the National Key R&D Program of China, 2018YFB2101100, 2018YFB2101101 and NSFC under Grant Nos. 61972111, U1836107, 61602132 and 61572158.

References

[Bruna *et al.*, 2013] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[Defferrard *et al.*, 2016] 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.

[Grover and Leskovec, 2016] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 855–864, 2016.

[Hamilton *et al.*, 2017] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

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

[Jain, 1979] Anil K Jain. A sinusoidal family of unitary transforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (4):356–365, 1979.

[Jørgensen *et al.*, 2018] Peter Bjørn Jørgensen, Karsten Wedel Jacobsen, and Mikkel N Schmidt. Neural message passing with edge updates for predicting properties of molecules and materials. *arXiv preprint arXiv:1806.03146*, 2018.

[Khan and Blumenstock, 2019] Muhammad Raza Khan and Joshua E. Blumenstock. Multi-gcn: Graph convolutional networks for multi-view networks, with applications to global poverty. In *Thirty-Third AAAI Conference on Artificial Intelligence*, pages 606–613, 2019.

[Kilmer and Martin, 2011] Misha E Kilmer and Carla D Martin. Factorization strategies for third-order tensors. *Linear Algebra and its Applications*, 435(3):641–658, 2011.

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

[Li *et al.*, 2013] Xutao Li, Michael K Ng, and Yunming Ye. Multicomm: Finding community structure in multi-dimensional networks. *IEEE Transactions on Knowledge and Data Engineering*, 26(4):929–941, 2013.

[Li *et al.*, 2017] Xutao Li, Yunming Ye, and Xiaofei Xu. Low-rank tensor completion with total variation for visual data inpainting. In *Thirty-First AAAI Conference on Artificial Intelligence*, pages 2210–2216, 2017.

[Ma *et al.*, 2019] Yao Ma, Suhang Wang, Chara C Aggarwal, Dawei Yin, and Jiliang Tang. Multi-dimensional graph convolutional networks. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 657–665, 2019.

[Monti *et al.*, 2017] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124, 2017.

[Ng *et al.*, 2011] Michaek Kwok-Po Ng, Xutao Li, and Yunming Ye. Multirank: co-ranking for objects and relations in multi-relational data. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1217–1225, 2011.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 701–710, 2014.

[Schlichtkrull *et al.*, 2018] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607, 2018.

[Song *et al.*, 2019] Guangjing Song, Michael K Ng, and Xiongjun Zhang. Robust tensor completion using transformed tensor svd. *arXiv preprint arXiv:1907.01113*, 2019.

[Sun *et al.*, 2019] Yiwei Sun, Suhang Wang, Tsung-Yu Hsieh, Xianfeng Tang, and Vasant Honavar. Megan: A generative adversarial network for multi-view network embedding. In *Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 3527–3533, 2019.

[Tang *et al.*, 2015] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *The World Wide Web Conference*, pages 1067–1077, 2015.

[Wang *et al.*, 2019] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The World Wide Web Conference*, pages 2022–2032, 2019.

[Yang *et al.*, 2016] Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *International Conference on Machine Learning*, pages 40–48, 2016.

[Ying *et al.*, 2018] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 974–983, 2018.

[Yun *et al.*, 2019] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. In *Advances in Neural Information Processing Systems*, pages 11960–11970, 2019.