ON GRAPH CONVOLUTION FOR GRAPH CNNS

Jian Du, John Shi, Soummya Kar, and José M. F. Moura

Department of Electrical and Computer Engineering
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA 15213, USA.

ABSTRACT

The graph convolutional layer is core in the architecture of graph convolutional neural networks (CNNs). In the literature, both spectrum domain based and vertex domain based graph convolutional layers have been proposed. This paper analyzes these two types of graph convolutional layers and demonstrates that the spectrum domain based graph convolutional layer suffers from output inconsistencies when the graph shift matrix has repeated eigenvalues. In contrast, the vertex domain based graph convolutional layer is output consistent and inherits the local feature extraction property of classical CNNs with low computational complexity. Experimental results on different data sets also demonstrate that vertex domain based graph CNNs exhibit better performance than existing methods for classification problems.

***Index Terms*—** Graph CNN, graph signal processing, graph Fourier transform

1. INTRODUCTION

Convolutional neural networks (CNNs) exhibit state-of-the-art performance on a variety of learning tasks dealing with grid-structured data such as acoustic signals, images, and videos. These symbols and images are characterized by regular graph structures. However, traditional CNNs can not adapt to the irregular graph structures in other domains such as in knowledge graphs, social networks, citation networks, and traffic networks since the convolutional layer is not adaptive to the graph topology.

Recently, there has been an increasing interest in graph CNNs [1, 2, 3, 4] attempting to generalize CNNs to data defined on irregular graph domains by adapting the convolutional layer to the graph. In [1], the convolutional layer in the CNN architecture was generalized to data defined on irregular graphs. Convolution is achieved by a pointwise product of the graph-data spectrum and the graph filter spectrum using a set of B-spline bases. Later, [2] proposed a spectrum filter, which replaces the graph filter spectrum in [1] with a non-parametric graph spectrum. Reference [2] also utilizes Chebyshev polynomials to approximate graph convolution. In [3], the Chebyshev polynomial method is simplified further to reduce computational complexity. In contrast to the spectrum based methods mentioned above, [5] proposed topology adaptive graph convoltuional networks (TAGCNs) with the graph convolutional layer defined on the vertex domain.

In this paper, we use vertex domain based graph convolution as introduced in [6] in the convolutional layers of CNNs. Graph signal processing (GSP) [6, 7, 8, 9, 10] extends classical signal processing from 1-D signals to signals defined on arbitrary graphs. GSP considers graph domain spectrum analysis and the graph Fourier transform (GFT) based on either the graph adjacency matrix [6] or the Laplacian matrix [8] leading to different definitions of graph convolution. These GFT definitions are the basis of graph filter design, which has been studied extensively in recent years.

In this paper, we analyze the graph convolutional layer for graph CNNs. We show that, for spectrum based graph CNNs, the coordinatization of the graph spectrum, which is the matrix representing the graph spectrum bases, is not unique when there are repeated eigenvalues. Therefore, the output of the graph convolutional layer is not unique resulting in inconsistencies in the graph CNN outputs. In contrast, the TAGCN [5], with graph convolutional layers defined on the vertex domain, preserves the uniqueness of the graph CNN outputs. Further, TAGCN [5], like with traditional CNNs, still extracts local features with low learning complexity. Experimental results show that TAGCNs [5] exhibit better performance than existing graph CNNs on a number of data sets.

2. GRAPH CNN ARCHITECTURE

CNNs have a sequence of layers including convolutional layers, pooling layers, and a fully connected layer. Every layer transforms one volume of activations into another through a nonlinear activation function. Graph CNNs [1, 2, 3, 5] follow the same architecture, which includes graph convolutional layers, pooling layers, and a fully connected layer.[1] The main difference from CNNs is the replacement of convolu-

The work is partially supported by NSF grant CCF-1513936.

[1]For the problem of graph vertex classification, pooling layers are not used to avoid graph vertex information loss.

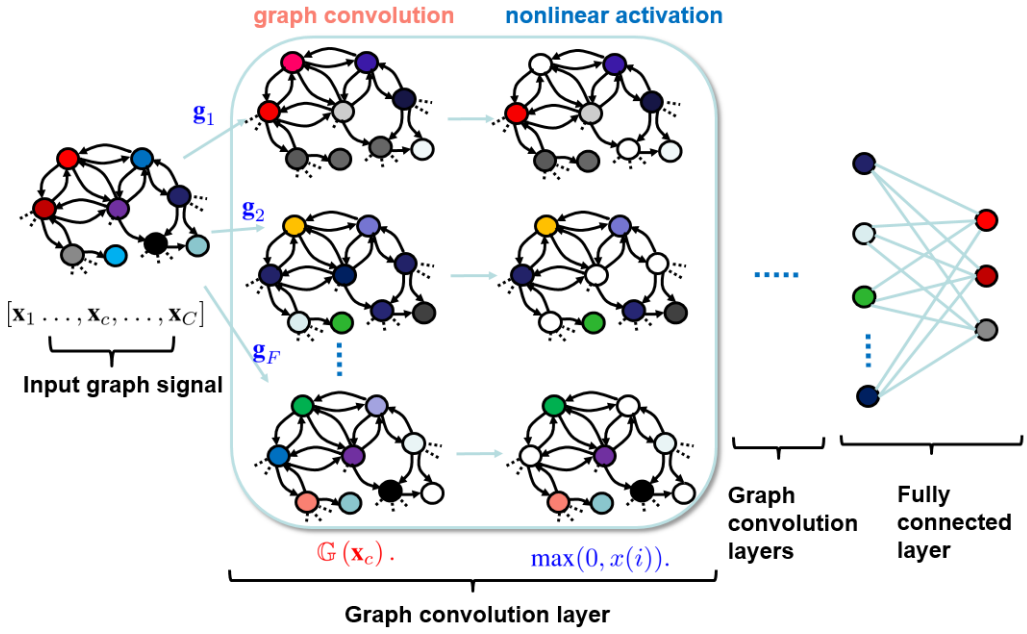

Fig. 1. The graph CNN Architecture.

tional layers by graph convolutional layers. Fig. 1 shows the general architecture of a graph CNN.

In recent graph CNN work [1, 2, 3], the convolution operation is defined in the signal spectrum domain by a pointwise product of the graph signal spectrum and the graph filter spectrum. Then, the result is transformed back to the graph vertex domain. These works use the graph Laplacian $\mathbf{L}$ eigendecomposition $\mathbf{L} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$. The graph convolution operation $\mathbb{G}(\cdot)$ (see Fig. 1) for a graph signal $\mathbf{x}$ is defined in [1, 2] as

$$\mathbf{y} = \mathbf{V} g_\theta(\mathbf{\Lambda}) \mathbf{V}^{-1}\mathbf{x}. \tag{1}$$

In the original spectral formulation of graph CNN [1], the filter $g_{\boldsymbol{\theta}}(\mathbf{\Lambda})$ is defined as

$$g_{\boldsymbol{\theta}}(\mathbf{\Lambda}) = \mathbf{B}\boldsymbol{\theta}, \tag{2}$$

where $\mathbf{B}$ is the cubic B-spline basis, and $\boldsymbol{\theta}$ is the vector of control points. Later, in [2], $g_{\boldsymbol{\theta}}(\mathbf{\Lambda})$ is defined as a diagonal non-parametric matrix with respect to $\mathbf{\Lambda}$ with

$$g_{\boldsymbol{\theta}}(\mathbf{\Lambda}) = \mathrm{diag}\{\boldsymbol{\theta}\}, \tag{3}$$

where $\boldsymbol{\theta}$ is the graph filter coefficients.

In the next section, we demonstrate that in all of the above mentioned spectrum based graph CNNs, the output of the convolutional layer lacks consistency since the basis of the graph Fourier transform $\mathbf{V}$ (or equivalently $\mathbf{V}^{-1}$) is not unique when $\mathbf{L}$ has repeated eigenvalues.

3. INCONSISTENCY OF SPECTRUM BASED GRAPH CNN

Let graph Laplacian $\mathbf{L} \in \mathbb{R}^{N\times N}$ have k distinct eigenvalues $\lambda_1, \ldots, \lambda_k$, with $k \leq N$. Eigenvalues of $\mathbf{L}$ are roots of the characteristic polynomial $\phi_{\mathbf{L}} = \det(\mathbf{L} - \lambda\mathbf{I}) = \prod_{i=1}^{k}(\lambda - \lambda_i)^{a_i}$, where $\mathbf{I}$ is the identity matrix, and a_i is the algebraic multiplicity of λ_i with $i = 1, 2, \ldots, k$. The graph Laplacian $\mathbf{L}$ is diagonalizable, it can be factorized by the eigendecomposition with $\mathbf{V} = [\mathbf{v}_1 \ldots \mathbf{v}_n]$ and $\mathbf{\Lambda} = \mathrm{diag}\{\lambda_1, \lambda_2, \ldots \lambda_N\}$. There are a_i eigenvectors corresponding to λ_i. These a_i eigenvectors can be chosen orthonormal to each other and form a basis of the eigenspace for λ_i. However, the orthogonal basis for the eigenspace corresponding to λ_i is not unique when $a_i > 1$. We can rotate any orthogonal basis to obtain another orthogonal basis for the same eigenspace. Therefore, neither $\mathbf{V}$ nor the output of the corresponding graph convolutional layer are unique.

An example of the inconsistency with graph convolutional layer outputs is illustrated in the following. Let $\mathbf{L}$ be the Laplacian matrix of the graph in Fig 2. Let the graph signal $\mathbf{x} = [1, 2, 3, 1, 2, 3, 1, 2]^T$ with graph signal indices following the graph vertex indices in Fig. 2.

The distinct eigenvalues of 8×8 graph Laplacian $\mathbf{L}$ corresponding to the graph in Fig. 2 are $\lambda_1 = 0$, $\lambda_2 = 2$, $\lambda_3 = 4$, and $\lambda_4 = 6$. The eigenvalues $\lambda_1 = 0$ and $\lambda_4 = 6$ both have algebraic multiplicity 1, and the eigenvalues $\lambda_2 = 2$ and $\lambda_3 = 4$ both have algebraic multiplicity 3. Table 1 shows a set of corresponding eigenvectors for each eigenvalue accurate up to two decimal places. Table 2 shows another orthogonal basis

Table 1. Eigenvalues and eigenvectors of $\mathbf{L}$ corresponding to the undirected graph in Fig. 2.

Eigenvalue	Eigenvector
$\lambda_1 = 0$	$\mathbf{v}_1 = [.35, .35, .35, .35, .35, .35, .35, .35]^T$
$\lambda_2 = 2$	$\mathbf{v}_2 = [-.05, .42, -.08, -.56, -.42, .05, .56, .08]^T$ $\mathbf{v}_3 = [.53, .39, .10, .23, -.39, -.53, -.23, -.10]^T$ $\mathbf{v}_4 = [-.30, 0.20, .60, .09, -.20, .30, -.09, -.60]^T$
$\lambda_3 = 4$	$\mathbf{v}_5 = [-.13, -.14, .60, -.32, -.14, -.13, -.32, .60]^T$ $\mathbf{v}_6 = [-.26, .59, -.06, -.27, .59, -.26, -.27, -.06]^T$ $\mathbf{v}_7 = [-.53, -.02, .12, .44, -.02, -.54, .44, .12]^T$
$\lambda_4 = 6$	$\mathbf{v}_8 = [.35, -.35, .35, -.35, .35, -.35, .35, -.35]^T$

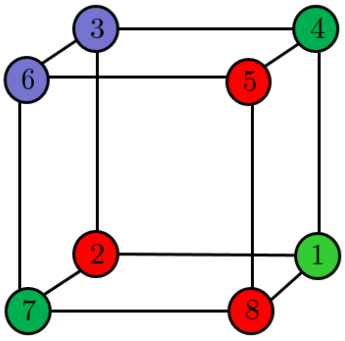

Fig. 2. Topology of an undirected graph. Different colors represent different graph signal values on the vertices, i.e., nodes with indices $1, 4, 7$ have the same value 1.0, nodes with indices $2, 58$ have the same value 2.0, and nodes with indices $3, 6$ have the same value 3.0.

obtained as follows. First, let

$$\widetilde{\mathbf{v}}_5 = \frac{\mathbf{v}_5 + \mathbf{v}_6 + \mathbf{v}_7}{\|\mathbf{v}_5 + \mathbf{v}_6 + \mathbf{v}_7\|}.$$

Then by applying Gram-Schmidt orthogonalization to $\widetilde{\mathbf{v}}_5$, $\mathbf{v}_6$, and $\mathbf{v}_7$, we obtain a new set of orthogonal basis $\{\widetilde{\mathbf{v}}_5, \widetilde{\mathbf{v}}_6, \widetilde{\mathbf{v}}_7\}$ for the eigenspace of $\lambda_3 = 4$. Let the graph convolutional coefficients in (3) be $\boldsymbol{\theta} = [1, 2, 3, 4, 5, 6, 7, 8]^T$. The outputs $\mathbf{y}$ (with $g_{\boldsymbol{\theta}}$ in (3)) computed using the two different $\mathbf{V}$'s in Table 1 and Table 2, are $[-0.79, 3.67, 6.39, -2.55, 1.97, 7.19, -4.06, 3.19]^T$ and $[-0.89, 3.73, 6.90, -3.02, 2.02, 7.09, -4.53, 3.70]^T$, respectively. This example shows that the output of the graph convolutional layer is inconsistent if different graph eigenbases are adopted.

4. VERTEX DOMAIN BASED GRAPH CNN

In contrast to the spectrum based graph CNN, the vertex based TAGCN [5] keeps the uniqueness of each convolutional layer. Without loss of generality, we demonstrate graph convolution on the ℓ-th graph convolutional layer. Suppose the input feature map for each vertex of the graph in this layer has C_ℓ features. We collect the ℓ-th layer input data on all vertices for the c-th feature by the vector $\mathbf{x}_c^{(\ell)} \in \mathbb{R}^{N_\ell}$, where $c = 1, 2, \ldots C_\ell$ and N_ℓ is the number of vertices. The components of $\mathbf{x}_c^{(\ell)}$ are indexed by vertices of the graph representation $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \bar{\mathbf{A}})$[2]. Let $\mathbf{G}_{c,f}^{(\ell)} \in \mathbb{R}^{N_\ell \times N_\ell}$ denote the f-th graph filter. The graph convolution is the matrix-vector product $\mathbf{G}_{c,f}^{(\ell)} \mathbf{x}_c^{(\ell)}$. Then, the f-th output feature map is

$$\mathbf{y}_f^{(\ell)} = \sum_{c=1}^{C_\ell} \mathbf{G}_{c,f}^{(\ell)} \mathbf{x}_c^{(\ell)} + b_f^{(\ell)} \mathbf{1}_{N_\ell}, \tag{4}$$

where $b_f^{(\ell)}$ is a learnable bias, $\mathbf{1}_{N_\ell}$ is the N_ℓ dimension vector of all ones, and [5, 6]

$$\mathbf{G}_{c,f}^{(\ell)} = \sum_{k=0}^{K} g_{c,f,k}^{(\ell)} \mathbf{A}^k. \tag{5}$$

In (5), the $g_{c,f,k}^{(\ell)}$ are the graph filter polynomial coefficients; the quantity $\mathbf{A} = \mathbf{D}^{-\frac{1}{2}} \bar{\mathbf{A}} \mathbf{D}^{-\frac{1}{2}}$ is the normalized adjacency matrix of the graph, and $\mathbf{D} = \text{diag}[\mathbf{d}]$ with the ith component $\mathbf{d}(i) = \sum_j \mathbf{A}_{i,j}$.[3] We adopt the normalized adjacency matrix to guarantee that all the eigenvalues of $\mathbf{A}$ are inside the unit circle, and therefore, $\mathbf{G}_{c,f}^{(\ell)}$ is computationally stable [11].

It is evident that the output $\mathbf{y}_f^{(\ell)}$ in (4) is unique given the same input $\mathbf{x}_c^{(\ell)}$. Furthermore, the filter represented by K-th order polynomials of the adjacency matrix is K-localized with learning complexity $\mathcal{O}(K)$. Thus, it has the same complexity as classical CNN.

Further, the spectrum based method in (1) does not scale up due to the eigendecomposition is required with a computational complexigty of $\mathcal{O}(N^3)$. Moreover, both forward and inverse Fourier transforms require a computional cost of $\mathcal{O}(N^2)$. In contrast, the above vertex domain based TAGCN avoids the computation burden of eigendecomposition and Fourier transforms.

[2]We use superscript (ℓ) to denote data on the ℓth layer and superscript ℓ to denote the ℓ-th power of a matrix.

[3]There is freedom to normalize $\mathbf{A}$ in different ways; here it is assumed that $\bar{\mathbf{A}}_{m,n}$ is nonnegative and the above normalization is well defined.

Table 2. A rotation of the eigenvectors in Table. 1.

Eigenvalue	Eigenvector
$\lambda_1 = 0$	$\mathbf{v}_1 = [.35, .35, .35, .35, .35, .35, .35, .35]^T$
λ_2 = 2	$\mathbf{v}_2 = [-.05, .42, -.08, -.56, -.42, .05, .56, .08]^T$ $\mathbf{v}_3 = [.53, .39, .10, .23, -.39, -.53, -.23, -.10]^T$ $\mathbf{v}_4 = [-.30, 0.20, .60, .09, -.20, .30, -.09, -.60]^T$
$\lambda_3 = 4$	$\widetilde{\mathbf{v}}_5 = [-.30, .20, .60, .09, -.20, .30, -.09, -.60]^T$ $\widetilde{\mathbf{v}}_6 = [-.54, .25, .38, -.09, .25, -.54, -.09, .38]^T$ $\widetilde{\mathbf{v}}_7 = [.22, -.35, .47, -.33, -.35, .22, -.33, .47]^T$
$\lambda_4 = 6$	$\mathbf{v}_8 = [.35, -.35, .35, -.35, .35, -.35, .35, -.35]^T$

Table 3. Dataset statistics

Dataset	**Nodes**	**Edges**	**Classes**	**Features**	**Label rate**
Citeseer	3,327	4,732	6	3,703	0.036
Cora	2,708	5,429	7	1,433	0.052
Pubmed	19,717	44,338	3	500	0.003

5. EXPERIMENTS

In this section, TAGCN is evaluated on three citation network data sets: Citeseer, Cora, and Pubmed. We closely follow the data set split and experimental setup in [12]. Each data set consists of certain classes of documents with only a few labeled documents. The task is to classify the documents in the test set with these limited number of labels. In each data set, the vertices are the documents and the edges are the citation links between them. Each document is represented by sparse bag-of-words feature vectors, and the citation links between documents are provided. Detailed statistics of these three data sets are summarized in Table 3.

Table 4. Summary of results in terms of percentage classification accuracy

	Citeseer	**Cora**	**Pubmed**
Planetoid[12]	64.7	75.7	77.2
ChebNet (K=2)[2]	69.6	81.2	73.8
ChebNet (K=3)[2]	69.8	79.5	74.4
GCN[3]	70.3	81.5	79.0
Ours TAGCN	**71.4**	**83.3**	**81.1**

We use a graph CNN with two hidden layers for the semi-supervised node classification. In each hidden layer, the proposed (4) is applied for convolution, followed by a ReLU activation. In total, 16 hidden units (filters) are designed for each hidden layer, and dropout is applied after each hidden layer. The softmax activation function is applied to the output of the second hidden layer for the final classification.

To make a fair comparison, we closely follow the same split of training, validation, and testing sets as in [3, 12], i.e., 500 labeled examples for hyperparameters (filter size, dropout rate, and number of hidden layers) optimization and cross-entropy error is used for classification accuracy evaluation. The performance results of the proposed TAGCN method are averaged over 100 runs with random initializations [13].

We compare the classification accuracy with other recently proposed graph CNN methods as well as a graph embedding method known as Planetoid [12]. The quantitative results are summarized in Table 4. Reported numbers denote classification accuracy in percent. Results for Planetoid, GCN, and ChebNet are taken from [3]. All the experiments for different methods are based on the same data statistics shown in Table 3. The datasets split and experiment settings closely follow the standard criteria in [12, 3]. Table 4 shows that our method outperforms all the recent state-of-the-art methods by significant margins for all the three datasets. These results verify the efficacy of the proposed TAGCN.

6. CONCLUSION

In this paper, we have analyzed the inconsistencies of spectrum domain based graph convolutional neural networks (CNNs). In contrast, the vertex domain based graph CNN demonstrates a good application potential due to its output uniqueness and low computational complexity. Experiments also show good classification accuracy for vertex domain based graph CNNs in semi-supervised learning.

7. REFERENCES

[1] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun, "Spectral networks and locally connected networks on graphs," in *Internatonal Conference on Learning Representations (ICLR)*. 2014.

[2] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst, "Convolutional neural networks on graphs

with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, 2016.

[3] Thomas N. Kipf and Max Welling, "Semi-supervised classification with graph convolutional networks," in *Internatonal Conference on Learning Representations (ICLR)*. 2017.

[4] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodolà, 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 (CVPR)*. 2017.

[5] Jian Du, Shanghang Zhang, Guanhang Wu, José M-F Moura, and Soummya Kar, "Topology adaptive graph convolutional networks," *arXiv preprint arXiv:1710.10370*, 2017.

[6] Aliaksei Sandryhaila and José M. F. Moura, "Discrete signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1644–1656, 2013.

[7] Aliaksei Sandryhaila and José M. F. Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 80–90, 2014.

[8] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[9] Aliaksei Sandryhaila and José M. F. Moura, "Discrete signal processing on graphs: Frequency analysis.," *IEEE Trans. Signal Processing*, vol. 62, no. 12, pp. 3042–3054, 2014.

[10] Joya. A. Deri and José M. F. Moura, "Spectral projector-based graph Fourier transforms," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 785–795, Sept 2017.

[11] Jian Du, Shaodan Ma, Yik-Chung Wu, Soummya Kar, and José M. F. Moura, "Convergence analysis of distributed inference with vector-valued Gaussian belief propagation," *Journal of Machine Learning Research*, 2018.

[12] Zhilin Yang, William Cohen, and Ruslan Salakhutdinov, "Revisiting semi-supervised learning with graph embeddings," in *Internatonal Conference on Learning Representations (ICLR)*. 2016.

[13] Xavier Glorot and Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 249–256. 2010.