Tensor based Graph Convolutional Networks for High-dimensional and Low-sample Size Data

Anonymous Author(s)

ABSTRACT

Semi-supervised classification is a valuable approach for predicting labels for all data with limited labeled samples, particularly in tackling high-dimensional and low-sample size (HDLSS) data. In recent years, graph convolutional networks(GCNs) have achieved significant success in the field of semi-supervised classification. However, existing GCN-based methods suffer from several major limitations. Firstly, typical GCN relies on pairwise similarity, which is not well-compatible for real-world applications and its robustness is challenged by the collapse of sample distances and noise interference under HDLSS setting. Secondly, the low-pass filtering property of GCN tends to ignore valuable higher-order signals, which becomes particularly detrimental in high-dimensional data. On the other hand, employing complex polynomial filters is ineffective in capturing higher-order neighborhood information while posing the risk of overfitting. In order to address these issues, we propose a tensor-based graph convolutional network (Tensor-GCN) that directly model higher-order relationships and incorporate them into the graph convolution process. Additionally, we adopt an innovative tensor similarity measure that describes the geometric relationship between three samples, providing complementary information to compensate for the missing pairwise similarities. Finally, we design a multi-layer GCN framework that seamlessly integrates both traditional low-order and high-order neighborhood information to achieve more accurate and robust predictions. Extensive experiments on benchmark HDLSS data indicate that Tensor-GCN can exploit higher-order feature information and provide the best predictive performance and robustness for HDLSS data.

CCS CONCEPTS

• **Do Not Use This Code → Generate the Correct Terms for Your Paper**; *Generate the Correct Terms for Your Paper*; Generate the Correct Terms for Your Paper; Generate the Correct Terms for Your Paper.

KEYWORDS

Graph convolutional networks, network representation learning, high-order similarity

ACM Reference Format:

Anonymous Author(s). 2024. Tensor based Graph Convolutional Networks for High-dimensional and Low-sample Size Data. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Semi-supervised classification aims to learn representation from limited labeled samples and categorize all data accordingly, it's valuable in situations where acquiring labeled data is challenging, particularly in addressing the high-dimension m yet low-sample size n (HDLSS) data when $n \gg m$ [22, 28, 34]. In this context, Graph Convolutional Networks (GCNs) have shown significant promise in semi-supervised classification and capable of tackling versatile data, such as graph [20], image [8], text [2, 36] and bioinformatics data [6, 32].

Traditional GCN primarily follows an information propagation approach to aggregate feature representation from neighboring nodes. Its essence lies in the Laplacian smoothing [33] of node features [23], ultimately facilitating the propagation of node features within the network structure and forming graph embeddings. Therefore, GCN models can effectively learn from limited labeled samples and capture relationships between data, enabling more accurate predictions and insights. Remarkable success over the past few years is achieved utilizing GCN framework. However, even state-of-the-art GCN models struggle to cope with high-dimensional data in practical scenarios. This is partly due to the low-pass filtering characteristic of GCNs, which disregards useful high-order signals while denoising [9]. Additionally, the correlations in high-dimensional data can't be represented solely through pairwise relationships between nodes [11]. Due to the concentration effects [12] under HDLSS setting, where the pairwise distances between samples collapse to a constant, resulting in the difficulty to distinguish between them[14]. Therefore the aforementioned weaknesses of GCN become even more pronounced in HDLSS cases.

Technically, GCN models generally exhibit considerable scalability, one may believe that more complex GCN framework tend to possess stronger feature extraction capabilities. However, this may not be as straightforward as initially anticipated. For instance, the performance of ChebNet [10], which utilizes high-order Chebyshev polynomial expansions, has been found to be inferior to that of simpler GCN [20]. This discrepancy can be attributed to the fact that the illegal coefficients learned by ChebNet approximate analytic fitting functions, leading to overfitting [17]. Fundamentally, the issue lies in the loss of information by the polynomial convolution kernel during the process of Laplacian smoothing.

Urgently needed is a solution that effectively captures high-order relationships to enable the GCN model to capture high-order feature representations. In this paper, we propose Tensor-GCN, which adopts a novel perspective of understanding high-order correlation among samples using tensor similarity. Tensor-GCN enhances the output of low-pass filters and achieves comprehensive capture

of high-order information. Moreover, we meticulously integrates low-dimensional and high-dimensional information, employing a deep network based on multi-order similarity for exploiting latent embedding.

Our main contributions are summarized as follows:

- (No need...)We present Tensor based GCN that draws inspiration from the second-order Chebyshev polynomials [16]. Allowing for direct modeling of higher-order relationships among samples and seamlessly integrates high-order neighborhood message into the graph convolution process.
- A inventively tensor similarity is adopted in our spectral convolution process, which depicts geometric links among three samples and therefore provides complementary information that the pairwise similarity missed.
- We present a carefully-designed multilayer GCN framework that seamlessly integrates both conventional low-order and high-order neighborhood information to achieve more accurate and robust predictions.
- Comparative evaluations of Tensor-GCN against other state-of-the-art GCN methods on public HDLSS datasets demonstrate significant advantages in terms of classification accuracy and robustness, indicating that the proposed Tensor-GCN is capable of enhancing node representations and is well-suited for HDLSS data.

The remaining content will be organized in the following manner: Section 2 introduces relevant symbols, definitions, and previous work. In Section 3 we presents the construction and implementation of the proposed Tensor-GCN model. Section 4 showcases the experimental results and analysis on the HDLSS dataset. Finally, we summarizes the paper in Section 5.

2 PRELIMINARIES AND RELATED WORK

2.1 Notations

In this paper, we employ the use of bold calligraphy, uppercase letters, and lowercase letters to symbolize tensors, matrices, and vectors, respectively. For an order-3 tensor $\mathcal{T} \in \mathbb{R}^{I\times J\times K}$, $\mathcal{T}(:,:,i)$,$\mathcal{T}(:,i,:)$,$\mathcal{T}(i,:,:)$ represents the i-th frontal, lateral and horizontal slices of $\mathcal{T}$, respectively. $\mathcal{T}(:,:,i)$ can be abbreviated as $\mathcal{T}^{(i)}$. A tensor can be transformed into a matrix through a series of operations known as unfolding. For example, the three-order tensor unfold operations is as follows:

Definition 2.1. ***Unfolding 3-order Tensor*** **:** Let $\mathcal{T}_3 \in \mathbb{R}^{N\times N\times N}$ represent an order-3 n-dimensional tensor. This tensor can unfold to an $n^2 \times n$ matrix $\hat{T}_3$ as follows:

$$\hat{T}_3 = unfold(\mathcal{T}_3) = \begin{bmatrix} \mathcal{T}_3^{(1)} \\ \mathcal{T}_3^{(2)} \\ \vdots \\ \mathcal{T}_3^{(n)} \end{bmatrix} \tag{1}$$

Several matrix/tensor products are important in the sections that follow, namely, Hadmand product, Kronecker product, Khatri-Rao product and k-mode product [27], we briefly define them here.

Definition 2.2. ***Hadmand Product*** The Hadamard product of two matrices $A \in \mathbb{R}^{I\times J}$ and $B \in \mathbb{R}^{I\times J}$ is defined by:

$$A \odot B = \begin{bmatrix} a_{11}b_{11} & \cdots & a_{1J}b_{1J} \\ \vdots & \ddots & \vdots \\ a_{I1}b_{I1} & \cdots & a_{IJ}b_{IJ} \end{bmatrix} \in \mathbb{R}^{I\times J} \tag{2}$$

Definition 2.3. ***Kronecker Product*** The Kronecker product of matrices $A \in \mathbb{R}^{I\times J}$ and $B \in \mathbb{R}^{K\times L}$ is defined by:

$$\begin{aligned} A \otimes B &= \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1J}B \\ a_{21}B & a_{22}B & \cdots & a_{2J}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}B & a_{I2}B & \cdots & a_{IJ}B \end{bmatrix} \in \mathbb{R}^{IK\times JL} \\ &= \begin{bmatrix} a_1 \otimes b_1 & a_1 \otimes b_2 & \cdots & a_J \otimes b_{L-1} & a_J \otimes b_L \end{bmatrix}. \end{aligned} \tag{3}$$

Definition 2.4. ***Khatri-Rao Product*** The Khatri-Rao product [30] is the "matching columnwise" Kronecker product. Given matrices $A \in \mathbb{R}^{I\times K}$ and $B \in \mathbb{R}^{J\times K}$ is defined as the matrix:

$$A * B = \begin{bmatrix} a_1 \otimes b_1 & a_2 \otimes b_2 & \cdots & a_K \otimes b_K \end{bmatrix} \in \mathbb{R}^{IJ\times K}. \tag{4}$$

The Khatri–Rao and Kronecker products are identical if a and b are vectors, i.e., $a \otimes b = a * b$. Tensor multiplication is much more complex than matrix multiplication, here we consider only the tensor k-mode product [21], i.e., multiplying a tensor by a matrix (or a vector) in mode k.

Definition 2.5. k***-mode Product*** The k-mode product between an order-m tensor $\mathcal{T} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_m}$ and a matrix $V \in \mathbb{R}^{P\times I_k}$, denoted by $\mathcal{T} \otimes_k U \in \mathbb{R}^{I_1\times\cdots\times I_{k-1}\times P\times I_{k+1}\times\cdots\times I_m}$, with

$$(\mathcal{T} \otimes_k V)_{i_1\ldots i_{k-1}ji_{k+1}\ldots i_m} = \sum_{i_k=1}^{I_k} \mathcal{T}_{i_1\ldots i_{k-1}i_k i_{k+1}\ldots i_m} V_{ji_k}. \tag{5}$$

2.2 Experimental Endeavor on Exploring High-Order Graph Convolutional Networks/Related Work

Numerous approaches were proposed to extend convolutional networks to graphs, most of which fall into two broad categories: spatial approaches and non-spectral (spatial) approaches. For spatial approaches, the convolution operation is defined for groups of spatially neighboring nodes. DCNNs [3] utilizes the powers of a transition matrix to define the neighborhood of nodes. Monti et al. [25] employs local path operators in the form of Gaussian mixture models to generalize convolution in spatial domain. GAT [35] assigns different weights to neighboring nodes based on their importance, allowing the model to focus on more informative nodes during aggregation. GraphSAGE [15] leverages a sampling strategy to select representative neighboring nodes and aggregates their features to update node representations.

We primarily focuses on spectral graph convolutional network, initially introduced in Bruna et al. [4] which formulates the convolution operation in spectral domain of graph and later extended by Defferrard et al. [10] with a Chebyshev expansion of the graph Laplacian that approximate the spectral filters. In Kipf et al. [20], an efficient layer-wise propagation model is proposed by simplify the chebyshev polynomials to first-order. These methods have played a

significant role in advancing the field of GNN and have been widely used as foundational approaches for graph-based learning tasks.

Despite the remarkable achievements of GCNs, in real-world applications, data structures often transcend simple pairwise connections, evolving into highly complex and intricate networks, particularly when dealing with HDLSS data. Under such circumstance, traditional Graph Convolutional Networks face limitations in effectively capturing and formulating the intricate data correlations. Novel GCN methods have been recently proposed to tackle the issue of mining high-order information among data. MixHop [1] is proposed to learn difference operators by repeatly mixing feature representations of neighbors at various distances. BScNet [7] replaces the graph Laplacian with the block Hodge Laplacian to obtain high-order feature representations, but the Hodge theory is inherently limited to modeling interactions between simplices that differ by only one order.

Alternatively, Gao and Feng et al. [11, 13] attempt to utilize hypergraphs to encode high-order data correlation, which generated hyperedge with two or more samples thus introduced a new form of high-order similarity. Nervertheless, their framework is still based on the first-order expansion of Chebyshev polynomials proposed by Hammond et al. [16], resulted in hypergraph models relying on the high-order similarity by averaging all pairwise relationship within hyperedges, so the learned representation relies heavily on pairwise information. Moreover, it has been shown that a hypergraph could degenerate into a standard graph. GRACES [6] introduce feature selection when tackling HDLSS data. HiGCN [19] grounded in FP Laplacians, capable of discerning features across varying topological scales, but it is still unclear whether it can overcome the concentration effect and noise interference under the challenge of HDLSS conditions.

2.3 Introduction to high-order similarity

Most GCNs require the Laplacian matrix L and sample features X as inputs. To generate L, we first compute the pairwise distances of all sample pairs using a suitable distance metric, such as correlation coefficients, Euclidean distance, heat kernel, or cosine similarity. Subsequently, a kNN graph is constructed based on distance, where similarity matrix S reflects the connectivity and similarity among samples. Finally, let the diagonal degree matrix D, denoted as $D_{ii} = \sum_{j=1}^{n} S_{ij}$, the Laplacian matrix is then defined as $L = I - D^{-\frac{1}{2}} S D^{-\frac{1}{2}}$.

2.3.1 Introduction to decomposable third-order tensor similarity. Despite the aforementioned similarity measures are capable of extracting local and global features to some extent, the reduction of intricate interactions to pairwise simplifications inevitably leads to a loss of valuable message. Our framework simultaneously leverages pairwise similarity and higher-order similarity, integrating low-order and high-order neighborhood information to obtain embeddings. In this paper, we adopts a tensor similarity theory proposed by Cai et al [5, 26]. We give a brief introduction to this theory in the following section, the reader is referred to Cai et al. [5, 26] for an in-depth discussion of tensor affinity. To balance model complexity and effectiveness, we focus on capturing tensor similarity among three samples, denoted by a third-order tensor $\mathcal{T}_3 = [\mathcal{T}_{ijk}] \in \mathbb{R}^{n\times n\times n}$. An intuitive way is to utilize composite similarity based on paired similarity S such that each entry T_{ijk} could be given by:

$$\mathcal{T}_{3_{ijk}} = S_{ij} S_{kj} \tag{6}$$

By the definition in Eq. (1), the decomposable third-order similarity $\mathcal{T}_3$ defined in Eq. (6) can be unfolded into matrix $\hat{T}_3 \in \mathbb{R}^{n^2\times n}$. $\hat{T}_3$ can be further written by:

$$\begin{aligned}
\hat{T}_3 &= \begin{bmatrix} \mathcal{T}_{3_{111}} & \cdots & \mathcal{T}_{3_{1n1}} \\ \vdots & \ddots & \vdots \\ \mathcal{T}_{3_{n1n}} & \cdots & \mathcal{T}_{3_{nnn}} \end{bmatrix} \\
&= \begin{bmatrix} S_{11}S_{:1} & \cdots & S_{1n}S_{:n} \\ \vdots & \ddots & \vdots \\ S_{n1}S_{:1} & \cdots & S_{nn}S_{:n} \end{bmatrix} \\
&= \begin{bmatrix} S_{:1} \otimes S_{:1} & \cdots & S_{:n} \otimes S_{:n} \end{bmatrix} \\
&= S * S
\end{aligned} \tag{7}$$

Eq. (7) reveals that a decomposable tensor similarity can be obtained through the Khatri-Rao product of two similarity matrices. Let a matrix $\hat{L}_3 = \hat{D}_{3_1}^{-\frac{1}{2}} \hat{T}_3 \hat{D}_{3_2}^{-\frac{1}{2}}$ with diagonal matrices $\hat{D}_{3_1}, \hat{D}_{3_2}$ given by $\hat{D}_{3_1} = \sqrt{\sum_i \hat{T}_{3:i} \sum_i \hat{T}_{3:i}}$ and $\hat{D}_{3_2} = \sum_i \hat{T}_{3_{:i}}$. We can obtain the normalized third-order similarity tensor $\mathcal{L}_3$ via folding $\hat{L}_3$. Furthermore, assuming $\hat{L}$ being the normalized Laplacian matrix constructed based on the KNN graph generated from S, it has been demonstrated by Cai et [26] that:

$$\hat{L}_3 = \hat{L} * \hat{L}. \tag{8}$$

Eq. (7) and Eq. (8) bridge the third-order similarity $\mathcal{T}_3$ and pairwise similarity S together via the Khatri-Rao product. Indeed, this also indicates that the structural feature extracted from decomposable high-order similarity is determined by the Khatri-Rao product of pairwise similarity, which holds true for pairwise Laplacian matrix and its eigenvectors. Therefore the decomposable high-order similarity suffers from the same drawbacks as the pairwise similarity similarity does, necessitating the design of a indecomposable tensor similarity to provide complementary information for which the pairwise may miss.

2.3.2 Introduction to indecomposable third-order tensor similarity. Inspired by the spatial relationships among samples, Cai et al. [5] introduced the Unified Tensor Clustering (UTC) and defined an indecomposable third-order similarity to mining the relationships among three samples from different perspectives. The indecomposable similarity $\mathcal{T} \in \mathbb{R}^{n\times n\times n}$ is defined as follows:

$$\mathcal{T}_{3_{ijk}} = 1 - \frac{< (x_i - x_j), (x_k - x_j) >}{d_{ij} d_{jk}} \tag{9}$$

for $i, j, k \in n$, where d_{ij} denotes the distance between node x_i and node x_j, i.e., Euclidean distance. The spirit lies behind this high-order similarity is as such: considering arbitrary three samples, x_i, x_j, and x_k, we treat x_j as the anchor point between x_i and x_k, then observe the relationship between the other two points from the perspective of x_j. If x_i and x_k are sufficiently similar, regardless of the position of x_j, there will be a significant similarity between x_i and x_k. Conversely, for outliers, their similarity to other data should be small when observed from most anchor points. Similar to the decomposable similarity, the entry $\mathcal{T}_{3_{ijk}}$ can be unfolded

into $\hat{T}_3 \in \mathbb{R}^{n^2 \times n}$ and then formulate $\mathcal{L}_3$, which when unfolded along the mode-3 direction satisfies $\hat{L}_3 = \hat{D}_{3_1}^{-\frac{1}{2}} \hat{T}_3 \hat{D}_{3_2}^{-\frac{1}{2}}$, where $\hat{D}_{3_1} = \sqrt{\sum_i \hat{T}_{3:i} \sum_i \hat{T}_{3:i}}$ and $\hat{D}_{3_2} = \sum_i \hat{T}_{3:i}$. The experimental validation on both synthetic and real-world datasets [5, 26] demonstrated that indecomposable similarity can complement pairwise similarity with three-dimensional spatial information, thereby enhancing the expressive power and robustness of the model.

3 TENSOR-GCN: THE PROPOSED MODEL

In this section, we introduce our proposed Tensor based Graph Convolutional Network (Tensor-GCN). We first review the construction and optimization of conventional spectral graph convolutions. Then, we present the tensor based graph convolution architecture. Lastly, we provide implementation details of the Tensor-GCN model.

3.1 Revisit spectral graph convolution

The Fourier bases $U \in \mathbb{R}^{n \times n}$ is obtained from the eigen-decomposition of the Laplacian, i.e., $L = U \Lambda U^T$, with $\Lambda = diag(\lambda_1, \lambda_2, \cdots, \lambda_n)$ corresponding to the eigenvalue of L in the increasing order. Fourier bases allows for the transfer of a graph signals $x \in \mathbb{R}^n$ to the spectral domain, defined as $U^T x$ [29]. The Fourier transform enables spectral graph convolution to be written as the multiplication between the signal x and the filter g, i.e.:

$$g \star x = U((U^T g) \odot (U^T x)) = U(g(\Lambda) U^T x). \tag{10}$$

Where $g(\Lambda)$ is the counterpart of filter g in the Fourier domain, which is often viewed as a function of the Laplacian eigenvalues [20], i.e., $g(\Lambda) = diag(g(\lambda_1), g(\lambda_2), \cdots, g(\lambda_n))$. It it noted that this definition of graph convolution with n independent parameters can be computationally complex. In practice, many methods [16, 18, 29] choose to approximate graph filters using polynomial filters, which can be represented by selecting a fixed set of filter coefficients, thereby simplifying the computation process. For example, Hammond et al. [16] suggested that K^{th}-order Chebyshev polynomials can be leveraged to approximate $g(\Lambda)$ effectively as $g(\Lambda) \approx \sum_{k=0}^{K} \beta_k T_k(\hat{\Lambda})$, where $\hat{\Lambda}$ is re-scaled from Λ, β_k is presented as Chebyshev coefficients. $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ is defined recurisively as $T_0(x) = 1$, $T_1(x) = x$. Based on the aforementioned approximation, graph convolution in the spectral domain is simplified to the following form:

$$g \star x \approx \sum_{k=0}^{K} \beta_k T_k(\hat{L}) x, \tag{11}$$

where $\hat{L}$ undergoes the same rescaling process as $\hat{\Lambda}$. The application of Chebyshev polynomials reduces the number of free parameters in graph convolution to K. Here, the expression of spectral graph convolution is K-localized since it is only influenced by nodes within the K^{th}-order neighborhood, which are at a maximum distance of K steps from the central node. One is allowed to choose the value of K freely to balance the model performance [10] or further simplify the polynomial filter, such as setting K=1 [20].

3.2 High-order graph convolutions

When attempting to design graph convolutions that capture high-order information, a common intuition is to increase the order of the polynomial filter. This can enhance the receptive field of the filter. For example, expanding the order of the Chebyshev polynomial to 2, we can express it as:

$$\begin{aligned} x \star g &\approx \sum_{k=0}^{2} \beta_k T_k(\hat{L}) x \\ &= \beta_0 T_0(\hat{L}) x + \beta_1 T_1(\hat{L}) x + \beta_2 T_2(\hat{L}) x \\ &= \beta_0 x + \beta_1 \hat{L} x + \beta_2 (2\hat{L}^2 - I) x \\ &= [x, \hat{L}x, 2\hat{L}^2 x - x]\theta, \end{aligned} \tag{12}$$

with $\theta = [\beta_0, \beta_1, \beta_2]^T$. By employing such formulation, it appears to enhance the amount of neighborhood information that the spectral filter can capture, comes at the cost of increased model complexity and slower algorithm convergence. However, in practice, the structure of features is often complex, nonlinear, and contaminated with noise. A significant hidden flaw in the previous approach is its reliance on the Laplacian, which is designed based on pairwise similarity between points and has a linear structure. As a result, there are missing and erroneous characterizations of data correlations. Increasing the order of the polynomials filter directly may even inadvertently amplify this error since a L^2 term was added earlier [17]. In particular, these limitations become fatal when addressing high-dimensional feature.

To achieve the acquisition of high-order neighborhood correlations while mitigating the impact of deviation and noise, we present a carefully-crafted and novel graph convolutional kernel capable of simultaneously capturing nodes feature from themselves, as well as their first-order and second-order neighborhoods. The proposed polynomial filter operates on a single graph signal x as follows:

$$Y_k(x) = \begin{cases} Ix & \text{k=0} \\ \hat{L}x & \text{k=1} \\ (\mathcal{L}_3 \times_3 x^T \times_2 x^T) - x & \text{k=2} \end{cases}. \tag{13}$$

Here, Y_1 corresponds to the nodes themselves, Y_2 denotes the first-order neighborhood characterized by the conventional Laplacian matrix, and Y_3 is leveraged to exploit the latent representations among samples in the second-order neighborhood. Graph spectral convolution that integrates high-order tensor similarity with low-order neighborhood information is then defined as:

$$g \star_t x = (||_{k=0}^{2} Y_k(x))\theta, \tag{14}$$

where '||' concatenates feature representation from three distinct domains, note that the polynomial coefficient θ is learnable. The specific steps for constructing Tensor-GCN model will be outlined in the subsequent sections.

3.3 Implementation

3.3.1 Graph construction. Tensor-GCN framework focus on classification tasks on feature graph of HDLSS dataset. To this end, we first construct graph structure based on sample features X. Specifically, we calculate the pairwise similarity matrix S and high-order similarity tensor $\mathcal{T}_3$, later, traditional Laplacian matrix L and third-order Laplacian tensor $\mathcal{L}_3$ are formulated utilizing S and $\mathcal{T}_3$, respectively.

For pairwise Laplacian matrix, we begin with computing the distances between samples. Here, the Euclidean distance is employed to derive a distance matrix E, which consists of distance of the k-nearest neighbors for each sample. Subsequently, a Gaussian kernel function is applied to filter E, yielding the similarity matrix S. which is defined as:

$$S_{ij}(d) = e^{-\frac{d^2}{2\sigma^2}}, \tag{15}$$

where d denotes the Euclidean distance between node i and node j, σ determines the width or standard deviation of the distribution. A smaller value of σ results in a sharper kernel, while larger one produces a smoother kernel, here we utilize the average of distance E as σ. We then have graph Laplacian $L = I - D^{-\frac{1}{2}}SD^{-\frac{1}{2}}$. The acquisition method for $\mathcal{L}_3$ has been discussed in section 2.3.2.

3.3.2 Simplification of the Tensor-GCN Model. Let graph $X \in \mathbb{R}^{N\times C}$, $\hat{L}$ is re-sclaed from L, we can derive the output $Z \in \mathbb{R}^{N\times F}$ of a single-layer Tensor-GCN model generalizing Eq.(14):

$$Z = [X, \hat{L}X, M - X]\Theta, \tag{16}$$

with $M = \begin{bmatrix}\mathcal{L}_3 \times_2 x_1^T \times_3 x_1^T & \cdots & \mathcal{L}_3 \times_2 x_C^T \times_3 x_C^T\end{bmatrix}$ and trainable parameter $\Theta \in \mathbb{R}^{3C\times F}$. This framework considers representations from different domains comprehensively to seek embeddings. Nonetheless, multiple iterations of k-mode product will result in significant computational overhead. Note that $\mathcal{L}_3 \times_2 x^T \times_3 x^T = L_3(x \otimes x)$, L_3 is folded from $\mathcal{L}_3$ along mode-3 direction. Meanwhile, based on the definition of the Khatri-Rao product [30], we can express M as:

$$\begin{aligned} M &= \begin{bmatrix}\mathcal{L}_3 \times_2 x_1^T \times_3 x_1^T & \cdots & \mathcal{L}_3 \times_2 x_C^T \times_3 x_C^T\end{bmatrix} \\ &= \begin{bmatrix}\hat{L}_3(x_1 \otimes x_1) & \cdots & \hat{L}_3(x_C \otimes x_C)\end{bmatrix} \\ &= \begin{bmatrix}\hat{L}_3 X * X\end{bmatrix} \end{aligned} \tag{17}$$

As a result, we can build a tensor-based convolutional layer $Z(X, \Theta)$ in the following formulation:

$$Z = \begin{bmatrix}X & \hat{L}X & (\hat{L}_3 X * X - X)\end{bmatrix}\Theta \tag{18}$$

3.3.3 Layer-wise model for node classification. In semi-supervised classification, only a small fraction of nodes are labeled. The model utilizes these labeled nodes along with the graph structure to make predictions for all vertices. We first construct graph structure as the section above, which yields the pairwise similarity S and tensor similarity $\mathcal{T}_3$. Then we calculate Laplacian matrices $\hat{L}$ and $\hat{L}_3$, feed them along with feature X as inputs into our defined multi-layer Tensor-GCN model, and the iterative process for each layer is as follows:

$$X^{(l+1)} = \sigma(\begin{bmatrix}X^{(l)} & \hat{L}X^{(l)} & (\hat{L}_3 X^{(l)} * X^{(l)} - X^{(l)})\end{bmatrix} W^{(l)}), \tag{19}$$

where $X^{(l)}$ denotes the graph signal at l layer, $X^{(0)} = X$, $W^{(l)}$ being trainable parameter and σ is the activation function, we apply softmax function row-wise here. During training, the Tensor GCN model updates the parameter matrix W via back-propagation. We evaluate the cross-entropy error over training data:

$$Loss = -\sum_{i\in Y_l}\sum_{j=1}^{F} Y_{ij} ln Z_{ij}, \tag{20}$$

with Y_l denotes the set of labeled nodes. Our framework exploits representations from three distinct neighborhoods and conducts meticulous fusion, allowing for more precise and robust predictions, even in HDLSS scenarios.

4 EXPERIMENTS

In this section, we demonstrate the superiority of the proposed Tensor GCN method by comparing it with other state-of-the-art GCN methods.

4.1 Experimental Settings

Table 1: The statistics of the datasets

Dataset	Instances	Features	Classes
Colon	62	2000	2
Leukemia	72	7070	2
ALLAML	72	7129	2
GLI	85	22283	2
Prostate GE	102	5966	2
CLL_SUB	111	11340	3
SMK_CAN	187	19993	2
Lung	203	3312	5

4.1.1 Datasets and Baselines. In this experiment, we validated the performance of Tensor-GCN by applying it to six public biological datasets summarized in Table 1. These datasets are all affected by the challenges of high-dimensional and small-sample size.

- **Leukemia** dataset is the gene expression data from 72 patients with Acute Lymphoblastic Leukemia (ALL) and normal controls. Each instance has 7070 features.
- **ALLAML** dataset consists of gene expression data from 72 leukemia patients, classified into two categories: Acute Lymphoblastic Leukemia (ALL) and Acute Myeloid Leukemia (AML), each sample has 7,129 features.
- **GLI-85** dataset consists of transcriptional data from 85 cases of Diffuse Intrinsic Pontine Glioma (DIPG) in 74 patients. Each instance has 22,283 features. These gliomas are classified into two categories.
- **Prostate_GE** dataset is the gene expression data of prostate cancer patients, consisting of 102 instances with 5,966 features per instance.
- **Lung** dataset is the gene expression data of lung cancer patients, containing a total of 203 samples with 5 categories.

NEED MODIFY We compare our proposed Tensor-GCN with state-of-the-art graph convolutional network models, and for reproducibility, we provide source code for these methods in the supplement.

- **Chebnet** [10]
- **GCN** [20]
- **GAT** [35]

4.1.2 Experimental settings. In this study, we construct kNN graph for features using the method presented in Section 3.3.1, as the HDLSS datasets lack natural graph structure. Due to the limited

sample size, we set k to 5. Samples are partitioned into labeled and unlabeled sets at an 8:2 ratio, with 20% of the labeled samples. Baseline methods are initialized with the parameters suggested in their respective works. In addition, we carefully tune the parameters during training to ensure that baseline model achieves optimal performance.

A multilayer Tensor-GCN is implemented in this experiment, which is trained utilizing Adam optimizer with 0.005 learning rate. The subsequent parameter values is determined via grid search. The number of layers N ranged from 1 to 8, and the hidden layer size for each layer is searched in $\{8, 16, 32, 64, 128, 256, 512\}$. Additionally, dropout [31]$\in \{0.4...0.9\}$ and weight decay [24] $\in \{5e-5, 5e-4, 5e-3, 5e-2, 5e-1\}$ is introduced to alleviate overfitting. The specific parameter settings for each dataset will be provided in the supplement for reproducibility. Furthermore, to comprehensively assess the ability of Tensor-GCN and the compared methods to address the HDLSS problem, we use four common evaluation metrics, namely, Accuracy (ACC) , F1-score, Area Under the Curve (AUC) , and Recall. In this experiment, each method was run 10 times under the same split and report average results.

4.2 Node Classification

The experimental results on the HDLSS dataset are presented in Table 2, where bold value indicates the best performing method and underlined values represent the runner-up. As demonstrated in the results, we have the following observations:

1. Our proposed Tensor-GCN outperforms all baseline methods in the vast majority of cases. In particular, Tensor-GCN achieves a significant ACC improvement of up to 10% compared to other methods on the Lung dataset.
2. Tensor-GCN performs better than GCN and ChebNet on all datasets, which further demonstrates that the high-order similarity introduced by our approach can provide a complementary information to the representation extracted by low-pass filters.
3. Explain from the way of exploiting high-dimensional message

4.3 Analysis of Variants

4.4 Visualization

5 CONCLUSION

In this paper, we presents a tensor-based graph convolutional network (Tensor-GCN) for addressing the problem of semi-supervised classification in high-dimensional low-sample size (HDLSS) data. The framework leverages tensor similarity to capture higher-order relationships among samples, thereby overcoming the limitations of traditional pairwise similarity-based graph convolutional networks. Moreover, we incorporates a multi-layered network architecture that seamlessly integrates low-order and high-order neighborhood information, enabling deep exploration of sample features. Comparisons with state-of-the-art GCNs on HDLSS data, demonstrate the effectiveness of the proposed Tensor GCN model.

6 ACKNOWLEDGMENTS

REFERENCES

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. 2019. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*. PMLR, 21–29.

[2] Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. 2017. Learning to represent programs with graphs. *arXiv preprint arXiv:1711.00740* (2017).

[3] James Atwood and Don Towsley. 2016. Diffusion-convolutional neural networks. *Advances in neural information processing systems* 29 (2016).

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

[5] Hongmin Cai, Fei Qi, Junyu Li, Yu Hu, Yue Zhang, Yiu-ming Cheung, and Bin Hu. 2023. Uniform tensor clustering by jointly exploring sample affinities of various orders. *arXiv preprint arXiv:2302.01569* (2023).

[6] Can Chen, Scott T Weiss, and Yang-Yu Liu. 2023. Graph convolutional network-based feature selection for high-dimensional and low-sample size data. *Bioinformatics* 39, 4 (2023), btad135.

[7] Yuzhou Chen, Yulia R Gel, and H Vincent Poor. 2022. BScNets: Block simplicial complex neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 6333–6341.

[8] Zhao-Min Chen, Xiu-Shen Wei, Peng Wang, and Yanwen Guo. 2019. Multi-label image recognition with graph convolutional networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5177–5186.

[9] Ganqu Cui, Jie Zhou, Cheng Yang, and Zhiyuan Liu. 2020. Adaptive graph encoder for attributed graph embedding. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*. 976–985.

[10] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems* 29 (2016).

[11] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 3558–3565.

[12] Damien François, Vincent Wertz, and Michel Verleysen. 2007. The concentration of fractional distances. *IEEE Transactions on Knowledge and Data Engineering* 19, 7 (2007), 873–886.

[13] Yue Gao, Yifan Feng, Shuyi Ji, and Rongrong Ji. 2022. HGNN+: General hypergraph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 3 (2022), 3181–3199.

[14] Peter Hall, James Stephen Marron, and Amnon Neeman. 2005. Geometric representation of high dimension, low sample size data. *Journal of the Royal Statistical Society Series B: Statistical Methodology* 67, 3 (2005), 427–444.

[15] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[16] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis* 30, 2 (2011), 129–150.

[17] Mingguo He, Zhewei Wei, and Ji-Rong Wen. 2022. Convolutional neural networks on graphs with chebyshev approximation, revisited. *Advances in Neural Information Processing Systems* 35 (2022), 7264–7276.

[18] Mingguo He, Zhewei Wei, Hongteng Xu, et al. 2021. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. *Advances in Neural Information Processing Systems* 34 (2021), 14239–14251.

[19] Yiming Huang, Yujie Zeng, Qiang Wu, and Linyuan Lü. 2023. Higher-order graph convolutional network with flower-petals laplacians on simplicial complexes. *arXiv preprint arXiv:2309.12971* (2023).

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

[21] Tamara G Kolda and Brett W Bader. 2009. Tensor decompositions and applications. *SIAM review* 51, 3 (2009), 455–500.

[22] Haoyang Li, Shuye Tian, Yu Li, Qiming Fang, Renbo Tan, Yijie Pan, Chao Huang, Ying Xu, and Xin Gao. 2020. Modern deep learning in bioinformatics. *Journal of molecular cell biology* 12, 11 (2020), 823–827.

[23] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

[24] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

[25] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. 2017. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5115–5124.

[26] Hong Peng, Yu Hu, Jiazhou Chen, Haiyan Wang, Yang Li, and Hongmin Cai. 2020. Integrating tensor similarity to enhance clustering performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 5 (2020), 2582–2593.

[27] Stephan Rabanser, Oleksandr Shchur, and Stephan Günnemann. 2017. Introduction to tensor decompositions and their applications in machine learning. *arXiv preprint arXiv:1711.10781* (2017).

[28] Liran Shen, Meng Joo Er, and Qingbo Yin. 2022. Classification for high-dimension low-sample size data. *Pattern Recognition* 130 (2022), 108828.

Table 2: Comparison of the performance of models under HDLSS dataset

Dataset	Method	ACC	F-Score	AUC	Recall
ALLAML	ChebNet	82.7586	1.0	1.0	1.0
	GCN	82.7586	1.0	1.0	1.0
	TGCN(Decomposable)	79.3103	0.7947	0.7481	0.8056
	TGCN(UTC)(Ours)	**84.4827**	0.8464	0.8	0.8571
Leukemia	ChebNet	89.6551	1.0	1.0	1.0
	GCN	89.6551	1.0	1.0	1.0
	TGCN(Decomposable)	89.6551	1.0	1.0	1.0
	TGCN(UTC)(Ours)	89.6551	1.0	1.0	1.0
GLI_85	ChebNet	82.3529	0.8849	0.9091	0.8824
	GCN	82.3529	0.942	0.9545	0.9412
	TGCN(Decomposable)	86.7647	0.8475	0.7799	0.8588
	TGCN(UTC)(Ours)	85.2941	0.8775	0.8292	0.8824
Lung	ChebNet	79.1411	0.7311	0.971	0.8
	GCN	82.8220	0.8073	0.9457	0.8374
	TGCN(Decomposable)	85.2458	0.8604	0.9668	0.9
	TGCN(UTC)(Ours)	**95.7055**	1.0	1.0	1.0
Prostate_GE	ChebNet	65.4320	0.5916	0.6318	0.619
	GCN	82.7160	0.8626	0.8623	0.8627
	TGCN(Decomposable)	81.4814	0.8520	0.8546	0.8529
	TGCN(UTC)(Ours)	**85.1851**	0.8824	0.8823	0.8824

[29] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine* 30, 3 (2013), 83–98.

[30] Age K Smilde, Rasmus Bro, and Paul Geladi. 2005. *Multi-way analysis: applications in the chemical sciences*. John Wiley & Sons.

[31] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[32] Felipe Petroski Such, Shagan Sah, Miguel Alexander Dominguez, Suhas Pillai, Chao Zhang, Andrew Michael, Nathan D Cahill, and Raymond Ptucha. 2017. Robust spatial filtering with graph convolutional neural networks. *IEEE Journal of Selected Topics in Signal Processing* 11, 6 (2017), 884–896.

[33] Gabriel Taubin. 1995. A signal processing approach to fair surface design. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. 351–358.

[34] Emil Uffelmann, Qin Qin Huang, Nchangwi Syntia Munung, Jantina De Vries, Yukinori Okada, Alicia R Martin, Hilary C Martin, Tuuli Lappalainen, and Danielle Posthuma. 2021. Genome-wide association studies. *Nature Reviews Methods Primers* 1, 1 (2021), 59.

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

[36] Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 7370–7377.

A SUPPLEMENT

A.1 Baselines and Datasets

Code to reproduce our experiments is available at the following URLs:

A.2 Experiments Setting

A.3 Implementation Details

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009