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

Anonymous Author(s)

Abstract

Semi-supervised classification aims to predict labels for all samples using only limited labeled samples. Graph Convolutional Networks (GCN) have achieved great success in semi-supervised small-sample classification due to their integration capabilities with data graph structures. However, when dealing with High-Dimensional and Low-Sample Size (HDLSS) data, such methods are prone to overfitting due to concentration effects. To address this issue, we introduced high-order tensor similarity to describe relationships among multiple samples. Building upon this, we proposed the Tensor Graph Convolutional Network (Tensor-GCN) that effectively integrates traditional graph information with high-order graph information. By employing a multi-layer Tensor-GCN framework, traditional pairwise information with high-order neighborhood information is seamlessly integrated, thus achieving more accurate and robust predictive capabilities. Extensive experiments on public HDLSS datasets indicate that Tensor-GCN can exploit higher-order feature information and exhibit superior predictive performance and robustness for HDLSS data.

CCS Concepts

• **Computing methodologies** → **Semi-supervised learning settings**; • **Theory of computation** → **Semi-supervised learning**.

Keywords

Graph convolutional networks, network representation learning, semi-supervised classification, high-order similarity

ACM Reference Format:

Anonymous Author(s). 2018. Tensor 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 email (Conference acronym 'XX)*. ACM, New York, NY, USA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Semi-supervised learning utilizes limited labeled data and unlabeled data simultaneously to enhance model performance, particularly showcasing advantages in handling complex data with high label acquisition costs [15, 24, 31, 36, 39, 42]. This technology holds great promise in the analysis of complex data in fields such as graphs [13, 22, 43], images [8], text [2, 44], and bioinformatics [6, 35]. In this scenario, the characteristic of semi-supervised classification is to extracting discriminative correlations with limited labeled samples. Fortunately, graph-based methods show their superiority in analyzing the intrinsic properties of complex data, thereby achieving significant success in semi-supervised classification tasks. Meanwhile, Graph Convolutional Networks (GCN) synergistically harness the salient attributes of the aforementioned methodologies, endowing them with formidable capabilities to extract intricate correlations in data.

GCN can be broadly categorized into two types based on different theoretical derivations: Spectral approaches and Spatial approaches. For Spatial approaches, the convolution operation is defined for groups of spatially neighboring nodes. DCNNs [3] utilize the powers of a transition matrix to define the neighborhood of nodes. MoNet [27] employs local path operators in the form of Gaussian mixture models to generalize convolution in the spatial domain. GAT [41] assigns different weights to neighboring nodes based on their importance, allowing the model to focus on more informative nodes during aggregation. GraphSAGE [17] leverages a sampling strategy to select representative neighboring nodes and aggregates their features to update node representations.

On the other hand, Spectral approaches transform node features from node space to the spectral domain through Fourier transformation for convolution operations, often involving the computation of eigenvectors. In GCN [22], an efficient layer-wise propagation framework is proposed by simplify the Chebyshev polynomials to first-order. GCN-based methods have played a significant role in advancing the field of Graph Neural Network and have been widely used as foundational approaches for graph-based learning tasks. Traditional spectral GCN primarily follows an information propagation manner to aggregate feature representation from neighboring nodes. Its essence lies in the Laplacian smoothing of node features [25, 38], ultimately facilitating the propagation of node features within the network structure and forming graph embeddings. This low-pass filtering characteristic [9, 28] has contributed to the remarkable success of GCN in the past few years.

However, there are certain inherent limitations of GCN. Firstly, the graph structure in GCN models utilizes only the pairwise relationships between samples, overlooking the high-order relationships that exist in real-world data. In practice, complex patterns and dependencies often extend beyond simple pairwise relationships, making it challenging for graph based methods to capture the associations between data accurately [11]. Additionally, real-world data with high-dimensional characteristics typically face challenges in describing data correlations due to concentration effects [12, 16] and noise interference. Ultimately, this further impacts the ability of GCN to extract meaningful representations from input data.

Recently, high-order methods have been proposed to address the aforementioned shortcomings of GCN. MixHop [1] was proposed to learn difference operators by repeatedly mixing feature

Conference acronym 'XX, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

representations of neighbors at various distances. BScNet [7] replaces the graph Laplacian with the block Hodge Laplacian to obtain high-order feature representations. GRACES [6] introduces feature selection to deal with HDLSS data. HiGCN [37] employs a hierarchical GCN model to comprehensively consider information from both the feature space and the sample space. Alternatively, Gao and Feng et al. [11, 14] attempt to utilize hypergraph to encode high-order correlation, which introduced hyperedge to represent high-order relationship between samples. However, these methods still do not fundamentally address the limitations of pairwise-relationship-based methods when handling high-dimension m yet low-sample size n (HDLSS) data with $m \gg n$. For instance, hypergraph methods essentially use high-order relationships to infer an approximate sample-to-sample similarity matrix.

In the pursuit of naturally modeling data correlations and harnessing high-order representations, we propose Tensor Graph Convolutional Network (Tensor-GCN), which measures relationships between samples by integrating multi-order similarities, providing a new insight for data modeling. The framework allows for direct modeling of higher-order relationships among samples and seamlessly integrates high-order neighborhood message into the graph convolution process. Furthermore, a deep convolutional network is employed to leverage multi-order similarities for exploiting latent embedding. Our main contributions are summarized as follows:

- An inventively tensor similarity is adopted in our graph convolution process, which depicts intrinsic links among multiple samples and therefore provides complementary information that the pairwise similarity missed.
- We formulate a meticulously-designed multi-layer 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 baseline methods on public HDLSS datasets demonstrate significant advantages and robustness in semi-supervised classification, indicating that Tensor-GCN is capable of enhancing node representations while well-suited for HDLSS data.

The remaining content will be organized in the following manner: Section 2 introduces relevant definitions and fundamental concepts necessary for this paper. In Section 3 we presents the proposed Tensor-GCN model and its implementation. Section 4 showcases the experimental results and analysis on the HDLSS dataset. Finally, we summarizes the paper in Section 5.

2 Preliminaries

2.1 Notations

In this paper, we employ calligraphic, uppercase, and lowercase letters to represent tensors, matrices, and vectors, respectively. For a third-order 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 third-order tensor unfold operations is as follows:

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

$$\boldsymbol{T}_3 = \text{unfolding}(\mathcal{T}_3) = \begin{bmatrix} \mathcal{T}_3^{(1)} \\ \mathcal{T}_3^{(2)} \\ \vdots \\ \mathcal{T}_3^{(N)} \end{bmatrix}. \tag{1}$$

Several matrix/tensor products are important in the sections that follow, namely the Hadamard product, Kronecker product, Khatri-Rao product, and k-mode product [30], we briefly define them here.

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

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

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

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} \boldsymbol{a}_{11}\boldsymbol{B} & \boldsymbol{a}_{12}\boldsymbol{B} & \cdots & \boldsymbol{a}_{1J}\boldsymbol{B} \\ \boldsymbol{a}_{21}\boldsymbol{B} & \boldsymbol{a}_{22}\boldsymbol{B} & \cdots & \boldsymbol{a}_{2J}\boldsymbol{B} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{a}_{I1}\boldsymbol{B} & \boldsymbol{a}_{I2}\boldsymbol{B} & \cdots & \boldsymbol{a}_{IJ}\boldsymbol{B} \end{bmatrix} \in \mathbb{R}^{IK\times JL}. \tag{3}$$

Definition 2.4. ***Khatri-Rao Product*** The Khatri-Rao product [33] is the "matching columnwise" Kronecker product. Given matrices $\boldsymbol{A} \in \mathbb{R}^{I\times K}$ and $\boldsymbol{B} \in \mathbb{R}^{J\times K}$, the Khatri-Rao product is defined as:

$$\boldsymbol{A} * \boldsymbol{B} = \begin{bmatrix} \boldsymbol{a}_1 \otimes \boldsymbol{b}_1 & \boldsymbol{a}_2 \otimes \boldsymbol{b}_2 & \cdots & \boldsymbol{a}_K \otimes \boldsymbol{b}_K \end{bmatrix} \in \mathbb{R}^{IJ\times K}. \tag{4}$$

The Khatri–Rao and Kronecker products are identical if $\boldsymbol{a}$ and $\boldsymbol{b}$ are vectors, i.e., $\boldsymbol{a} \otimes \boldsymbol{b} = \boldsymbol{a} * \boldsymbol{b}$. Tensor multiplication is much more complex than matrix multiplication, here we consider only the tensor k-mode product [23], 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 $\boldsymbol{V} \in \mathbb{R}^{P\times I_k}$, denoted by $\mathcal{T} \otimes_k \boldsymbol{V} \in \mathbb{R}^{I_1\times\cdots\times I_{k-1}\times P\times I_{k+1}\times\cdots\times I_m}$, with

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

2.2 Revisit Spectral Graph Convolution

Spectral graph convolution performs convolution operations by multiplying the input signal with convolution kernels in the spectral domain, typically implemented using Fourier transformation. This manner utilizes the eigenvectors $\boldsymbol{U}$ of the graph Laplacian matrix $\boldsymbol{L}$ based on input data $\boldsymbol{X}$ as basis functions [32]. The normalized Laplacian matrix can be obtained from the similarity matrix and is defined as:

$$\hat{\boldsymbol{L}} = \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{S}\boldsymbol{D}^{-\frac{1}{2}}, \tag{6}$$

where $\boldsymbol{S}$ represents the similarity matrix and diagonal degree matrix $\boldsymbol{D}$ is denoted as $\boldsymbol{D}_{ii} = \sum_{j=1}^{n} \boldsymbol{S}_{ij}$. The eigenvectors $\boldsymbol{U}$ of the Laplacian $\boldsymbol{L}$ are used as the basis functions for transforming the graph signal.

Given a graph signal $\boldsymbol{x} \in \mathbb{R}^N$ and a kernel g, a spectral graph convolution can be expressed as:

$$g \star \boldsymbol{x} = \boldsymbol{U}((\boldsymbol{U}^\top g) \odot (\boldsymbol{U}^\top \boldsymbol{x})) = \boldsymbol{U}(g(\Lambda)\boldsymbol{U}^\top \boldsymbol{x}), \tag{7}$$

where $g(\Lambda)$ is the counterpart of g in the Fourier domain, which is often viewed as a function of the Laplacian eigenvalues [22], i.e.,

$$g(\Lambda) = \mathrm{diag}(g(\lambda_1), g(\lambda_2), \cdots, g(\lambda_n)). \tag{8}$$

In the primary GCN approach, the number of parameters in a single convolutional kernel is linearly related to the number of samples, leading to computational burden. In practice, many methods [18, 20, 32] 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. [18] suggested that K-th order Chebyshev polynomials can be leveraged to approximate $g(\boldsymbol{\Lambda})$ effectively as:

$$g(\boldsymbol{\Lambda}) \approx \sum_{k=0}^{K} \beta_k T_k(\hat{\boldsymbol{\Lambda}}), \tag{9}$$

where $\hat{\boldsymbol{\Lambda}} = \frac{2}{\lambda_{\max}}\boldsymbol{\Lambda} - \boldsymbol{D}$ is rescaled from $\boldsymbol{\Lambda}$, $[\beta_0, \ldots, \beta_K] \in \mathbb{R}^K$ is a vector of Chebyshev coefficients. The Chebyshev polynomials are defined as $T_k(\boldsymbol{x}) = 2\boldsymbol{x}T_{k-1}(\boldsymbol{x}) - T_{k-2}(\boldsymbol{x})$, with $T_0(\boldsymbol{x}) = 1$ and $T_1(\boldsymbol{x}) = \boldsymbol{x}$.

Based on the aforementioned approximation, graph convolution in the spectral domain is simplified to the following form:

$$g \star \boldsymbol{x} \approx \sum_{k=0}^{K} \beta_k T_k(\hat{\boldsymbol{L}})\boldsymbol{x}, \tag{10}$$

where $\hat{\boldsymbol{L}}$ undergoes the same rescaling process as $\hat{\boldsymbol{\Lambda}}$, we approximate $\lambda_{\max} \approx 2$ according to [22]. 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 $K = 1$ [22].

2.3 Tensor Spectral Analysis

As a fundamental tool in various computational tasks such as clustering, classification, and recommendation systems, the choice of similarity measurement between samples plays a crucial role in the effectiveness of the final task. Traditional methods mostly rely on measuring the relationships between any two samples. For instance, the Euclidean distance is a popular similarity measure for numerical data, while the Gaussian kernel function is suitable for nonlinearly separable data. Other measures such as cosine similarity, Jaccard similarity, and Pearson correlation coefficient offer additional options depending on the nature of the data being analyzed.

However, these traditional pairwise similarity measures exhibit limitations in extracting high-order information. The reduction of intricate interactions to pairwise simplifications inevitably leads to a loss of valuable message. Researchers have been striving to identify an effective approach for representing the intrinsic relationships among data, Cai et al. [4, 5, 29] propose that characterizing high-order sample correlations via high-order tensor similarity. For instance, the connection among three samples can be represented by a third-order tensor $\boldsymbol{\mathcal{T}}_3 = [\boldsymbol{\mathcal{T}}_{ijk}] \in \mathbb{R}^{N\times N\times N}$. An intuitive approach is to utilize composite similarity based on pairwise similarity $\boldsymbol{S}$, where each entry $\boldsymbol{T}_{ijk}$ is given by:

$$\boldsymbol{\mathcal{T}}_{3_{ijk}} = \boldsymbol{S}_{ij}\boldsymbol{S}_{kj}. \tag{11}$$

According to the definition in Eq. (1), the decomposable third-order similarity $\boldsymbol{\mathcal{T}}_3$, as defined in Eq. (11), can be unfolded into matrix $\hat{\boldsymbol{T}}_3 \in \mathbb{R}^{N^2\times N}$. $\hat{\boldsymbol{T}}_3$ can be further written by:

$$\begin{aligned}
\boldsymbol{T}_3 &= \begin{bmatrix} \boldsymbol{\mathcal{T}}_{3_{111}} & \cdots & \boldsymbol{\mathcal{T}}_{3_{1n1}} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\mathcal{T}}_{3_{n1n}} & \cdots & \boldsymbol{\mathcal{T}}_{3_{nnn}} \end{bmatrix} \\
&= \begin{bmatrix} \boldsymbol{S}_{11}\boldsymbol{S}_{:1} & \cdots & \boldsymbol{S}_{1N}\boldsymbol{S}_{:N} \\ \vdots & \ddots & \vdots \\ \boldsymbol{S}_{N1}\boldsymbol{S}_{:1} & \cdots & \boldsymbol{S}_{nn}\boldsymbol{S}_{:N} \end{bmatrix} \\
&= \begin{bmatrix} \boldsymbol{S}_{:1} \otimes \boldsymbol{S}_{:1} & \cdots & \boldsymbol{S}_{:N} \otimes \boldsymbol{S}_{:N} \end{bmatrix} \\
&= \boldsymbol{S} * \boldsymbol{S}.
\end{aligned} \tag{12}$$

Eq. (12) reveals that a decomposable tensor similarity can be obtained through the Khatri-Rao product of two similarity matrices. Let a matrix $\hat{\boldsymbol{L}}_3 = \boldsymbol{D}_{3_1}^{-\frac{1}{2}}\boldsymbol{T}_3\boldsymbol{D}_{3_2}^{-\frac{1}{2}}$ with diagonal matrices $\boldsymbol{D}_{3_1}$, $\boldsymbol{D}_{3_2}$ given by $\boldsymbol{D}_{3_1} = \sqrt{\sum_i \boldsymbol{T}_{3:i} \sum_i \boldsymbol{T}_{3:i}}$ and $\boldsymbol{D}_{3_2} = \sum_i \hat{\boldsymbol{T}}_{3_{:i}}$. We can obtain the normalized third-order similarity tensor $\boldsymbol{\mathcal{L}}_3$ via folding $\hat{\boldsymbol{L}}_3$. Furthermore, assuming that $\hat{\boldsymbol{L}}$ is the normalized Laplacian matrix, which is constructed from the kNN (k-Nearest Neighbor) graph generated based on $\boldsymbol{S}$, it has been demonstrated by [29] that:

$$\hat{\boldsymbol{L}}_3 = \hat{\boldsymbol{L}} * \hat{\boldsymbol{L}}. \tag{13}$$

The two equations of Eq. (12) and Eq. (13) bridge the third-order similarity $\boldsymbol{\mathcal{T}}_3$ and pairwise similarity $\boldsymbol{S}$ together via the Khatri-Rao product. Indeed, this also indicates that the structural feature extracted from decomposable high-order similarity is inherently determined by 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, necessitating the design of novel metrics capable of extracting complementary information beyond the scope of Eq. (12). For the entire elimination of reliance on pairwise relationships, [29] designed an indecomposable tensor similarity metric based on their observation of spatial relationships within the sample data. Indecomposable tensor similarity will be introduced in section 3.2.1. In addition, the reader is referred to [4, 29] for an in-depth discussion of tensor similarity.

3 Tensor-GCN: The Proposed Model

In this section, we introduce the Tensor-GCN. We begin with the introduction of the tensor graph convolution architecture. Subsequently, the implementation details of the proposed Tensor-GCN are presented. The overall framework of Tensor-GCN is shown in Figure 1.

3.1 High-order Graph Convolutions

A common intuition to design high-order graph convolutions is to increase the order of the polynomial, thereby enhancing the

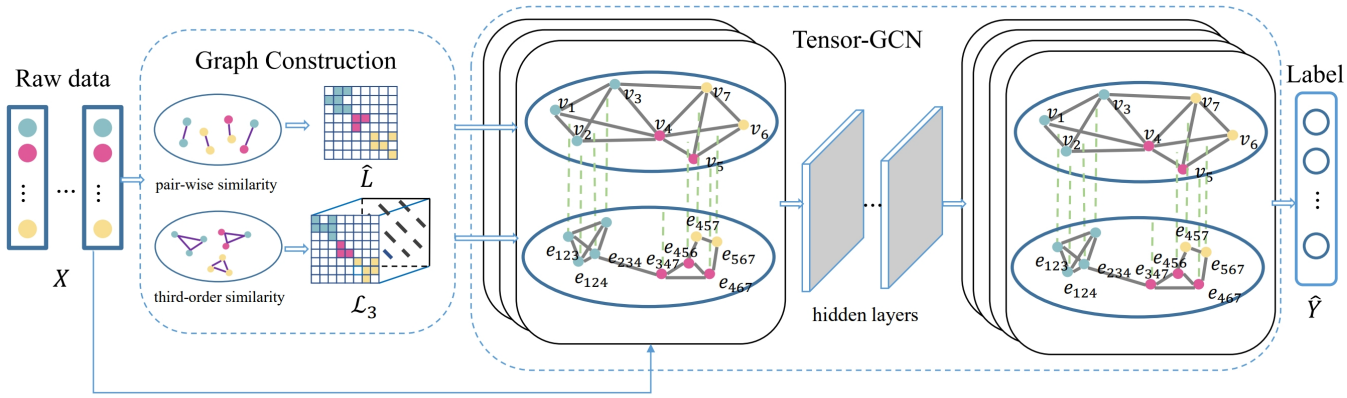


Figure 1: Tensor-GCN framework. Node feature X is to construct second-order Laplacian matrix and third-order Laplacian tensor, respectively. Subsequently, multi-order similarities undergo GCN layers for feature extraction.

receptive field of the filter. For example, expanding the Chebyshev polynomial to second-order, expressed as:

$$\begin{aligned} g \star \boldsymbol{x} &\approx \sum_{k=0}^{2} \beta_k T_k(\hat{\boldsymbol{L}})\boldsymbol{x} \\ &= \beta_0 T_0(\hat{\boldsymbol{L}})\boldsymbol{x} + \beta_1 T_1(\hat{\boldsymbol{L}})\boldsymbol{x} + \beta_2 T_2(\hat{\boldsymbol{L}})\boldsymbol{x} \\ &= \beta_0 \boldsymbol{x} + \beta_1 \hat{\boldsymbol{L}}\boldsymbol{x} + \beta_2 (2\hat{\boldsymbol{L}}^2 - \boldsymbol{I})\boldsymbol{x} \\ &= [\boldsymbol{x}, \hat{\boldsymbol{L}}\boldsymbol{x}, 2\hat{\boldsymbol{L}}^2\boldsymbol{x} - \boldsymbol{x}]\boldsymbol{\theta}, \end{aligned} \tag{14}$$

with $\boldsymbol{\theta} = [\beta_0, \beta_1, \beta_2]^\top$. Eq. (14) may capture more neighborhood information as it expands the receptive field of the kernel g. However, a notable limitation of Eq. (14) lies in its reliance on a Laplacian matrix constructed using traditional pairwise similarity measures which disregards high-order message. Consequently, the Laplacian matrix may contain the missing and erroneous characterizations of data correlations and proves to be sensitive to concentration effects when dealing with HDLSS data. Increasing the order of the polynomials filter directly may even inadvertently amplify this error since a $\boldsymbol{L}^2$ term is added [19]. 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 [22]. Fundamentally, the issue lies in the loss of information by the polynomial convolution kernel during the process of Laplacian smoothing. In particular, these limitations become fatal when addressing HDLSS problems.

To capture 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 node features from themselves, as well as their first-order and second-order neighborhoods. The proposed polynomial filter operates on a single graph signal $\boldsymbol{x}$ as follows:

$$Y_k(\boldsymbol{x}) = \begin{cases} \boldsymbol{I}\boldsymbol{x}, & k{=}0 \\ \hat{\boldsymbol{L}}\boldsymbol{x}, & k{=}1 \\ (\boldsymbol{\mathcal{L}}_3 \times_3 \boldsymbol{x}^\top \times_2 \boldsymbol{x}^\top) - \boldsymbol{x}, & k{=}2 \end{cases}. \tag{15}$$

Here, Y_0 corresponds to the information about samples themselves, Y_1 denotes the first-order neighborhood characterized by the conventional Laplacian matrix, and Y_2 is leveraged to exploit the latent representations among samples in the second-order neighborhood. $\boldsymbol{\mathcal{L}}_3$ refers to the Laplacian tensor, derived from a tensor similarity measure, such as the aforementioned decomposable tensor similarity. Graph spectral convolution that integrates high-order tensor similarity with low-order neighborhood information is then defined as:

$$g \star_t \boldsymbol{x} = (||_{k=0}^{2} Y_k(\boldsymbol{x}))\boldsymbol{\theta}, \tag{16}$$

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

3.2 Implementation

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

For the pairwise Laplacian matrix, we begin with computing the distances between samples. Here, the Euclidean distance is employed to derive a distance matrix $\boldsymbol{E}$, which consists of distance of the k-nearest neighbors for each sample. Subsequently, a Gaussian kernel function is applied to filter $\boldsymbol{E}$, yielding the similarity matrix $\boldsymbol{S} = [S_{ij}]_{N\times N}$, which each entry being defined as:

$$S_{ij} = e^{-\frac{d_{ij}^2}{2\sigma^2}}, \tag{17}$$

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, we utilize the average of distance $\boldsymbol{E}$ as σ. We then have graph Laplacian $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{S}\boldsymbol{D}^{-\frac{1}{2}}$.

For the high-order similarity, we borrow the indecomposable third-order tensor similarity proposed in [4]. Inspired by the spatial relationships among samples, [4] defined an indecomposable third-order similarity to mining the relationships among three samples from different perspectives. The indecomposable similarity tensor $\mathcal{T}_3 = [\mathcal{T}_{3(ijk)}] \in \mathbb{R}^{N\times N\times N}$ is defined as follows:

$$\mathcal{T}_{3_{ijk}} = 1 - \frac{< (\boldsymbol{x}_i - \boldsymbol{x}_j),\ (\boldsymbol{x}_k - \boldsymbol{x}_j) >}{d_{ij}d_{jk} + \epsilon}, \tag{18}$$

for $i, j, k \in N$, where d_{ij} denotes the Euclidean distance between a node $\boldsymbol{x}_i$ and another node $\boldsymbol{x}_j$, ϵ is a given small parameter less than 0.001 to overcome the instability caused by a zero denominator. The spirit of this definition is: considering arbitrary three samples, $\boldsymbol{x}_i$, $\boldsymbol{x}_j$, and $\boldsymbol{x}_k$, where $\boldsymbol{x}_j$ is treated as the anchor point. Apparently, if $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$ are sufficiently close, regardless of the position of $\boldsymbol{x}_j$, $\mathcal{T}_{ijk}$ should assume a relatively large value. 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 $\boldsymbol{T}_3 \in \mathbb{R}^{N^2\times N}$ and then formulate $\boldsymbol{\mathcal{L}}_3$, which when unfolded along the mode-3 direction satisfies $\hat{\boldsymbol{L}}_3 = \boldsymbol{D}_{3_1}^{-\frac{1}{2}}\boldsymbol{T}_3\boldsymbol{D}_{3_2}^{-\frac{1}{2}}$, where $\boldsymbol{D}_{3_1} = \sqrt{\sum_i \boldsymbol{T}_{3:i}\sum_i \boldsymbol{T}_{3:i}}$ and $\boldsymbol{D}_{3_2} = \sum_i \boldsymbol{T}_{3_{:i}}$. Experimental validation conducted in [4, 29] demonstrate that indecomposable similarity can complement pairwise similarity with three-dimensional spatial information, thereby enhancing the expressive power and robustness of the model.

3.2.2 Simplification of Tensor-GCN Model. Given a graph $\boldsymbol{X} \in \mathbb{R}^{N\times C}$, with $\hat{\boldsymbol{L}}$ is rescaled from a Laplacian matrix $\boldsymbol{L}$, one can derive the output $\boldsymbol{Z} \in \mathbb{R}^{N\times F}$ of a single-layer Tensor-GCN model by generalizing Eq. (16):

$$\boldsymbol{Z} = \left[\boldsymbol{X} \quad \hat{\boldsymbol{L}}\boldsymbol{X} \quad \boldsymbol{M} - \boldsymbol{X}\right]\boldsymbol{\Theta}, \tag{19}$$

with $\boldsymbol{M} = \left[\boldsymbol{\mathcal{L}}_3 \times_2 \boldsymbol{x}_1^\top \times_3 \boldsymbol{x}_1^\top, \quad \cdots, \quad \boldsymbol{\mathcal{L}}_3 \times_2 \boldsymbol{x}_C^\top \times_3 \boldsymbol{x}_C^\top\right]$ and trainable parameter $\boldsymbol{\Theta} \in \mathbb{R}^{3C\times F}$. 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 $\boldsymbol{\mathcal{L}}_3 \times_2 \boldsymbol{x}^\top \times_3 \boldsymbol{x}^\top = \boldsymbol{L}_3^\top(\boldsymbol{x} \otimes \boldsymbol{x})$, $\boldsymbol{L}_3$ is folded from $\boldsymbol{\mathcal{L}}_3$ along mode-3 direction. Meanwhile, using the definition of the Khatri-Rao product [33], one can rewrite the matrix $\boldsymbol{M}$ as:

$$\begin{aligned}\boldsymbol{M} &= \left[\boldsymbol{\mathcal{L}}_3 \times_2 \boldsymbol{x}_1^\top \times_3 \boldsymbol{x}_1^\top, \quad \cdots, \quad \boldsymbol{\mathcal{L}}_3 \times_2 \boldsymbol{x}_C^\top \times_3 \boldsymbol{x}_C^\top\right] \\ &= \left[\hat{\boldsymbol{L}}_3^\top(\boldsymbol{x}_1 \otimes \boldsymbol{x}_1), \quad \cdots, \quad \hat{\boldsymbol{L}}_3^\top(\boldsymbol{x}_C \otimes \boldsymbol{x}_C)\right] \\ &= \left[\hat{\boldsymbol{L}}_3^\top(\boldsymbol{X} * \boldsymbol{X})\right].\end{aligned} \tag{20}$$

As a result, a tensor-based convolutional layer $\boldsymbol{Z}(\boldsymbol{X}, \boldsymbol{\Theta})$ is formulated as:

$$\boldsymbol{Z} = \left[\boldsymbol{X} \quad \hat{\boldsymbol{L}}\boldsymbol{X} \quad (\hat{\boldsymbol{L}}_3^\top\boldsymbol{X} * \boldsymbol{X} - \boldsymbol{X})\right]\boldsymbol{\Theta}. \tag{21}$$

3.2.3 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 $\boldsymbol{S}$ and tensor similarity $\mathcal{T}_3$. Then we calculate Laplacian matrices $\hat{\boldsymbol{L}}$ and $\hat{\boldsymbol{L}}_3$, feed them along with feature $\boldsymbol{X}$ as inputs into our defined multi-layer Tensor-GCN model, and the iterative process for each layer is as follows:

$$\boldsymbol{X}^{(l+1)} = \sigma(\left[\boldsymbol{X}^{(l)} \quad \hat{\boldsymbol{L}}\boldsymbol{X}^{(l)} \quad (\hat{\boldsymbol{L}}_3^\top\boldsymbol{X}^{(l)} * \boldsymbol{X}^{(l)} - \boldsymbol{X}^{(l)})\right]\boldsymbol{W}^{(l)}), \tag{22}$$

where $\boldsymbol{X}^{(l)}$ denotes the graph signal at l layer, $\boldsymbol{X}^{(0)} = \boldsymbol{X}$, $\boldsymbol{W}^{(l)}$ 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 $\boldsymbol{W}$ via backpropagation. We evaluate the cross-entropy error over training data:

$$Loss = -\sum_{i\in Y_l}\sum_{j=1}^{F} Y_{ij} ln Z_{ij}, \tag{23}$$

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. The workflow of Tensor-GCN is summarized in Figure 1.

Algorithm 1 Tensor-GCN Workflow

Input: Node feature matrix $\boldsymbol{X}$; Labeled node set Y_L;
Parameters: Number of layers N; Number of epochs T; Learning rate η
Output: Predicted labels for all nodes $\hat{Y}_{pred}$

Compute the normalized second-order Laplacian matrix $\hat{\boldsymbol{L}}$.
Compute the folded normalized third-order Laplacian tensor $\boldsymbol{L}_3$.
for $t = 0, 1, \ldots, T-1$ **do**
  {Epoch loop} **for** $l = 0, 1, \ldots, N-1$ **do** {Layer loop}
    Compute first-order features: $\boldsymbol{H}_1^{(l)} \leftarrow \hat{\boldsymbol{L}}\boldsymbol{X}^{(l)}$.
    Compute second-order features: $\boldsymbol{H}_2^{(l)} \leftarrow \left(\boldsymbol{L}_3^\top(\boldsymbol{X}^{(l)} * \boldsymbol{X}^{(l)})\right) - \boldsymbol{X}^{(l)}$.
    Concatenate features and update layer output: $\boldsymbol{Z}^{(l)} \leftarrow \sigma\left([\boldsymbol{X}^{(l)}, \boldsymbol{H}_1^{(l)}, \boldsymbol{H}_2^{(l)}]\boldsymbol{W}^{(l)}\right)$
    Compute cross-entropy loss over Y_L at layer l: $Loss = -\sum_{i\in Y_L}\sum_{j=1}^{F} Y_{ij}\ln \boldsymbol{Z}_{ij}^{(l)}$.
    Update all layer weights $\boldsymbol{W}^{(l)}$ via backpropagation using learning rate η.
  end for
end for
For each node i: $\hat{\boldsymbol{y}}_i \leftarrow \arg\max_j \boldsymbol{Z}_{ij}^{(N-1)}$.
return Predicted labels $\hat{Y}_{pred}$.

4 Experiments

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

4.1 Experimental Setup

4.1.1 Datasets. In this experiment, we validated the performance of Tensor-GCN by applying it to five public biological datasets[1]

[1] https://anonymous.4open.science/r/High-Dimensional-Low-Sample-Size-78E8

Table 1: The statistics of the datasets

Dataset	Instances	Features	Classes
Leukemia	72	7070	2
ALLAML	72	7129	2
GLI_85	85	22283	2
Prostate_GE	102	5966	2
Lung	203	3312	5

summarized in Table 1. These datasets are all affected by the challenges of HDLSS. A brief introduction to these datasets is provided as follows:

- **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, and 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.

4.1.2 Baselines. We compare our proposed Tensor-GCN with state-of-the-art graph convolutional network models. The corresponding URLs for baseline methods with official implementations are provided. The introduction to the baseline methods is given as follows:

- **MLP**, also known as Multi-layer Perceptron, is a classic and widely used feedforward neural network model.
- **ChebNet** [10] leverages the Chebyshev polynomials to approximate the spectral graph convolution, enabling it to capture graph structures at different scales effectively.
- **GCN** [22] simplifies the graph convolution process, applying a localized first-order approximation of spectral graph convolutions to efficiently learn node representations.
- **GAT** [41] incorporates attention mechanisms into the graph convolutional framework, allowing for the dynamic weighting of node contributions to achieve more expressive node representations.
- **HGNN** [11] utilizes hypergraph to mine the local structure between multiple samples, thereby constructing expressive sample similarities.
- **HiGCN** [37] is a hierarchical framework that simultaneously utilizes a sparse GCN and a feature-weighted GCN to aggregate neighbor information from both the sample space and the feature space.
- **GRACES** [6] leverages GCN to build dynamic similarity graphs, enabling it to iteratively select the most informative features from HDLSS data. It mitigates overfitting by integrating multiple dropout strategies, Gaussian noise injection, and ANOVA F-test-based correction into its gradient-driven feature selection process.

4.1.3 Comparative Metrics. In order to comprehensively assess the performance of Tensor-GCN and the compared methods to address the HDLSS problem, we employed four evaluation metrics, namely, Accuracy (ACC), F-score, Area Under the Curve (AUC) and Recall. The F-score harmoniously balances precision and recall, offering a nuanced measure of the model's ability to handle class imbalances and accurately identify positive instances. AUC assesses the discriminative power of the model by quantifying its ability to distinguish between classes across various threshold settings. For non-binary classification tasks, we adopt the OvR (One-vs-Rest) strategy to obtain AUC. Recall measures the proportion of actual positive cases that the model successfully identifies, highlighting its effectiveness in capturing relevant data points. Together, these metrics ensure a thorough and balanced evaluation of the model's performance.

4.1.4 Experimental Settings. In this study, we construct a kNN graph for features using the method presented in Section 3.2.1, 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 a 2:8 ratio, with 20% of the samples labeled. Baseline methods are initialized with the parameters suggested in their respective works. In addition, we carefully tune the parameters during training to ensure that the baseline models achieve optimal performance.

A multi-layer Tensor-GCN is implemented in this experiment, which is trained utilizing the Adam optimizer [21] with a learning rate of 0.005. The subsequent parameter values are determined via grid search. The number of layers ranged from 2 to 5, and the hidden layer size for each layer is searched in $\{8, 16, 32, 64, 128, 256, 512\}$. Additionally, dropout [34]$\in \{0.4, \cdots, 0.9\}$ and weight decay [26] $\in \{5e-5, 5e-4, 5e-3, 5e-2, 5e-1\}$ are introduced to alleviate overfitting. In this experiment, each method was run 10 times under the same split and reported average results.

4.2 Performance Comparison

4.2.1 Node Classification. The experimental results on public HDLSS datasets are presented in Table 2, where bold values indicate the best performing method. As illustrated in the results, we have the following observations:

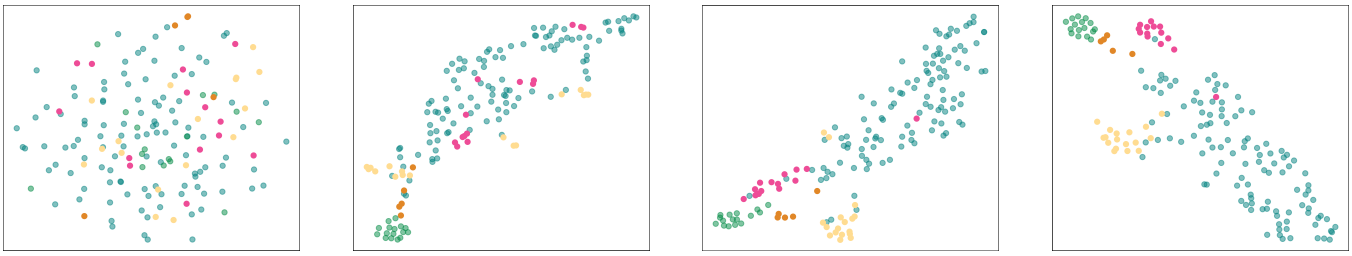
1. Tensor-GCN outperforms most baseline methods on all datasets with a notable margin. Specifically, on the Leukemia dataset, Tensor-GCN outperforms the second-best method, HGNN, by 2.07%. Similarly, on the ALLAML dataset, Tensor-GCN leads with an ACC of 83.908%, outperforming the runner-up method by 0.81%. The superiority of Tensor-GCN is further evident on the GLI_85 dataset, where it attains an ACC of 84.559%, outperforming the second-best by 2.21%, and on the Lung dataset with an impressive ACC of 94.070%, exceeding the runner-up method by 2.12%.
2. Compared with traditional GCN-based methods (GCN, ChebNet, and GAT), Tensor-GCN consistently achieves higher ACC

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

Dataset	Metric/Method	MLP	ChebNet	GCN	GAT	HGNN	HiGCN	GRACES	Tensor-GCN (Ours)
Leukemia	ACC (%)	72.586	82.184	82.759	85.334	87.586	82.826	85.235	**89.655**
	F-Score	0.7305	0.8464	0.8881	0.8160	0.8702	0.8283	0.8005	**0.9927**
	AUC	0.7754	0.8000	0.8722	0.8205	0.8318	0.8596	0.8958	**0.9444**
	Recall	0.7259	0.8571	0.8929	0.8205	0.8759	0.8283	0.8005	**0.9280**
ALLAML	ACC (%)	70.862	78.736	80.460	82.353	83.103	76.495	82.293	**83.908**
	F-Score	0.7134	0.7794	0.8251	0.7831	0.8289	0.7650	0.8019	**0.8824**
	AUC	0.7551	0.7444	0.7818	0.7647	0.8189	0.7614	0.7923	**0.8610**
	Recall	0.7086	0.7857	0.8333	0.7647	0.8310	0.7650	0.8337	**0.8869**
GLI_85	ACC (%)	70.588	78.431	80.882	80.393	82.353	73.794	74.020	**84.559**
	F-Score	0.7059	0.8326	0.8029	0.7539	0.8217	0.7181	0.7379	**0.8718**
	AUC	0.6458	0.8030	0.7223	0.7348	0.7986	0.7989	0.8194	**0.8520**
	Recall	0.7059	0.8431	0.8235	0.8402	0.8235	0.7181	0.7379	**0.8765**
Prostate_GE	ACC (%)	63.171	65.432	72.840	79.838	82.439	61.774	**84.097**	83.951
	F-Score	0.5308	0.5916	0.7568	0.7977	0.8238	0.6177	0.9231	**0.9412**
	AUC	0.6395	0.6318	0.7698	0.7993	0.8244	0.6927	**0.9473**	0.9411
	Recall	0.6317	0.6190	0.7647	0.8056	0.8277	0.6177	0.9231	**0.9412**
Lung	ACC (%)	67.362	77.914	84.049	91.951	91.779	78.528	80.368	**94.070**
	F-Score	0.5763	0.7099	0.8256	0.8826	0.8979	0.8101	0.8632	**0.9864**
	AUC	0.7410	0.8641	0.9674	0.9987	0.8899	0.8850	0.8722	**1.000**
	Recall	0.6736	0.7875	0.9624	0.8292	0.9178	0.7187	0.7924	**0.9875**

Table 3: Running time comparison of GCN and Tensor-GCN

Method/Datasets (seconds)	Leukemia	ALLAML	GLI_85	Proatete_GE	Lung
GCN	2.3148	2.1242	3.7712	2.6646	2.6293
Tensor-GCN	4.5277	4.6319	5.2959	5.1904	9.9975

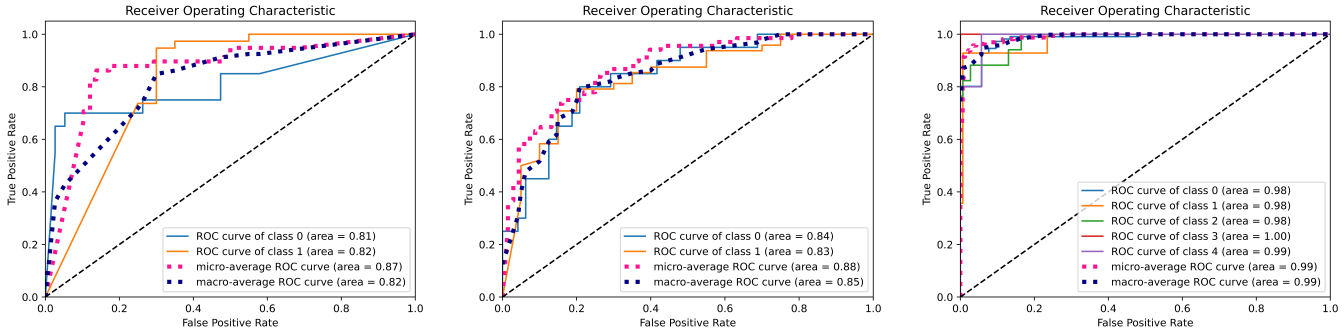


(a) Raw data **(b) GCN** **(c) Tensor-GCN-Dec** **(d) Tensor-GCN**

Figure 2: Visualization of the learned embeddings with t-SNE on Lung dataset.

across all datasets. For example, on the Leukemia dataset, Tensor-GCN obtains an ACC of 89.655%, which is 6.896%, 7.471%, and 4.321% higher than GCN (82.759%), ChebNet (82.184%), and GAT (85.334%), respectively. On the Prostate_GE dataset, it reaches 83.951%, amounting to 1.15×, 1.19×, and 4.113% improvements over GCN (72.840%), ChebNet (65.432%), and GAT (79.838%). Similarly, on the Lung dataset, Tensor-GCN achieves 94.070%, outperforming GCN by 9.021%, ChebNet by 1.17×, and GAT by 2.119%. These results demonstrate that Tensor-GCN effectively captures complex data correlations, addressing the limitations of conventional approaches.

3. Tensor-GCN consistently surpasses other high-order methods, including HGNN, HiGCN, and GRACES. Specifically, Tensor-GCN outperforms HGNN by 2.07% on the Leukemia dataset, 0.81% on ALLAML, 2.21% on GLI_85, 1.66% on Prostate_GE, and 2.29% on the Lung dataset. Additionally, compared to HiGCN, Tensor-GCN shows substantial improvements of 13.542% on Prostate_GE and 15.542% on Lung, underscoring its robust performance in capturing high-order relationships. Although GRACES



(a) ALLAML **(b) GLI_85** **(c) Lung**

Figure 3: ROC curves of Tensor-GCN on ALLAML, GLI_85 and Lung datasets

exhibits competitive performance, Tensor-GCN either matches or slightly exceeds its ACC, particularly excelling in datasets like Leukemia, ALLAML, and Lung, where Tensor-GCN achieves ACCs of 89.655%, 83.908%, and 94.070%, respectively, compared to GRACES' 85.235%, 82.293%, and 80.368%. These findings validate Tensor-GCN's effectiveness in leveraging high-order similarity to maximize information retention and enhance classification performance.

4. Beyond ACC, Tensor-GCN consistently excels in F-score, AUC, and Recall across all datasets, demonstrating its comprehensive capabilities and robustness. On the Leukemia dataset, Tensor-GCN achieves an F-score of 0.9927, an AUC of 0.9444, and a Recall of 0.9280, significantly higher than the runner-up method. These high-performance metrics indicate that Tensor-GCN not only accurately classifies instances but also maintains a strong balance between precision and recall.

The comparison experiments demonstrate the superior performance of Tensor-GCN across diverse HDLSS datasets and multiple evaluation metrics. By outperforming both classic GCN-based methods and high-order models, Tensor-GCN showcases that it is well-adapted to HDLSS environments and it can effectively distinguish samples even in the presence of concentration effects.

4.2.2 Running Time Comparison. As a sophisticated graph convolutional network architecture, Tensor-GCN requires additional time to process the complex relationships between samples. Table 3 provides the time taken to train GCN [22] and Tensor-GCN for 200 epochs on various datasets.[2] Despite requiring a few extra seconds of runtime, Tensor-GCN consistently provides superior performance in all metrics. The additional computational cost is a worthwhile trade-off for the enhanced predictive power and reliability that Tensor-GCN brings to the table.

4.3 Ablation Analysis

In this subsection, we conducted a comprehensive ablation analysis to illuminate the pivotal components of our proposed Tensor-GCN model. The primary focus was to assess the contribution of the high-order module Y_2 and the efficacy of employing an indecomposable tensor similarity within this module. To this end, we evaluated three distinct variants: Tensor-GCN-Dec, which shares the same structure with Tensor-GCN, but adopts the decomposable similarity derived from pairwise relationships as given in Eq. (13) for Laplacian tensor $\mathcal{L}_3$ used in Y_2; w/o Y_2, which entirely omits the high-order module, relying solely on Y_0 and Y_1; and the vanilla GCN, representing the classical Graph Convolutional Network [22].

The experimental results, as depicted in Table **??**, unequivocally demonstrate the superiority of the full Tensor-GCN model across all five datasets. Notably, Tensor-GCN achieved an impressive accuracy of 94.070% on the Lung dataset, significantly outperforming Tensor-GCN-Dec (84.244%), w/o Y_2 (77.560%), and vanilla GCN (84.049%). This trend is consistently observed across all datasets.

These findings underscore two critical insights. First, the inclusion of the high-order module Y_2 is instrumental in capturing complex and nuanced relationships within the data, thus improving the discriminative capabilities of the model. The marked decline in performance observed in the w/o Y_2 variant across all datasets substantiates the indispensable role of Y_2 in the Tensor-GCN architecture. Second, the superiority of Tensor-GCN over Tensor-GCN-Dec highlights the advantage of utilizing an indecomposable tensor similarity. This approach allows for a more intricate and holistic representation of feature, facilitating better preservation of essential relational information.

In conclusion, the ablation analysis convincingly validates the strategic incorporation of the high-order module Y_2 and the adoption of indecomposable tensor similarity in Tensor-GCN. These design choices collectively contribute to the model's enhanced performance, affirming their effectiveness in modeling complex and advancing the state-of-the-art in graph convolutional networks.

4.4 Visualization

A visualization experiment is conducted on the Lung dataset for a more intuitive demonstration of Tensor-GCN's performance. The spatial distributions of Tensor-GCN (or GCN, Tensor-GCN-Dec) after t-SNE [40] on the resulting embedding are shown in Figure 2. From Figure 2, we can draw the following conclusions: Firstly, the high clarity of clustering and the presence of clear boundaries between classes in the visualization of Tensor-GCN indicate its superior discriminative ability for samples. Secondly, compared

[2]Hardware used: 16-core Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz, NVIDIA(R) A100-40GB

to other methods, the visualization of Tensor-GCN exhibits better internal consistency and maintains larger separations between different class data points.

Figure 3 illustrates ROC curves of Tensor-GCN on ALLAML, GLI_85, and Lung datasets, which indicate that Tensor-GCN has high discriminative power and error tolerance. Overall, the visualization of Tensor-GCN surpasses baseline methods, further substantiating the superior capabilities and robustness of Tensor-GCN.

5 Conclusion

In this paper, we present Tensor-GCN, a novel tensor graph convolutional network for semi-supervised classification of HDLSS data. It leverages tensor similarity to capture high-order relationships among samples—transcending the limitations of traditional pairwise GCN—and employs a multi-layer architecture that seamlessly integrates multi-order information for deep feature exploration. Experiments on public HDLSS datasets confirm its effectiveness and superiority over state-of-the-art methods.

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*. 21–29.

[2] Miltiadis Allamanis, Marc Brockschmidt, and Mahmoud Khademi. 2018. Learning to Represent Programs with Graphs. In *International Conference on Learning Representations*.

[3] James Atwood and Don Towsley. 2016. Diffusion-convolutional neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 2001–2009.

[4] Hongmin Cai, Fei Qi, Junyu Li, Yu Hu, Bin Hu, Yue Zhang, and Yiu-Ming Cheung. 2024. Uniform tensor clustering by jointly exploring sample affinities of various orders. *IEEE Transactions on Neural Networks and Learning Systems* (2024).

[5] Hongmin Cai, Yu Wang, Fei Qi, Zhuoyao Wang, and Yiu-ming Cheung. 2024. Multiview Tensor Spectral Clustering via Co-regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024), 1–14.

[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* (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*. 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. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*. 3844–3852.

[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*. 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] Xinyi Gao, Wentao Zhang, Junliang Yu, Yingxia Shao, Quoc Viet Hung Nguyen, Bin Cui, and Hongzhi Yin. 2024. Accelerating scalable graph neural network inference with node-adaptive propagation. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. 3042–3055.

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

[15] Jifeng Guo, Zhulin Liu, and CL Philip Chen. 2024. An incremental-self-training-guided semi-supervised broad learning system. *IEEE Transactions on Neural Networks and Learning Systems* (2024).

[16] 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.

[17] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. *arXiv preprint arXiv:1706.02216* (2017).

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

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

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

[21] Diederik P Kingma. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

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

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

[24] 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.

[25] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*. 3538–3545.

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

[27] 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.

[28] Hoang NT and Takanori Maehara. 2019. Revisiting Graph Neural Networks: All We Have is Low-Pass Filters. *arXiv e-prints* (2019), arXiv–1905.

[29] 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* 5 (2020), 2582–2593.

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

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

[32] 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* 3 (2013), 83–98.

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

[34] 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* 1 (2014), 1929–1958.

[35] 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.

[36] Henan Sun, Xunkai Li, Zhengyu Wu, Daohan Su, Rong-Hua Li, and Guoren Wang. 2024. Breaking the Entanglement of Homophily and Heterophily in Semi-supervised Node Classification. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. 2379–2392.

[37] Kaiwen Tan, Weixian Huang, Xiaofeng Liu, Jinlong Hu, and Shoubin Dong. 2021. A hierarchical graph convolution network for representation learning of gene expression data. *IEEE Journal of Biomedical and Health Informatics* (2021), 3219–3229.

[38] 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.

[39] 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.

[40] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing Data using t-SNE. *Journal of Machine Learning Research* 9 (2008), 2579–2605.

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

[42] Yanling Wang, Jing Zhang, Lingxi Zhang, Lixin Liu, Yuxiao Dong, Cuiping Li, Hong Chen, and Hongzhi Yin. 2024. Open-World Semi-Supervised Learning for

Node Classification. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*.

[43] Zichong Wang, Giri Narasimhan, Xin Yao, and Wenbin Zhang. 2023. Mitigating multisource biases in graph neural networks via real counterfactual samples. In *2023 IEEE International Conference on Data Mining (ICDM)*. 638–647.

[44] 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.