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

Anonymous Authors

***Abstract*—Semi-supervised classification can predict labels for all samples only using limited labeled samples. Graph Convolutional Networks (GCNs) 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 the problem of ineffective pairwise similarity caused by the concentration effect, leading to performance falling short of expected levels. To address this issue, we introduced high-order tensor similarity to describe relationships among multiple samples, aiming to extract more valuable information from the graph structure. Building upon this, we proposed the Tensor-based 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.**

***Index Terms*—graph convolutional networks, representation learning, semi-supervised classification, high-order similarity**

I. 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 [1]–[3]. This technology holds great promise in the analysis of complex data in fields such as graphs [4], [5], images [6], text [7], [8], and bioinformatics [9], [10]. In this scenario, the characteristic of semi-supervised classification is to extracting discriminative correlations with limited labeled samples. Fortunately, graph-based methods show its superiority on analyzing the intrinsic properties of complex data, thereby achieving significant success in semi-supervised classification tasks. Meanwhile, Graph Convolutional Networks (GCNs) synergistically harness the salient attributes of the aforementioned methodologies, endowing them with formidable capabilities to extract intricate correlations in data.

GCNs can be broadly categorized into two categories 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 [11] utilizes the powers of a transition matrix to define the neighborhood of nodes. MoNet [12] employs local path operators in the form of Gaussian mixture models to generalize convolution in spatial domain. GAT [13] assigns different weights to neighboring nodes based on their importance, allowing the model to focus on more informative nodes during aggregation. GraphSAGE [14] 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 [4], an efficient layer-wise propagation framework is proposed by simplify the chebyshev polynomials to first-order. GCN 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 [15], [16], ultimately facilitating the propagation of node features within the network structure and forming graph embeddings. This low-pass filtering characteristic [17], [18] have contributed to the remarkable success of GCNs in the past few years.

However, there are certain inherent limitations of GCNs. Firstly, the graph structure in GCN models only utilizes the pairwise relationships of 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 [19]. Additionally, real-world data with high-dimensional characteristics typically face challenges in describing data correlations due to concentration effects [20], [21] and noise interference. Ultimately, this further impacts the ability of GCNs to extract meaningful representations from input data.

Recently, novel methods have been proposed to address the aforementioned shortcomings of GCNs. MixHop [22] is proposed to learn difference operators by repeatly mixing feature representations of neighbors at various distances. BScNet [23] replaces the graph Laplacian with the block Hodge Laplacian to obtain high-order feature representations. GRACES [9] introduce feature selection to deal with HDLSS data. HiGCN [24] employs a hierarchical GCN model to comprehensively consider information from both the feature space and the sample space. Alternatively, Gao and Feng et al. [19], [25] attempt to utilize hypergraphs 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-GCN, which measures relationships between samples by integrating multi-order similarities, providing a new perspective 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:

- A 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 present 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 roubustness 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 II introduces relevant definitions and fundamental concepts necessary for this paper. In Section III we presents the proposed Tensor-GCN model and its implementation. Section IV showcases the experimental results and analysis on the HDLSS dataset. Finally, we summarizes the paper in Section V.

II. PRELIMINARIES

A. Notations

In this paper, we employ the use of calligraphy, uppercase letters, and lowercase letters to symbolize tensors, matrices, and vectors, respectively. For an 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 $\hat{T}_3$ 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 [26], 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:

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

Definition 2.4. ***Khatri-Rao Product*** The Khatri-Rao product [27] 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 [28], 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

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

B. 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 U of the graph Laplacian matrix L based on input data X as basis functions [29]. The normalized Laplacian matrix can be obtained from the similarity matrix and is defined as $L = I - D^{-\frac{1}{2}}SD^{-\frac{1}{2}}$, where S represents similarity matrix

and diagonal degree matrix D denoted as $D_{ii} = \sum_{j=1}^{n} S_{ij}$. The spectral graph convolution can be written by:

$$g \star x = U((U^T g) \odot (U^T x)) = U(g(\Lambda)U^T x), \tag{6}$$

where $g(\Lambda)$ is the counterpart of kernel g in the Fourier domain, which is often viewed as a function of the Laplacian eigenvalues [4], i.e., $g(\Lambda) = diag(g(\lambda_1), g(\lambda_2), \cdots, g(\lambda_n))$.

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 [29]–[31] 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. [30] 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:

$$g \star x \approx \sum_{k=0}^{K} \beta_k T_k(\hat{L})x, \tag{7}$$

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 [32] or further simplify the polynomial filter, such as setting $K = 1$ [4].

C. 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. [33], [34] 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 $\mathcal{T}_3 = [\mathcal{T}_{ijk}] \in \mathbb{R}^{n \times n \times n}$. An intuitive way is to utilize composite similarity based on paired similarity S such that each entry T_{ijk} could be given by:

$$\mathcal{T}_{3_{ijk}} = S_{ij} S_{kj}. \tag{8}$$

By the definition in Eq. (1), the decomposable third-order similarity $\mathcal{T}_3$ defined in Eq. (8) can be unfolded into matrix $\hat{T}_3 \in \mathbb{R}^{n^2 \times n}$. $\hat{T}_3$ can be further written by:

$$\begin{aligned} \hat{T}_3 &= \begin{bmatrix} \mathcal{T}_{3_{111}} & \cdots & \mathcal{T}_{3_{1n1}} \\ \vdots & \ddots & \vdots \\ \mathcal{T}_{3_{n1n}} & \cdots & \mathcal{T}_{3_{nnn}} \end{bmatrix} \\ &= \begin{bmatrix} S_{11}S_{:1} & \cdots & S_{1n}S_{:n} \\ \vdots & \ddots & \vdots \\ S_{n1}S_{:1} & \cdots & S_{nn}S_{:n} \end{bmatrix} \\ &= \begin{bmatrix} S_{:1} \otimes S_{:1} & \cdots & S_{:n} \otimes S_{:n} \end{bmatrix} \\ &= S * S. \end{aligned} \tag{9}$$

Eq. (9) 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 [33] that:

$$\hat{L}_3 = \hat{L} * \hat{L}. \tag{10}$$

Eq. (9) and Eq. (10) 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 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 similarity does, necessitating the design of a novel metrics capable of extracting complementary information beyond the scope of Eq. (9). For the entire elimination of reliance on pairwise relationships, Cai et al. [33] design an indecomposable tensor similarity metric based on their observation of spatial relationships within the sample data. Indecomposable tensor similarity will be introduced in section III-B1. In addition, the reader is referred to Cai et al. [33], [34] for an in-depth discussion of tensor similarity.

III. Tensor-GCN: The Proposed Model

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

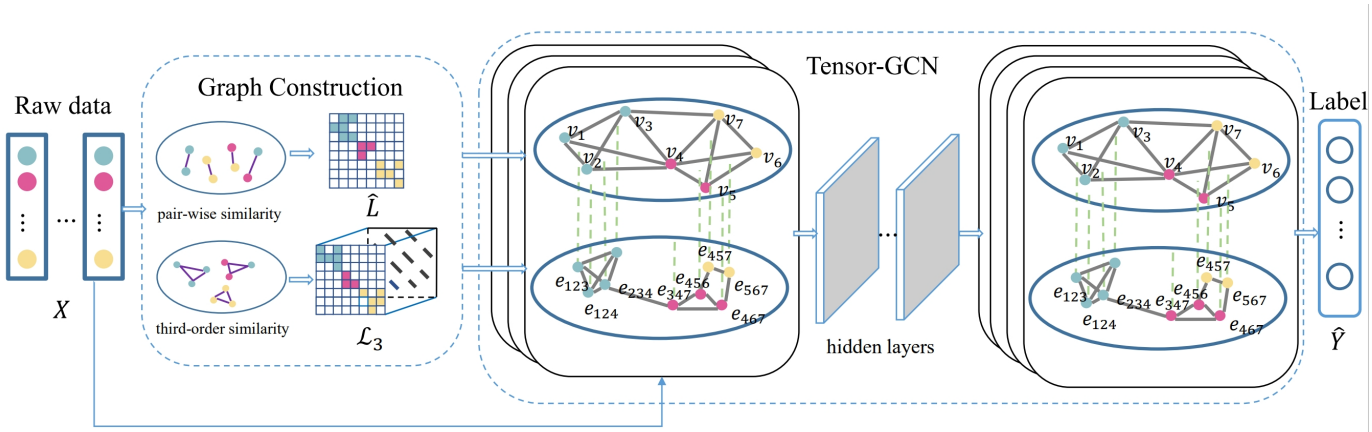


Fig. 1: Tensor-GCN framework. Node feature X is to construct second-order Laplacian matrix and third-order Laplacian tensor, respectively. Subsequently, muilt-order similarities undergo GCN layers for feature extraction.

A. High-order Graph Convolutions

A common intuition to design high-order graph convolutions is to increase the order of the polynomial, which enhance the receptive field of the filter. For example, expanding the Chebyshev polynomial to second-order, expressed as:

$$\begin{aligned} x \star g &\approx \sum_{k=0}^{2} \beta_k T_k(\hat{L}) x \\ &= \beta_0 T_0(\hat{L}) x + \beta_1 T_1(\hat{L}) x + \beta_2 T_2(\hat{L}) x \\ &= \beta_0 x + \beta_1 \hat{L} x + \beta_2 (2\hat{L}^2 - I) x \\ &= [x, \hat{L}x, 2\hat{L}^2 x - x]\theta, \end{aligned} \tag{11}$$

with $\theta = [\beta_0, \beta_1, \beta_2]^T$. Eq. (11) may capture more neighborhood information than the spectral filter. However, a notable limitation of Eq. (11) lies in its reliance on a Laplacian matrix constructed using traditional pairwise similarity measures which disregards high-order neighborhood information. As a result, 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 L^2 term was added earlier [35]. For instance, the performance of ChebNet [32], which utilizes high-order Chebyshev polynomial expansions, has been found to be inferior to that of simpler GCN [4]. 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 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:

$$Y_k(x) = \begin{cases} Ix & \text{, k=0} \\ \hat{L}x & \text{, k=1} \\ (\mathcal{L}_3 \times_3 x^T \times_2 x^T) - x & \text{, k=2} \end{cases}. \tag{12}$$

Here, Y_1 corresponds to the information about samples 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. $\mathcal{L}_3$ 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:

$$g \star_t x = (||_{k=0}^{2} Y_k(x))\theta, \tag{13}$$

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.

B. Implementation

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:

$$S_{ij} = e^{-\frac{d_{ij}^2}{2\sigma^2}}, \tag{14}$$

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 E as σ. We then have graph Laplacian $L = I - D^{-\frac{1}{2}} S D^{-\frac{1}{2}}$.

For high-order similarity, we introduce indecomposable third-order tensor simialrity proposed by Cai et al. [34]. Inspired by the spatial relationships among samples, they 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 tensor $\mathcal{T} \in \mathbb{R}^{n \times n \times n}$ is defined as follows:

$$\mathcal{T}_{3_{ijk}} = 1 - \frac{< (x_i - x_j), (x_k - x_j) >}{d_{ij} d_{jk}}, \tag{15}$$

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 of this definition is: considering arbitrary three samples, x_i, x_j, and x_k, where x_j is treated as the anchor point. Apparently, if x_i and x_k are sufficiently close, regardless of the position of 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 $\hat{T}_3 \in \mathbb{R}^{n^2 \times n}$ 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 $\hat{D}_{3_2} = \sum_i \hat{T}_{3:i}$. Experimental validation conducted by Cai et al. [33], [34] demonstrate that indecomposable similarity can complement pairwise similarity with three-dimensional spatial information, thereby enhancing the expressive power and robustness of the model.

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.(13):

$$Z = \begin{bmatrix} X & \hat{L}X & M - X \end{bmatrix} \Theta, \tag{16}$$

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 $\mathcal{L}_3 \times_2 x^T \times_3 x^T = L_3^T (x \otimes x)$, L_3 is folded from $\mathcal{L}_3$ along mode-3 direction. Meanwhile, based on the definition of the Khatri-Rao product [27], we can express M as:

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

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

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

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:

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

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.

IV. EXPERIMENTS

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

A. Experimental Settings

TABLE I: 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

1) Datasets and Baselines: In this experiment, we validated the performance of Tensor-GCN by applying it to five public biological datasets[1] summarized in Table I. These datasets are all affected by the challenges of HDLSS.

- **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

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

TABLE II: Comparison of the performance of models under HDLSS dataset

Dataset	Metric/Method	MLP	ChebNet	GCN	GAT	HGNN	HiGCN	GRACES	Tensor-GCN-DEC	Tensor-GCN (Ours)
Leukemia	ACC (%)	72.586	82.184	82.759	85.334	87.586	82.826	85.235	87.931	**89.655**
	F-Score	0.7305	0.8464	0.8881	0.8160	0.8702	0.8283	0.8005	0.8530	**0.9927**
	AUC	0.7754	0.8000	0.8722	0.8205	0.8318	0.8596	0.8958	0.8529	**0.9444**
	Recall	0.7259	0.8571	0.8929	0.8205	0.8759	0.8283	0.8005	0.8531	**0.9280**
ALLAML	ACC (%)	70.862	78.736	80.460	82.353	83.103	76.495	82.293	80.172	**83.908**
	F-Score	0.7134	0.7794	0.8251	0.7831	0.8289	0.7650	0.8019	0.8735	**0.8824**
	AUC	0.7551	0.7444	0.7818	0.7647	0.8189	0.7614	0.7923	0.8419	**0.8610**
	Recall	0.7086	0.7857	0.8333	0.7647	0.8310	0.7650	0.8337	0.8809	**0.8869**
GLI_85	ACC (%)	70.588	78.431	80.882	80.393	82.353	73.794	74.020	83.824	**84.559**
	F-Score	0.7059	0.8326	0.8029	0.7539	0.8217	0.7181	0.7379	0.8475	**0.8718**
	AUC	0.6458	0.8030	0.7223	0.7348	0.7986	0.7989	0.8194	0.7799	**0.8520**
	Recall	0.7059	0.8431	0.8235	0.8402	0.8235	0.7181	0.7379	0.8588	**0.8765**
Prostate_GE	ACC (%)	63.171	65.432	72.840	79.838	82.439	61.774	**84.097**	81.482	83.951
	F-Score	0.5308	0.5916	0.7568	0.7977	0.8238	0.6177	0.9231	0.8520	**0.9412**
	AUC	0.6395	0.6318	0.7698	0.7993	0.8244	0.6927	**0.9473**	0.8546	0.9411
	Recall	0.6317	0.6190	0.7647	0.8056	0.8277	0.6177	0.9231	0.8529	**0.9412**
Lung	ACC (%)	67.362	77.914	84.049	91.951	91.779	78.528	80.368	84.244	**94.070**
	F-Score	0.5763	0.7099	0.8256	0.8826	0.8979	0.8101	0.8632	0.8646	**0.9864**
	AUC	0.7410	0.8641	0.9674	0.9987	0.8899	0.8850	0.8722	0.9569	**1.000**
	Recall	0.6736	0.7875	0.9624	0.8292	0.9178	0.7187	0.7924	0.9000	**0.9875**

TABLE III: 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

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.

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 Multilayer Perceptron, is a classic and widely used feedforward neural network model.
- **ChebNet** [32][2] leverages the Chebyshev polynomials to approximate the spectral graph convolution, enabling it to capture graph structures at different scales effectively.
- **GCN** [4][3] simplifies the graph convolution process, applying a localized first-order approximation of spectral graph convolutions to efficiently learn node representations.
- **GAT** [13][4] incorporates attention mechanisms into the graph convolutional framework, allowing for the dynamic weighting of node contributions to achieve more expressive node representations.
- **HGNN** [19][5] utilize hypergraph to mine the local structure between multiple samples, thereby constructing expressive sample similarities.
- **HiGCN** [24][6] is a hierarchical framework that simultaneously utilizes a sparse GCN and a feature-weighted GCN to aggregate neighbour information from both the sample space and the feature space.
- **GRACES** [9][7] is a method specifically tailored for HDLSS data, using GCN to iteratively find a set of optimal features.
- **Tensor-GCN-DEC** shares the same structure with Tensor-GCN, but adopts the decomposable similarity derived from pairwise relationship as given in Eq. (10) for Laplacian tensor $\mathcal{L}_3$.

2) Experimental settings: In this study, we construct kNN graph for features using the method presented in Section III-B1, 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 multi-layer 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 2

[2]https://github.com/mdeff/cnn_graph.git
[3]https://github.com/tkipf/gcn.git
[4]https://github.com/Diego999/pyGAT.git
[5]https://github.com/iMoonLab/HGNN
[6]https://github.com/SCUT-CCNL/HiGCN
[7]https://github.com/canc1993/graces

to 5, and the hidden layer size for each layer is searched in $\{8, 16, 32, 64, 128, 256, 512\}$. Additionally, dropout [36]$\in \{0.4...0.9\}$ and weight decay [37] $\in \{5e{-}5, 5e{-}4, 5e{-}3, 5e{-}2, 5e-1\}$ is introduced to alleviate overfitting. 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.

B. Node Classification

1) Classification performance: The experimental results on public HDLSS datasets are presented in Table II, where bold value indicates the best performing method. As demonstrated in the results, we have the following observations:

1. The proposed Tensor-GCN outperforms most of the baseline methods across all datasets, and maintains a significant margin over other methods in each case, highlighting the effectiveness of our model. Specifically, Tensor-GCN outperforms the runner-up method by 2.07% on the Leukemia dataset, by 0.81% on the ALLAML dataset, by 0.74% on the GLI_85 dataset and by 2.12% on the Lung dataset, its performance on Prostate_GE is also very close to that of GRACES. These results highlight Tensor-GCN's significant advantage in achieving higher accuracy across diverse HDLSS datasets, underscoring its effectiveness and reliability in classification tasks.
2. Tensor-GCN exhibits a significant improvement of over 10% in ACC compared to ChebNet, GCN and HiGCN on the Prostate_GE and Lung datasets. The result futher demonstates that Tensor-GCN can address the limitations of traditional GCN based approaches in effectively capturing and formulating the intricate data correlations.
3. Compared to HGNN, Tensor-GCN demonstrates a clear leadership margin of 2.07%, 0.81%, 2.21%, 1.66% and 2.29% in terms of ACC. This indicates that directly modeling high-order sample similarity can maximize the retention of information.
4. Compared to variant method Tensor-GCN-DEC, which inherently relies on pairwise similarity, the proposed Tensor-GCN demonstrates superior effectiveness across various HDLSS datasets. This validates the efficacy of the indecomposable similarity in exploiting high-dimensional message.
5. In terms of F-score, AUC, and recall, Tensor-GCN consistently outperforms baseline methods, indicating its comprehensive capabilities and excellent robustness. This shows that Tensor-GCN is well-adapted to HDLSS environments and effectively distinguish samples even in the presence of concentration effect and noise interference.

2) Runtime analysis: As a sophisticated graph convolutional network architectures, Tensor-GCN requires additional time to process the complex relationships between samples. Table III provides the time taken to train GCN [4] and Tensor-GCN for 200 epochs on various datasets.[8] 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.

C. Visualization

Visualization experiment is conducted on Lung dataset for a more intuitive demonstration of Tensor-GCN performance. The spatial distributions of Tensor-GCN(or GCN, Tensor-GCN-Dec) after t-SNE [38] 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 to the comparative methods, the visualization ofTensor-GCN exhibits better internal consistency and maintains larger separations between different class data points.

Figure 3 illustrates ROC curves of our method, which demonstrate that Tensor-GCN has high discriminative power and error tolerance. Overall, the visualization of Tensor-GCN surpass baseline methods, which further substantiating the superior capabilities and robustness of Tensor-GCN.

V. CONCLUSION

In this paper, we present a tensor-based graph convolutional network (Tensor-GCN) for addressing the problem of semi-supervised classification for HDLSS data. Firstly, tensor similarity is adopted to capture high-order relationships among multiple samples, overcoming the limitations of traditional pairwise similarity-based GCNs. Subsequently, a multi-layer network architecture that seamlessly integrates low-order and high-order information is designed, enabling deep exploration of sample features. Experiments on several HDLSS datasets well demostrate the effectiveness and superiority of Tensor-GCN over state-of-the-art approaches.

ACKNOWLEDGMENT

REFERENCES

[1] H. Li, S. Tian, Y. Li, Q. Fang, R. Tan, Y. Pan, C. Huang, Y. Xu, and X. Gao, "Modern deep learning in bioinformatics," *Journal of molecular cell biology*, vol. 12, no. 11, pp. 823–827, 2020.

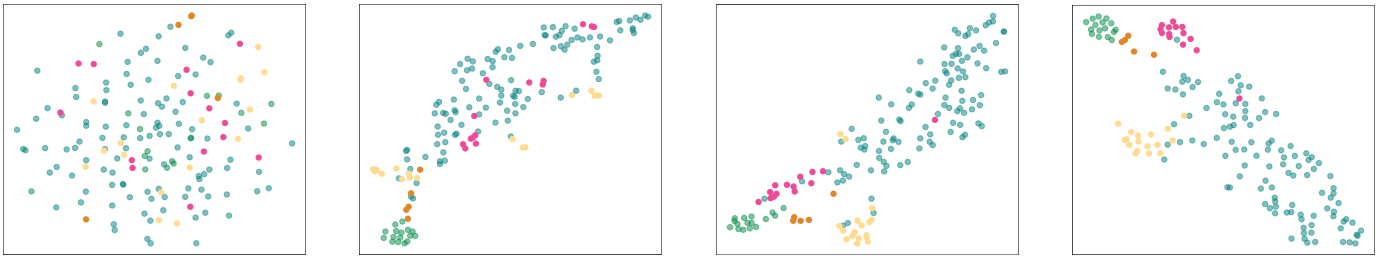
[2] E. Uffelmann, Q. Q. Huang, N. S. Munung, J. De Vries, Y. Okada, A. R. Martin, H. C. Martin, T. Lappalainen, and D. Posthuma, "Genome-wide association studies," *Nature Reviews Methods Primers*, vol. 1, no. 1, p. 59, 2021.

[3] L. Shen, M. J. Er, and Q. Yin, "Classification for high-dimension low-sample size data," *Pattern Recognition*, vol. 130, p. 108828, 2022.

[4] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

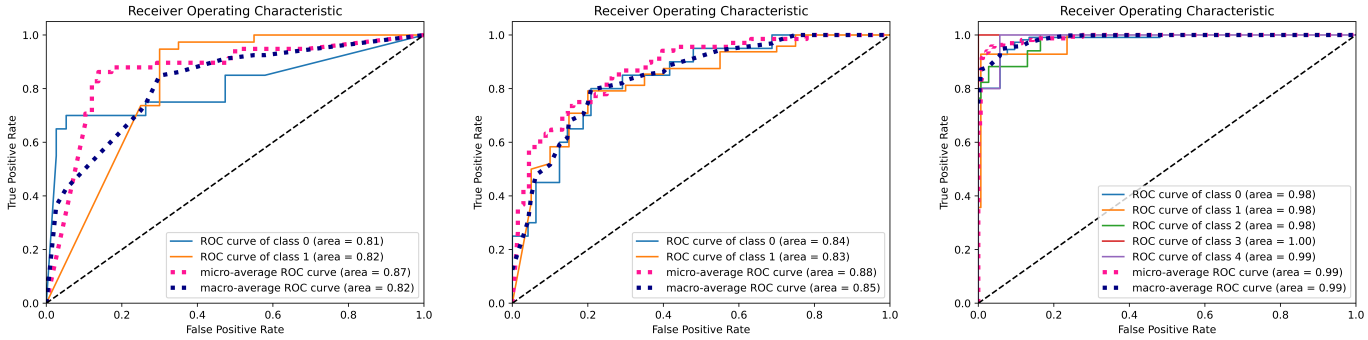
[5] Z. Wang, G. Narasimhan, X. Yao, and W. Zhang, "Mitigating multisource biases in graph neural networks via real counterfactual samples," in *2023 IEEE International Conference on Data Mining (ICDM)*, 2023, pp. 638–647.

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



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

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



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

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

[6] Z.-M. Chen, X.-S. Wei, P. Wang, and Y. Guo, "Multi-label image recognition with graph convolutional networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5177–5186.

[7] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 7370–7377.

[8] M. Allamanis, M. Brockschmidt, and M. Khademi, "Learning to represent programs with graphs," in *International Conference on Learning Representations*, 2018.

[9] C. Chen, S. T. Weiss, and Y.-Y. Liu, "Graph convolutional network-based feature selection for high-dimensional and low-sample size data," *Bioinformatics*, p. btad135, 2023.

[10] F. P. Such, S. Sah, M. A. Dominguez, S. Pillai, C. Zhang, A. Michael, N. D. Cahill, and R. Ptucha, "Robust spatial filtering with graph convolutional neural networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 884–896, 2017.

[11] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 2001–2009.

[12] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 5115–5124.

[13] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[14] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *arXiv preprint arXiv:1706.02216*, 2017.

[15] Q. Li, Z. Han, and X.-M. Wu, "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*, 2018, pp. 3538–3545.

[16] G. Taubin, "A signal processing approach to fair surface design," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 1995, pp. 351–358.

[17] G. Cui, J. Zhou, C. Yang, and Z. Liu, "Adaptive graph encoder for attributed graph embedding," in *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 2020, pp. 976–985.

[18] H. NT and T. Maehara, "Revisiting graph neural networks: All we have is low-pass filters," *arXiv e-prints*, pp. arXiv–1905, 2019.

[19] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," in *Proceedings of the AAAI conference on artificial intelligence*, no. 01, 2019, pp. 3558–3565.

[20] D. François, V. Wertz, and M. Verleysen, "The concentration of fractional distances," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 7, pp. 873–886, 2007.

[21] P. Hall, J. S. Marron, and A. Neeman, "Geometric representation of high dimension, low sample size data," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 67, no. 3, pp. 427–444, 2005.

[22] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. Ver Steeg, and A. Galstyan, "Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," in *international conference on machine learning*, 2019, pp. 21–29.

[23] Y. Chen, Y. R. Gel, and H. V. Poor, "Bscnets: Block simplicial complex neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, no. 6, 2022, pp. 6333–6341.

[24] K. Tan, W. Huang, X. Liu, J. Hu, and S. Dong, "A hierarchical graph convolution network for representation learning of gene expression data," *IEEE Journal of Biomedical and Health Informatics*, pp. 3219–3229, 2021.

[25] Y. Gao, Y. Feng, S. Ji, and R. Ji, "Hgnn+: General hypergraph neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 3, pp. 3181–3199, 2022.

[26] S. Rabanser, O. Shchur, and S. Günnemann, "Introduction to tensor decompositions and their applications in machine learning," *arXiv preprint arXiv:1711.10781*, 2017.

[27] A. K. Smilde, R. Bro, and P. Geladi, *Multi-way analysis: applications in the chemical sciences*, 2005.

[28] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM review*, pp. 455–500, 2009.

[29] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE signal processing magazine*, no. 3, pp. 83–98, 2013.
[30] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, no. 2, pp. 129–150, 2011.
[31] M. He, Z. Wei, H. Xu *et al.*, "Bernnet: Learning arbitrary graph spectral filters via bernstein approximation," *Advances in Neural Information Processing Systems*, pp. 14 239–14 251, 2021.
[32] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 3844–3852.
[33] H. Peng, Y. Hu, J. Chen, H. Wang, Y. Li, and H. Cai, "Integrating tensor similarity to enhance clustering performance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 5, pp. 2582–2593, 2020.
[34] H. Cai, Y. Wang, F. Qi, Z. Wang, and Y.-m. Cheung, "Multiview tensor spectral clustering via co-regularization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–14, 2024.
[35] M. He, Z. Wei, and J.-R. Wen, "Convolutional neural networks on graphs with chebyshev approximation, revisited," *Advances in Neural Information Processing Systems*, pp. 7264–7276, 2022.
[36] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, no. 1, pp. 1929–1958, 2014.
[37] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.
[38] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.