Multi-order Tensor Graph Convolutional Networks for High-Dimensional and Low-Sample Size Data

Anonymous Author(s)

Abstract

Semi-supervised classification aims to accurately predict labels for an entire dataset using only a limited number of labeled samples. Graph Convolutional Networks (GCNs) have demonstrated remarkable success in semi-supervised classification of small-sample datasets by leveraging the intrinsic structure of data graphs. However, when applied to High-Dimensional and Low-Sample Size (HDLSS) data, these methods suffer from overfitting due to the concentration phenomenon. To mitigate this issue, we introduce a high-order tensor similarity measure that captures relationships among multiple samples. Building on this concept, we propose the Multi-order Tensor Graph Convolutional Network (MTensor-GCN), which seamlessly integrates traditional pairwise graph information with higher-order neighborhood information. By adopting a multi-layer MTensor-GCN architecture, we effectively fuse first-order and high-order relational features, resulting in enhanced accuracy and robustness. Extensive experiments on public HDLSS datasets demonstrate that MTensor-GCN can exploit higher-order feature representations and consistently achieves superior predictive performance and stability across HDLSS scenarios.

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. Multi-order 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, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Semi-supervised learning leverages a small set of labeled samples alongside abundant unlabeled data to enhance model performance, especially in scenarios where label acquisition is expensive [15, 22, 28, 34, 37, 39]. This paradigm has demonstrated strong potential across diverse domains, including graph-structured data [13, 21, 40], images [8], text [2, 42], and bioinformatics [6, 33]. A central challenge in semi-supervised classification is to extract discriminative relationships from very few labeled examples. Graph-based methods excel in this regard by capturing the intrinsic topology of complex datasets, and have achieved remarkable success in semi-supervised tasks. In particular, Graph Convolutional Networks (GCNs) synergize graph modeling with deep feature learning, enabling them to uncover rich, high-order correlations.

GCNs are generally divided into two categories based on their theoretical foundations: spatial methods and spectral methods. Spatial approaches define convolution operations directly over each node's local neighborhood. For example, DCNNs [3] model neighborhoods via powers of the transition matrix; MoNet [25] generalizes spatial convolution using Gaussian mixture–based local path operators; GAT [38] employs attention mechanisms to weight neighbors by importance; and GraphSAGE [17] samples representative neighbors for aggregation to update node embeddings.

By contrast, spectral approaches perform convolution in the graph Fourier domain by projecting node features onto the eigenbasis of the Laplacian. The seminal GCN model [21] simplifies Chebyshev polynomial filters to a first-order approximation, yielding an efficient layer-wise propagation scheme. Traditional spectral GCNs aggregate neighbor features in a process equivalent to Laplacian smoothing [23, 36], which acts as a low-pass filter on node signals [9, 26]. This smoothing property has been a key factor in the empirical success of GCNs across a wide range of graph-based learning applications.

Despite their success, Graph Convolutional Networks (GCNs) exhibit several inherent limitations. First, traditional GCN architectures capture only pairwise relationships between samples, overlooking the higher-order interactions that often exist in real-world data. Complex dependencies frequently span more than two nodes, making it difficult for pairwise models to fully characterize these associations [11]. Moreover, high-dimensional datasets are prone to concentration phenomena and noise interference, which further degrade the quality of sample correlations and hinder effective feature learning [12, 16].

To address these shortcomings, various high-order graph methods have been proposed. MixHop [1] learns multi-scale interactions by repeatedly mixing neighbor representations at different distances. BScNet [7] adopts the block Hodge Laplacian to extract higher-order topological features. GRACES [6] introduces feature selection tailored to HDLSS data. HiGCN [35] employs a hierarchical structure to aggregate information across both feature and sample spaces. Hypergraph approaches [11, 14] encode multi-way relationships via hyperedges, enabling richer connectivity modeling. Nevertheless, these methods ultimately rely on approximating sample-to-sample affinities from higher-order constructs, which remains suboptimal when the feature dimensionality m greatly exceeds the sample size n.

Conference acronym 'XX, Woodstock, NY

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

In this work, we propose the Multi-order Tensor Graph Convolutional Network (MTensor-GCN), which directly integrates multi-order similarity measures into the convolutional process. MTensor-GCN naturally models higher-order interactions among samples and seamlessly fuses them with traditional pairwise neighborhoods. Furthermore, we employ a deep tensor convolutional module to learn latent embeddings from the aggregated multi-order information. Our main contributions are summarized as follows:

- MTensor-GCN leverages high-order tensor similarities to model associations among multiple samples, accurately capturing intrinsic relationships that pairwise measures overlook.
- We propose a novel multi-layer tensor graph convolutional framework that fuses tensor similarity with conventional pairwise similarity, enabling more precise and robust feature representation learning and classification.
- Extensive experiments on public HDLSS datasets demonstrate that MTensor-GCN significantly outperforms baseline methods, achieving enhanced node embeddings, superior predictive accuracy, and strong robustness in semi-supervised classification.

The remainder of this paper is organized as follows. Section 2 introduces the key definitions and fundamental concepts. Section 3 presents the proposed MTensor-GCN model and its implementation details. Section 4 reports comprehensive experimental results and analysis on public HDLSS datasets. Finally, Section 5 concludes the paper.

2 Preliminaries

2.1 Notations

We use calligraphic uppercase letters (e.g., $\mathcal{T}$) to denote tensors, bold uppercase letters (e.g., $\boldsymbol{A}$) for matrices, and bold lowercase letters (e.g., $\boldsymbol{v}$) for vectors. For a third-order tensor $\mathcal{T} \in \mathbb{R}^{I\times J\times K}$, the frontal, lateral, and horizontal slices are denoted by

$$\mathcal{T}(:,:,i), \quad \mathcal{T}(:,i,:), \quad \mathcal{T}(i,:,:),$$

respectively, and we abbreviate $\mathcal{T}(:,:,i)$ as $\mathcal{T}^{(i)}$. A tensor can be matricized by unfolding; for example:

Definition 2.1 (Unfolding a third-order tensor). Let $\mathcal{T}_3 \in \mathbb{R}^{N\times N\times N}$. Its mode-3 unfolding produces the matrix $\boldsymbol{T}_3 \in \mathbb{R}^{N^2\times N}$:

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

We briefly recall several matrix and tensor products that will be used later.

Definition 2.2 (Hadamard product). For $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{I\times J}$, their Hadamard (elementwise) product is

$$\boldsymbol{A} \odot \boldsymbol{B} = \left[a_{ij}\, b_{ij}\right]_{i=1,\dots,I}^{j=1,\dots,J} \in \mathbb{R}^{I\times J}.$$

Definition 2.3 (Kronecker product). For $\boldsymbol{A} \in \mathbb{R}^{I\times J}$ and $\boldsymbol{B} \in \mathbb{R}^{K\times L}$, the Kronecker product is

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{bmatrix} a_{11}\boldsymbol{B} & \cdots & a_{1J}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ a_{I1}\boldsymbol{B} & \cdots & a_{IJ}\boldsymbol{B} \end{bmatrix} \in \mathbb{R}^{IK\times JL}.$$

Definition 2.4 (Khatri-Rao product). Given $\boldsymbol{A} \in \mathbb{R}^{I\times K}$ and $\boldsymbol{B} \in \mathbb{R}^{J\times K}$, their column-wise Kronecker product is

$$\boldsymbol{A} * \boldsymbol{B} = \left[a_{:1} \otimes b_{:1},\ a_{:2} \otimes b_{:2},\ \dots,\ a_{:K} \otimes b_{:K}\right] \in \mathbb{R}^{IJ\times K},$$

where $a_{:k}$ and $b_{:k}$ are the kth columns of $\boldsymbol{A}$ and $\boldsymbol{B}$, respectively.

Definition 2.5 (k-mode product). For an order-m tensor $\mathcal{T} \in \mathbb{R}^{I_1\times\cdots\times I_m}$ and a matrix $\boldsymbol{V} \in \mathbb{R}^{P\times I_k}$, the k-mode product $\mathcal{T} \times_k \boldsymbol{V} \in \mathbb{R}^{I_1\times\cdots\times I_{k-1}\times P\times I_{k+1}\times\cdots\times I_m}$ is defined elementwise by

$$\left(\mathcal{T} \times_k \boldsymbol{V}\right)_{i_1\dots i_{k-1}\, j\, i_{k+1}\dots i_m} = \sum_{i_k=1}^{I_k} \mathcal{T}_{i_1\dots i_k\dots i_m}\, V_{j i_k}.$$

2.2 Revisiting Spectral Graph Convolution

Spectral graph convolution defines filtering operations in the graph Fourier domain by projecting signals onto the eigenbasis of the graph Laplacian [29]. Given a similarity matrix $\boldsymbol{S}$ and its degree matrix $\boldsymbol{D}$ with $D_{ii} = \sum_{j=1}^{n} S_{ij}$, the normalized Laplacian is

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

Let $\boldsymbol{U} \in \mathbb{R}^{n\times n}$ be the matrix of eigenvectors of $\hat{\boldsymbol{L}}$ and $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \dots, \lambda_n)$ the corresponding eigenvalues. For a graph signal $\boldsymbol{x} \in \mathbb{R}^n$ and spectral filter g, the convolution is expressed as

$$g \star \boldsymbol{x} = \boldsymbol{U}\left(g(\boldsymbol{\Lambda})\,\boldsymbol{U}^\top \boldsymbol{x}\right), \tag{1}$$

where $g(\boldsymbol{\Lambda}) = \text{diag}\big(g(\lambda_1), \dots, g(\lambda_n)\big)$ [21].

Directly parameterizing $g(\boldsymbol{\Lambda})$ entails n free parameters, which is computationally expensive for large graphs. To alleviate this, many methods approximate $g(\boldsymbol{\Lambda})$ with a Kth-order polynomial [18, 19, 29]. In particular, Chebyshev polynomial approximation yields

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

where $\widetilde{\boldsymbol{\Lambda}} = \frac{2}{\lambda_{\max}}\boldsymbol{\Lambda} - \boldsymbol{I}$, $\{\beta_k\}_{k=0}^{K}$ are learnable coefficients, and the Chebyshev polynomials satisfy

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_k(x) = 2x\,T_{k-1}(x) - T_{k-2}(x) \quad (k \geq 2).$$

Substituting into (1) and using $\lambda_{\max} \approx 2$ [21] leads to the efficient form

$$g \star \boldsymbol{x} \approx \sum_{k=0}^{K} \beta_k\, T_k\big(\hat{\boldsymbol{L}}\big)\,\boldsymbol{x}. \tag{2}$$

This approximation reduces the number of parameters to $K+1$ and ensures that the filter is K-localized, relying only on nodes within K hops of each target node. Choosing K balances computational complexity and expressive power, with $K = 1$ recovering the classical first-order GCN layer [10, 21].

2.3 Tensor Spectral Analysis

In many learning tasks—such as clustering, classification, and recommendation—the choice of similarity measure critically influences performance. Traditional approaches quantify relationships between sample pairs, using metrics like Euclidean distance, Gaussian kernels, cosine similarity, Jaccard index, or Pearson correlation, depending on data characteristics. However, reducing multi-way interactions to pairwise terms discards rich, higher-order information.

Recent work by Cai et al. [4, 5, 27] advocates representing intrinsic sample correlations via high-order tensor similarities. For example, a third-order similarity tensor $\mathcal{T}_3 = [\mathcal{T}_{ijk}] \in \mathbb{R}^{N\times N\times N}$ can encode the joint affinity among triplets of samples. A simple decomposable construction uses the pairwise similarity matrix $\boldsymbol{S}$, defining

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

By the mode-3 unfolding (Def. 2.1), this tensor admits the matrix form

$$\boldsymbol{T}_3 = \left[\mathcal{T}_3^{(1)}; \mathcal{T}_3^{(2)}; \ldots; \mathcal{T}_3^{(N)}\right] = [\, S_{:1} \otimes S_{:1} \;\cdots\; S_{:N} \otimes S_{:N}\,] = \boldsymbol{S} * \boldsymbol{S},$$

where "$*$" denotes the Khatri–Rao product [30]. Normalizing $\boldsymbol{T}_3$ by its row- and column-sums yields a third-order Laplacian-like tensor $\mathcal{L}_3$. Moreover, if $\hat{\boldsymbol{L}} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{S}\,\boldsymbol{D}^{-1/2}$ is the normalized graph Laplacian of the k-NN graph on $\boldsymbol{S}$, one can show [27]

$$\hat{\boldsymbol{L}}_3 = \hat{\boldsymbol{L}} * \hat{\boldsymbol{L}}, \tag{4}$$

revealing that decomposable tensor similarity is fully determined by pairwise relationships. Consequently, it inherits the same limitations—especially in HDLSS regimes where $m \gg n$. To overcome this, [27] introduce an *indecomposable* tensor similarity metric that captures truly higher-order structure; this construction will be detailed in Sec. **??**. For a comprehensive treatment of tensor-based similarities, we refer the reader to [4, 27].

3 MTensor-GCN: The Proposed Model

In this section, we present the MTensor-GCN model, whose central innovation lies in jointly leveraging low-order and high-order relational information within the graph convolution operation. Unlike conventional GCNs that rely exclusively on an adjacency or Laplacian matrix to encode pairwise affinities, MTensor-GCN constructs a high-order tensor similarity to directly capture complex multi-node interactions, thereby improving robustness and discrimination in high-dimensional regimes. As illustrated in Figure 1, given the node feature matrix $\boldsymbol{X}$, we first derive both a second-order normalized Laplacian $\hat{\boldsymbol{L}}$ and a third-order Laplacian tensor $\mathcal{L}_3$. These multi-order affinity structures are then processed in parallel by dedicated graph convolutional layers to produce enriched node representations for classification.

3.1 Tensor Graph Convolutional Framework

Building on the polynomial filter perspective, recall the Kth-order Chebyshev expansion in Eq. (**??**). Specializing to $K = 3$ yields

$$\begin{aligned} g \star \boldsymbol{x} &= \sum_{k=0}^{3} \beta_k T_k(\hat{\boldsymbol{L}}) \\ &= \beta_0 \boldsymbol{x} + \beta_1 \boldsymbol{L}\boldsymbol{x} + \beta_2(2\hat{\boldsymbol{L}}^2\boldsymbol{x} - \boldsymbol{x}) + \beta_3(4\hat{\boldsymbol{L}}^3\boldsymbol{x} - 3\hat{\boldsymbol{L}}\boldsymbol{x}), \end{aligned} \tag{5}$$

Reordering by powers of $\hat{\boldsymbol{L}}$ and enforcing a single trainable parameter θ via

$$\begin{cases} \beta_0 - \beta_2 = \theta, \\ \beta_1 - 3\beta_3 = \theta, \\ 2\beta_2 = \theta, \\ 4\beta_3 = \theta, \end{cases} \quad\Longrightarrow\quad \{\beta_0, \beta_1, \beta_2, \beta_3\} = \{\tfrac{3}{2}\theta, \tfrac{7}{4}\theta, \tfrac{1}{2}\theta, \tfrac{1}{4}\theta\},$$

yields the compact form

$$g \star \boldsymbol{x} = \left(\boldsymbol{x} + \hat{\boldsymbol{L}}\boldsymbol{x} + \hat{\boldsymbol{L}}^2\boldsymbol{x} + \hat{\boldsymbol{L}}^3\boldsymbol{x}\right)\theta. \tag{6}$$

For a feature matrix $\boldsymbol{X} \in \mathbb{R}^{N\times C}$ and weight tensor $\boldsymbol{\Theta} \in \mathbb{R}^{4C\times F}$, we explicitly concatenate the four basis responses:

$$\left[\boldsymbol{X},\ \hat{\boldsymbol{L}}\boldsymbol{X},\ \hat{\boldsymbol{L}}^2\boldsymbol{X},\ \hat{\boldsymbol{L}}^3\boldsymbol{X}\right]\boldsymbol{\Theta},$$

allowing the network to learn distinct contributions from each neighborhood order while remaining compatible with standard fully-connected layers. Crucially, this framework isolates the convolutional guidance from pairwise-only Laplacian structure, mitigating overfitting risks in HDLSS settings by incorporating inherently richer, multi-order information.

To overcome the limitations of the current framework, we introduce high-order similarity tensors into the graph convolutional architecture, thereby enhancing the model with additional informative cues. First, we define the mapping

$$\mathcal{F} : \mathbb{R}^{N\times C\times C_1\times\cdots\times C_q} \longrightarrow \mathbb{R}^{N\times C},$$

which sums the input tensor over its 3rd through $(q+2)$-th modes:

$$\mathcal{F}(\boldsymbol{\mathcal{T}})_{i,j} = \sum_{k_1=1}^{C_1}\sum_{k_2=1}^{C_2}\cdots\sum_{k_q=1}^{C_q} \boldsymbol{\mathcal{T}}_{i,j,k_1,\ldots,k_q}. \tag{7}$$

Assuming a parameter matrix $\boldsymbol{\Theta} \in \mathbb{R}^{MC\times F}$, the output of a single MTensor-GCN layer is defined as

$$\boldsymbol{Z}(\boldsymbol{X}, \boldsymbol{\Theta}) = \|_{k=1}^{M} Y_k(\boldsymbol{X})\, \boldsymbol{\Theta}. \tag{8}$$

where "||" denotes concatenation across orders, and each $Y_k(\boldsymbol{X})$ is given by

$$Y_k(\boldsymbol{X}) = \begin{cases} \boldsymbol{X}, & k = 1, \\ \hat{\boldsymbol{L}}\,\boldsymbol{X}, & k = 2, \\ \mathcal{F}\left(\mathcal{L}_3 \times_3 \boldsymbol{X}^\top \times_2 \boldsymbol{X}^\top\right), & k = 3, \\ \mathcal{F}\left(\mathcal{L}_4 \times_4 \boldsymbol{X}^\top \times_3 \boldsymbol{X}^\top \times_2 \boldsymbol{X}^\top\right), & k = 4, \\ \vdots & \\ \mathcal{F}\left(\mathcal{L}_M \times_M \boldsymbol{X}^\top \times_{M-1} \cdots \times_2 \boldsymbol{X}^\top\right), & k = M. \end{cases} \tag{9}$$

Equations (8) and (9) thus define a tensor graph convolution network capable of supporting similarity tensors of arbitrary order. In the following, we illustrate the full hierarchical MTensor-GCN model by specifically including third- and fourth-order tensor similarities.

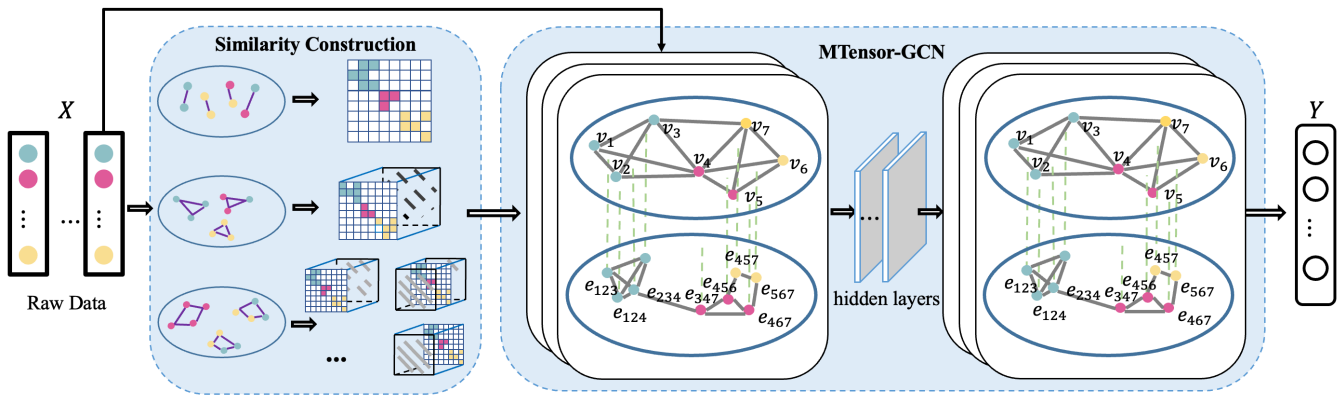


Figure 1: MTensor-GCN framework. Node feature X is to construct second-order Laplacian matrix and third-order Laplacian tensor, respectively. Subsequently, multi-order similarities undergo GCN layers for feature extraction.

3.2 Implementation

3.2.1 Graph Construction. The MTensor-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:

$$S_{ij} = \exp\left(-\frac{d_{ij}^2}{2\sigma^2}\right), \tag{10}$$

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 $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{S}\boldsymbol{D}^{-\frac{1}{2}}$.

3.2.2 Tensor Similarities Construction. 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 $\boldsymbol{\mathcal{T}}_3 = [\mathcal{T}_{3(ijk)}] \in \mathbb{R}^{N\times N\times N}$ is defined as follows:

$$\mathcal{T}_{3_{ijk}} = 1 - \frac{< (\boldsymbol{x}_i - \boldsymbol{x}_j),\ (\boldsymbol{x}_k - \boldsymbol{x}_j) >}{d_{ij}d_{jk} + \epsilon}, \tag{11}$$

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, 27] demonstrate that indecomposable similarity can complement pairwise similarity with three-dimensional spatial information, thereby enhancing the expressive power and robustness of the model.

To quantify fourth-order relations among four samples, the IPS2 model proposed in [27] employs a ratio-based quaternary similarity measure. This metric effectively mitigates the hubness phenomenon in high-dimensional spaces, thereby significantly enhancing clustering performance on data with high dimensionality and low sample size. Specifically, for any four nodes $\boldsymbol{h}_i$, $\boldsymbol{h}_j$, $\boldsymbol{h}_k$, and $\boldsymbol{h}_l$, the fourth-order similarity $\mathcal{T}_{4_{ijkl}}$ is defined as

$$\mathcal{T}_{4_{ijkl}} = \exp\left(-\frac{d_{ij} + d_{kl} + d_{il} + d_{jk}}{d_{ik} + d_{jl} + \epsilon}\right), \tag{12}$$

where d_{ab} denotes the Euclidean distance between nodes h_a and h_b, and ϵ is a small constant to avoid division by zero. In this formulation, the numerator

$$d_{ij} + d_{kl} + d_{il} + d_{jk}$$

represents the inter-pair distance—measuring the distances between different node pairs—while the denominator

$$d_{ik} + d_{jl}$$

captures the intra-pair distance—measuring the distances within the same node pair. The geometric interpretation of this fourth-order similarity is illustrated in Figure 2.

3.2.3 Layer-wise Model for Node Classification. In semi-supervised classification, only a small subset of nodes is labeled; the model leverages these labeled instances together with the graph structure to infer labels for all vertices. Following the procedure in Section **??**, we first construct the graph to obtain the pairwise similarity matrix $\boldsymbol{S}$ and the third-order tensor similarity $\boldsymbol{\mathcal{T}}_3$. We then compute the

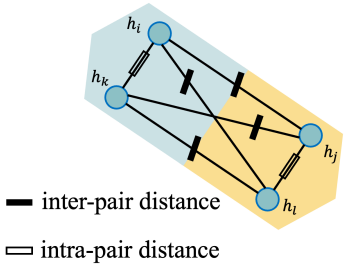


Figure 2: IPS2

normalized Laplacian matrices $\hat{\boldsymbol{L}}$ and $\hat{\boldsymbol{L}}_3$, and feed them, along with the feature matrix $\boldsymbol{X}$, into our multi-layer MTensor-GCN. The layer-wise propagation is given by

$$\boldsymbol{X}^{(l+1)} = \sigma\Big(\big[Y_1(\boldsymbol{X}^{(l)}),\ Y_2(\boldsymbol{X}^{(l)}),\ Y_3(\boldsymbol{X}^{(l)}),\ Y_4(\boldsymbol{X}^{(l)})\big]\,\boldsymbol{W}^{(l)}\Big), \tag{13}$$

where $\boldsymbol{X}^{(l)} \in \mathbb{R}^{N\times C}$ denotes the graph signal at layer l (with $\boldsymbol{X}^{(0)} = \boldsymbol{X}$), $\boldsymbol{W}^{(l)} \in \mathbb{R}^{4C\times F}$ is the trainable weight matrix, and $\sigma(\cdot)$ applies a row-wise softmax activation. During training, the parameters $\{\boldsymbol{W}^{(l)}\}$ are learned via backpropagation by minimizing the cross-entropy loss over the labeled node set $\mathcal{Y}_\ell$:

$$\mathcal{L} = -\sum_{i\in\mathcal{Y}_\ell}\sum_{j=1}^{F} Y_{ij}\,\ln Z_{ij}\,, \tag{14}$$

where Y_{ij} is the one-hot encoded ground-truth label and Z_{ij} is the predicted probability for node i and class j. By integrating representations from multiple tensor orders and performing meticulous fusion, our framework achieves more accurate and robust classification—even under high-dimensional, low-sample-size (HDLSS) conditions. The overall workflow of MTensor-GCN is illustrated in Algorithm 1.

Algorithm 1 MTensor-GCN Workflow

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

Compute the normalized second-order Laplacian matrix $\hat{\boldsymbol{L}}$.
Compute the folded normalized third-order Laplacian tensor $\boldsymbol{\mathcal{L}}_3$.
for $t = 0, 1, \ldots, T-1$ **do**
  for $l = 0, 1, \ldots, N-1$ **do**
    $Y_1^{(l)} \leftarrow \boldsymbol{X}^{(l)}$.
    $Y_2^{(l)} \leftarrow \hat{\boldsymbol{L}}\,\boldsymbol{X}^{(l)}$.
    $Y_3^{(l)} \leftarrow \mathcal{F}\big(\boldsymbol{\mathcal{L}}_3 \times_3 \boldsymbol{X}^{(l)^\top} \times_2 \boldsymbol{X}^{(l)^\top}\big)$.
    $Y_4^{(l)} \leftarrow \mathcal{F}\big(\boldsymbol{\mathcal{L}}_4 \times_4 \boldsymbol{X}^{(l)^\top} \times_3 \boldsymbol{X}^{(l)^\top} \times_2 \boldsymbol{X}^{(l)^\top}\big)$.
    Concatenate features and update layer output:
    $\boldsymbol{Z}^{(l)} \leftarrow \sigma\Big(\big[Y_1^{(l)},\ Y_2^{(l)},\ Y_3^{(l)},\ Y_4^{(l)}\big]\boldsymbol{W}^{(l)}\Big)$
    Compute cross-entropy loss over $\boldsymbol{Y}_L$ at layer l:
    $\mathcal{L} = -\sum_{i\in\mathcal{Y}_\ell}\sum_{j=1}^{F} Y_{ij}\ln Z_{ij}^{(l)}$.
    Update all layer weights $\boldsymbol{W}^{(l)}$ via backpropagation using learning rate η.
  end for
end for
For each node i: $\hat{\boldsymbol{y}}_i \leftarrow \arg\max_j \boldsymbol{Z}_{ij}$.
return Predicted labels $\hat{\boldsymbol{Y}}_{\text{pred}}$.

4 Experiments

In this section, we demonstrate the superiority of the proposed MTensor-GCN by comparing its performance with several state-of-the-art methods. The comparison highlights the effectiveness of MTensor-GCN in addressing the HDLSS problem and its ability to achieve superior results across multiple datasets.

4.1 Experimental Setup

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

4.1.1 Datasets. We evaluate the proposed MTensor-GCN on five widely used public biological datasets[1], summarised in Table 1. All datasets exhibit the high-dimension–low-sample-size (HDLSS) characteristic. A brief description of each dataset is given below:

- **Leukemia** : Gene-expression profiles collected from 72 subjects diagnosed with acute lymphoblastic leukaemia (ALL) or healthy controls, each represented by 7 070 features.
- **ALLAML** : Micro-array data from 72 leukaemia patients categorised into acute lymphoblastic leukaemia (ALL) and acute myeloid leukaemia (AML); every sample contains 7 129 gene-expression features.
- **GLI-85** : Transcriptional data obtained from 85 tumour biopsies of diffuse intrinsic pontine glioma (DIPG) in 74 patients, each characterised by 22 283 features; the tumours are grouped into two diagnostic categories.
- **Prostate_GE** : Prostate-cancer gene-expression dataset comprising 102 samples, each with 5 966 features.
- **Lung** : Gene-expression profiles of 203 lung-cancer patients spanning five histological subtypes.

4.1.2 Baselines. We compare MTensor-GCN against state-of-the-art graph convolutional networks with publicly available implementations:

- **ChebNet** [10] : Approximates spectral graph convolution with Chebyshev polynomials, capturing multi-scale graph structure efficiently.

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

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

Dataset	Metric/Method	ChebNet	GCN	GAT	HGNN	HiGCN	GRACES	RT-GCN	CHGNN	MTensor-GCN
Leukemia	ACC (%)	82.184	82.759	85.334	87.586	82.826	85.235	75.864	86.211	**91.017**
	F-Score	0.8464	0.8881	0.8160	0.8702	0.8283	0.8005	0.7181	0.7333	**0.9628**
	AUC	0.8000	0.8722	0.8205	0.8318	0.8596	0.8958	0.6958	0.8042	**0.9639**
	Recall	0.8571	0.8929	0.8205	0.8759	0.8283	0.8005	0.7181	0.6111	**0.9410**
ALLAML	ACC (%)	78.736	80.460	82.353	83.103	76.495	82.293	72.413	82.765	**87.931**
	F-Score	0.7794	0.8251	0.7831	0.8289	0.7650	0.8019	0.6869	0.6429	**0.8785**
	AUC	0.7444	0.7818	0.7647	0.8189	0.7614	0.7923	0.7014	0.7583	**0.8644**
	Recall	0.7857	0.8333	0.7647	0.8310	0.7650	0.8337	0.6931	0.5000	**0.8973**
GLI_85	ACC (%)	78.431	80.882	80.393	82.353	73.794	74.020	82.351	82.351	**86.765**
	F-Score	0.8326	0.8029	0.7539	0.8217	0.7181	0.7379	0.7809	0.8605	**0.8646**
	AUC	0.8030	0.7223	0.7348	0.7986	0.7989	0.8194	0.8500	**0.8677**	0.8252
	Recall	0.8431	0.8235	0.8402	0.8235	0.7181	0.7379	0.7729	0.7708	**0.8676**
Prostate_GE	ACC (%)	65.432	72.840	79.838	82.439	61.774	84.097	75.610	76.834	**87.805**
	F-Score	0.5916	0.7568	0.7977	0.8238	0.6177	0.9231	0.7524	0.7816	**0.9630**
	AUC	0.6318	0.7698	0.7993	0.8244	0.6927	0.9473	0.7620	0.8042	**0.9653**
	Recall	0.6190	0.7647	0.8056	0.8277	0.6177	0.9231	0.7561	0.8095	**0.9653**
Lung	ACC (%)	77.914	84.049	91.951	91.779	78.528	80.368	75.463	93.250	**95.706**
	F-Score	0.7099	0.8256	0.8826	0.8979	0.8101	0.8632	0.3350	0.7523	**0.9932**
	AUC	0.8641	0.9674	0.9123	0.8899	0.8850	0.8722	0.8707	0.9885	**0.9958**
	Recall	0.7875	0.9624	0.8292	0.9178	0.7187	0.7924	0.3760	0.7767	**0.9940**

- **GCN** [21] : Employs a first-order spectral approximation to learn node embeddings with linear complexity.
- **GAT** [38] : Introduces self-attention into graph convolution, enabling adaptive weighting of neighbouring nodes.
- **HGNN** [11] : Utilises hypergraph structure to exploit high-order sample relations and build expressive sample similarities.
- **HiGCN** [35] : A hierarchical framework that aggregates information through a sparse GCN in sample space and a feature-weighted GCN in feature space.
- **GRACES** [6] : Dynamically constructs similarity graphs to iteratively select informative features, combining multiple dropout strategies, Gaussian noise, and an ANOVA F-test to mitigate overfitting on HDLSS data.
- **RT-GCN** [41] : Stacks multiple augmented graph views into a third-order tensor and performs truncated T-SVD to enhance robustness against noise and adversarial perturbations.
- **CHGNN** [31] : An end-to-end semi-supervised contrastive hypergraph network that (i) generates InfoMin-compliant hyperedge perturbations and (ii) employs a homogeneity-aware encoder to reinforce semantically consistent hyperedges.

4.1.3 Evaluation Metrics. To provide a comprehensive assessment on HDLSS tasks we report *Accuracy* (**ACC**), *F-score*, *Area Under the ROC Curve* (**AUC**), and *Recall*. The F-score balances precision and recall, emphasising performance on imbalanced classes. AUC measures the discriminative capacity across all decision thresholds—computed with a one-vs-rest strategy for multi-class datasets. Recall indicates the proportion of true positives correctly identified. Together, these four metrics offer a balanced and rigorous evaluation.

4.1.4 Experimental Settings. Because none of the datasets are equipped with an intrinsic graph structure, we follow Section **??** and build a k-nearest-neighbour graph (k=5) in the *feature* space. All samples are randomly divided into labelled and unlabelled subsets with a 20:80 ratio; only 20 % of the samples are used as labelled data. All baselines employ the hyper-parameters recommended in their original publications, further fine-tuned via grid search for optimal performance. For our method, we implement a multi-layer MTensor-GCN trained with the Adam optimiser [20] (learning rate 0.005). Key hyper-parameters are selected by grid search:

- **Number of layers:** $L \in \{2, 3, 4, 5\}$;
- **Hidden size per layer:** $d \in \{16, 32, 64, 128, 256\}$;
- **Dropout** [32]: $p \in \{0.5, 0.6, 0.8\}$;
- **Weight decay** [24]: $\lambda \in \{5{\times}10^{-4}, 5{\times}10^{-3}, 5{\times}10^{-2}\}$.

Each method is executed ten times with identical data splits, and the averaged results are reported.

4.2 Performance Comparison

4.2.1 Node Classification. Table 2 summarises the node-classification results on the five HDLSS datasets; the best scores are highlighted in **bold**. From the table we note the following observations:

1. MTensor-GCN achieves the best accuracy (ACC) on all five HDLSS datasets. Specifically, it records 91.017% on Leukemia, 87.931% on ALLAML, 86.765% on GLI_85, 87.805% on Prostate_GE, and 95.706% on Lung. These values exceed the second–best competitors by +3.43%, +4.83%, +4.41%, +3.71%, and +2.46% absolute, respectively, highlighting a clear and consistent margin.
2. Compared with classical graph-convolution methods—GCN, ChebNet, and GAT—MTensor-GCN delivers sizeable gains across the board. For instance, on Leukemia it lifts ACC from GCN's

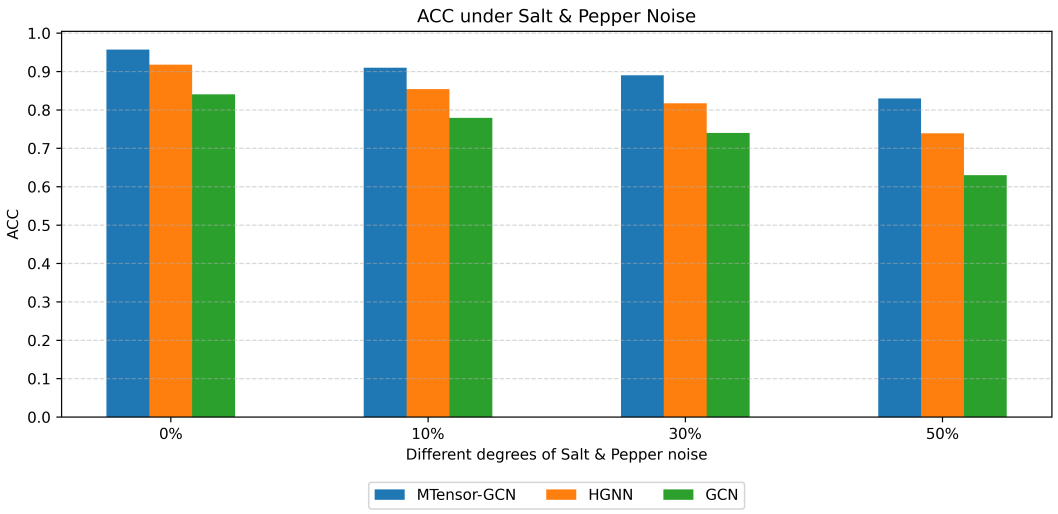


Figure 3: Classification ACC of MTensor-GCN, HGNN, and GCN on the Lung dataset under varying levels of salt-and-pepper noise.

82.759%, ChebNet's 82.184%, and GAT's 85.334% to 91.017% (+8.26, +8.83, and +5.68 percentage-point improvements). On the more challenging Prostate_GE dataset, it reaches 87.805%, surpassing GCN, ChebNet, and GAT by +14.97, +22.37, and +7.97 points, respectively. This evidences MTensor-GCN's stronger capacity to capture fine-grained correlations that traditional pairwise convolutions miss.

3. MTensor-GCN also outperforms advanced high-order models such as HGNN, HiGCN, GRACES, and CHGNN. Taking Lung as an example, it improves upon HGNN (91.779%), HiGCN (78.528%), and GRACES (80.368%) by +3.93, +17.18, and +15.34 percentage points. On Prostate_GE, it exceeds GRACES—the strongest competitor on that dataset—by +3.71 points, while widening the gap to HiGCN by a striking +26.03 points. These results confirm MTensor-GCN's effectiveness in exploiting indecomposable high-order similarities.
4. Beyond accuracy, MTensor-GCN consistently secures the highest F-score, Recall, and AUC on four of the five datasets, and ranks a close second in AUC on GLI_85. For example, on Leukemia it achieves an F-score of 0.9628, AUC of 0.9639, and Recall of 0.9410, substantially outperforming all alternatives. Such balanced precision–recall trade-offs demonstrate the model's reliability and generalisation, rather than isolated gains on a single metric.

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

4.3 Robustness Analysis on Noisy Dataset

To evaluate the robustness of MTensor-GCN under different noise levels, a noise-attack experiment was designed, using the Lung dataset as the benchmark. The experiment involved comparing the classification performance of three methods: MTensor-GCN, HGNN, and GCN, under varying degrees of salt-and-pepper noise. The results are presented in the bar plot below, which shows the ACC of each method at different noise levels (0%, 10%, 30%, and 50%). The results are shown in Figure 3.

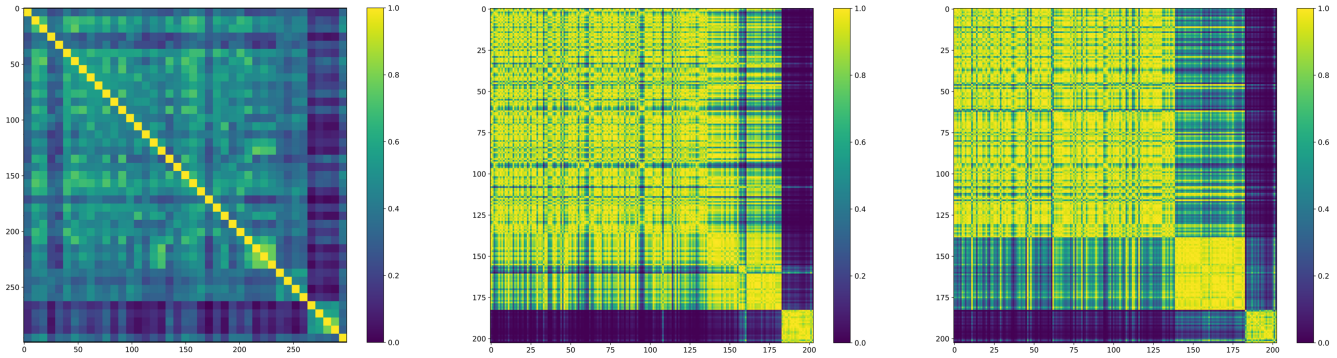
From the graph, it is evident that MTensor-GCN demonstrates a distinct advantage in terms of robustness. Firstly, its performance drops the least as noise increases. For example, even at 50% noise, MTensor-GCN retains a higher accuracy compared to both HGNN and GCN. This highlights its resilience to noise and suggests that its higher-order modeling capabilities contribute to better noise handling.

Secondly, MTensor-GCN consistently maintains an accuracy above 80%, which indicates that its classification ability remains reliable across all noise levels. In contrast, while HGNN and GCN also perform reasonably well, their accuracies fall below 80% as the noise level increases, indicating a significant degradation in their classification performance.

These findings underscore the robustness of MTensor-GCN, particularly in high-noise scenarios, where its performance remains stable and reliable compared to the other methods. The combination of high-order tensor representations and the ability to effectively manage noisy data gives MTensor-GCN a clear advantage in environments with significant data corruption.

4.4 Similarity Heatmap Analysis

To rigorously evaluate the ability of MTensor-GCN to uncover intrinsic relationships through an indecomposable tensor similarity, we visualize similarity heatmaps on the Lung dataset under three settings: (a) the raw feature space, (b) features transformed by a vanilla GCN, and (c) features produced by MTensor-GCN. The resulting heatmaps are shown in Figure 4.



(a) (b) (c)

Figure 4: Similarity heatmaps for (a) raw data, (b) GCN, and (c) MTensor-GCN on the Lung dataset.

In the heatmap of the raw data, the similarity distribution across classes is diffuse and class boundaries are hardly discernible, revealing the combined effects of noise and partial overlap in the original high-dimensional feature space. After transformation by the vanilla GCN, the global similarity structure becomes smoother and several boundaries are sharpened. Nevertheless, because the pairwise similarity of GCN neglects higher-order interactions, intra-class compactness remains inadequate and the transition regions between classes are still ambiguous.

By contrast, MTensor-GCN incorporates an indecomposable tensor similarity that fully exploits higher-order feature interactions, yielding markedly improved class separability. The block patterns in its heatmap are well defined, demonstrating stronger intra-class cohesion, clearer inter-class demarcation, and enhanced robustness to noise. These observations indicate that the proposed tensor similarity more precisely captures high-order relationships among samples and consequently delivers superior visual clarity and stability in HDLSS scenarios.

5 Conclusion

In this work, we propose MTensor-GCN, a tensor-based graph convolutional network tailored for semi-supervised classification of high-dimension and low-sample size data. By constructing tensorized similarity graphs, MTensor-GCN captures high-order relationships among samples—surpassing the pairwise modelling limitations of traditional GCNs—and employs a multi-layer architecture that fuses information from multiple tensor orders for deep feature exploration. Extensive experiments on public HDLSS benchmarks demonstrate the effectiveness of MTensor-GCN and its consistent 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, Hongteng Xu, et al. 2021. Bernnet: Learning arbitrary graph spectral filters via bernstein approximation. *Advances in Neural Information Processing Systems* (2021), 14239–14251.

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

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

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

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

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

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

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

[27] 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.

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

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

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

[31] Yumeng Song, Yu Gu, Tianyi Li, Jianzhong Qi, Zhenghao Liu, Christian S Jensen, and Ge Yu. 2024. CHGNN: a semi-supervised contrastive hypergraph learning network. *IEEE Transactions on Knowledge and Data Engineering* (2024).

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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.

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

[39] 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)*.

[40] 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.

[41] Zhebin Wu, Lin Shu, Ziyue Xu, Yaomin Chang, Chuan Chen, and Zibin Zheng. 2022. Robust tensor graph convolutional networks via t-svd based graph augmentation. In *Proceedings of the 28th ACM SIGKDD International Conference on Knowledge Discovery & Data mining*. 2090–2099.

[42] 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.