T-HyperGNNs: Hypergraph Neural Networks via Tensor Representations

Fuli Wang, *Student Member, IEEE*, Karelia Pena-Pena, *Student Member, IEEE*, Wei Qian, and Gonzalo R. Arce, *Life Fellow, IEEE*

***Abstract*— Hypergraph neural networks (HyperGNNs) are a family of deep neural networks designed to perform inference on hypergraphs. HyperGNNs follow either a spectral or a spatial approach, in which a convolution or message-passing operation is conducted based on a hypergraph algebraic descriptor. While many HyperGNNs have been proposed and achieved state-of-the-art performance on broad applications, there have been limited attempts at exploring high-dimensional hypergraph descriptors (tensors) and joint node interactions carried by hyperedges. In this article, we depart from hypergraph matrix representations and present a new tensor-HyperGNN (T-HyperGNN) framework with cross-node interactions (CNIs). The T-HyperGNN framework consists of T-spectral convolution, T-spatial convolution, and T-message-passing HyperGNNs (T-MPHN). The T-spectral convolution HyperGNN is defined under the t-product algebra that closely connects to the spectral space. To improve computational efficiency for large hypergraphs, we localize the T-spectral convolution approach to formulate the T-spatial convolution and further devise a novel tensor-message-passing algorithm for practical implementation by studying a compressed adjacency tensor representation. Compared to the state-of-the-art approaches, our T-HyperGNNs preserve intrinsic high-order network structures without any hypergraph reduction and model the joint effects of nodes through a CNI layer. These advantages of our T-HyperGNNs are demonstrated in a wide range of real-world hypergraph datasets. The implementation code is available at https://github.com/wangfuli/T-HyperGNNs.git.**

***Index Terms*— Convolution, hypergraphs, message passing, neural networks, tensors.**

Nomenclature

Symbol	Description
$\mathcal{G}$	Hypergraph structure $\mathcal{G}$.
$\mathcal{V}$	Vertex set of a hypergraph.
$\mathcal{E}$	Hyperedge set of a hypergraph.
N	Number of nodes in a hypergraph, i.e., $\lvert\mathcal{V}\rvert$.
M	Maximum cardinality of hyperedges.
$\mathbf{X}$	Original feature/signal matrix, $\mathbf{X} \in \mathbb{R}^{N\times D}$.
$\mathcal{A}$	Hypergraph adjacency tensor, $\mathcal{A} \in \mathbb{R}^{N^M}$.
$\mathcal{L}$	Hypergraph Laplacian tensor, $\mathcal{L} \in \mathbb{R}^{N^M}$.
$\mathcal{X}$	Hypergraph interaction tensor, $\mathcal{X} \in \mathbb{R}^{N\times D\times N^{(M-2)}}$.
$\mathcal{W}$	Weight tensor to be learned, $\mathcal{W} \in \mathbb{R}^{D\times D'\times N^{(M-2)}}$.
$\mathcal{Z}$	Convoluted hypergraph signal, $\mathcal{Z} \in \mathbb{R}^{N\times D'\times N^{(M-2)}}$.
$(\cdot)_s$	Symmetric tensor obtained according to Appendix A.
$(\cdot)^{\text{norm}}$	Normalized representation tensor according to Appendix C.
e^M	Mth-order hyperedge, $\texttt{length}(e^M) = M$.
a_e	Adjacency value associated with a hyperedge e.
$E^M(v)$	Mth-order incidence edge set of node v.
$\mathbf{m}_{E^M(v)}$	Embedding of $E^M(v)$, $\mathbf{m}_{E^M(v)} \in \mathbb{R}^{1\times D}$.
$\mathcal{N}^M(v)$	Mth-order neighborhood of node v.
$\mathbf{m}_{\mathcal{N}^M(v)}$	Embedding of $\mathcal{N}^M(v)$, $\mathbf{m}_{\mathcal{N}^M(v)} \in \mathbb{R}^{1\times D}$.

Manuscript received 31 January 2023; revised 21 September 2023 and 18 December 2023; accepted 23 February 2024. This work was supported in part by the National Science Foundation under Grant CCF-2230161, Grant CCF-2230162, and Grant DMS-1916376; in part by the Air Force Office of Scientific Research (AFOSR) Award under Grant FA9550-22-1-0362; and in part by the Institute Financial Services Analytics, University of Delaware. *(Corresponding author: Fuli Wang.)*

Fuli Wang is with the Institute for Financial Services Analytics, University of Delaware, Newark, DE 19716 USA (e-mail: fuliwang@udel.edu).

Karelia Pena-Pena and Gonzalo R. Arce are with the Department of Electrical and Computer Engineering, University of Delaware, Newark, DE 19716 USA

Wei Qian is with the Department of Applied Economics and Statistics, University of Delaware, Newark, DE 19716 USA.

This article has supplementary downloadable material available at https://doi.org/10.1109/TNNLS.2024.3371382, provided by the authors.

Digital Object Identifier 10.1109/TNNLS.2024.3371382

I. Introduction

MACHINE learning on graphs has drawn much attention in the last few years as graphs can represent non-Euclidean relations in data [1]. Graph neural networks (GNNs), in particular, have shown promise in various domains, such as computer vision [2], [3], recommendation [4], [5], and reinforcement learning [6]. These graph structures modeled by GNNs, however, are assumed to be pairwise relationships. In other words, each relational edge connects exactly two entities, as shown in Fig. 1(a). In real-world applications where polyadic relationships among multiple objects are important, GNNs become insufficient to capture all useful features [7]. For example, biomedical reactions often contain more than two drugs [8], and traffic flows usually are determined by more than two locations [9]. This brings up the concept of a hypergraph, a more general data abstraction in which each hyperedge binds a group of nodes simultaneously [see Fig. 1(b) and (c)].

One convenient way to study hypergraphs is to map them into regular graphs and adopt simple graph convolution to approximate high-order relationships. This approach of reducing hypergraphs is called hypergraph expansion, which includes clique expansion [10] and star expansion [11], among several others [12]. Since the graph convolution operation is originally derived in the spectral domain [10], we call

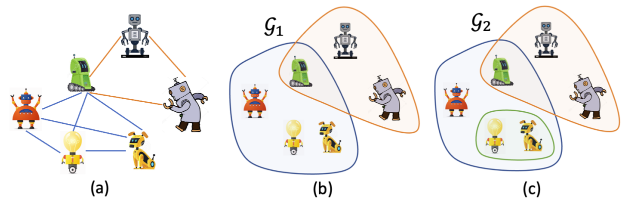


Fig. 1. Robot collaboration network represented by (a) simple graph and (b) hypergraph $\mathcal{G}_1$, and (c) another hypergraph $\mathcal{G}_2$. In (a), each cooperation relationship is denoted by a line connecting exactly two entities, whereas in (b) and (c), each hyperedge denoted by a colored ellipse represents multirobot cooperation.

them spectral hypergraph neural networks (HyperGNNs). Despite the simplicity, these methods could cause topological distortion and difficulty in downstream tasks since the mapping from a hypergraph to its corresponding simple graph is not one-to-one [13], [14]. For example, if we consider the clique expansion that connects any two nodes in a hyperedge, it is easy to verify that hypergraphs $\mathcal{G}_1$ in Fig. 1(b) and $\mathcal{G}_2$ in Fig. 1(c) have the same pairwise connections, which is the simple graph in Fig. 1(a). Other types of HyperGNNs, such as hypergraph network with hyperedge neurons (HNHN) [11] and HyperSAGE [15], defined by a two-stage spatial message-passing rule that gathers information from the neighboring nodes of each central node, utilize more advanced deep learning architectures but are still limited to linear aggregations without modeling higher order multiplicative interactions among nodes.

Recently, approaches that do not require the use of hypergraph expansions have been proposed to fully exploit polyadic relationships. In particular, tensor–tensor multiplications (t-products) [16] were introduced to better understand hypergraph operations such as signal shifting and spectral filtering, thus offering powerful tools to formulate spectral convolutions [17], [18]. Given these tensor representations and operations, several intriguing questions naturally arise: 1) can we efficiently describe hypergraph structures in a high-dimensional space without information loss? 2) can we model node interactions to represent their joint effects within a hyperedge? and 3) is it possible to generalize common GNN architectures, such as spectral convolution, spatial convolution, and message passing under the tensorial setting of hypergraphs? To address these questions, instead of collapsing hypergraphs to simple graphs and representing reduced graphs in matrix forms, we study the hypergraph representation learning by a tensor-based approach. For simplicity, we call this new framework tensor-hypergraph neural networks (T-HyperGNNs). Compared to matrix-based HyperGNNs, the proposed T-HyperGNN framework takes the following advantages.

1) Drawing from previous work, matrix representations of hypergraphs are limited by either information loss or an inability to undergo eigendecomposition [7], [11], [14] and, therefore, spectral analysis [12]. We leverage techniques from tensor hypergraph signal processing (t-HGSP) [18] to encode hypergraph structures in tensors and develop both spectral and spatial convolutions.
2) We introduce the idea of modeling node interactions via multiplicative interaction tensors. Inside the interaction tensor, the aggregation is raised from classic low-order linear operations (e.g., sum and average) [10], [11], [19] to higher order polynomial maps, thus significantly enhancing the expressiveness of HyperGNNs.
3) We scale up the tensor spatial convolution to formulate the tensor-message-passing operation by making use of tensor sparsity. The tensor-message-passing hypergraph neural network (T-MPHN), in particular, is capable of processing large hypergraphs with efficient space and computational complexities comparable to matrix-based HyperGNNs.

The rest of this article is organized as follows. We introduce the necessary background and related work in Section II. We then define the cross-node interaction (CNI) tensor and T-spectral convolution in Section III. To tackle the complexity of T-spectral convolution, in Section IV, we first localize the T-spectral convolution to form T-spatial convolution and then propose an inductive and scalable T-MPHN. Connections between our three methods and other existing HyperGNNs are illustrated in Section V. The numerical experiments are summarized in Section VI. A conclusion is given in Section VII.

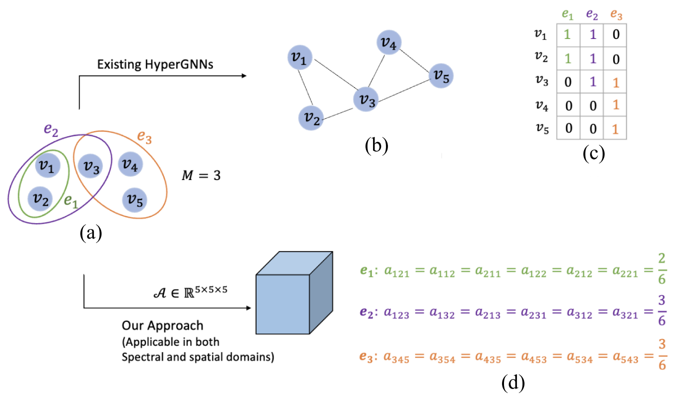
II. Background and Related Work

A. Hypergraph Signal Processing

A hypergraph $\mathcal{G}$ is defined as a pair of two sets $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ denotes the set of N nodes (or vertices) and $\mathcal{E} = \{e_1, e_2, \ldots, e_K\}$ is the set of K hyperedges whose elements e_k $(k = 1, 2, \ldots, K)$ are nonempty subsets of $\mathcal{V}$. The maximum cardinality of edges, or $m.c.e(\mathcal{G})$, is denoted by M, which defines the order of a hypergraph. Apart from the hypergraph structure, there are also features $\mathbf{x}_v \in \mathbb{R}^D$ associated with each node $v \in \mathcal{V}$, which are used as row vectors to construct the feature matrix $\mathbf{X} \in \mathbb{R}^{N\times D}$.

A hypergraph structure $\mathcal{G}$ can be encoded in either a matrix or a tensor form. We refer to these algebraic descriptors as $\mathcal{S}$. In matrix representations, a hypergraph is usually described as a vertex-to-hyperedge incidence matrix $\mathbf{H} \in \mathbb{R}^{N\times K}$. As shown in Fig. 2(c), entries of the incidence matrix are $h_{nk} = 1$ if node v_n lies in hyperedge e_k, and $h_{nk} = 0$, otherwise. While the incidence matrix representation is a loss-free representation, its rectangular shape hinders two of the most powerful techniques: convolution and spectral analysis. Another matrix descriptor known as the adjacency matrix of a hypergraph is defined as $\mathbf{A} = \mathbf{H}\mathbf{H}^T$, which projects out the hyperedge dimension but leads to clique expansion (see Fig. 2(b) where e_1 is dismissed) that causes distortion of hypergraph structures [13], [14]. To address the limitations in the hypergraph matrix representations, we propose to use tensor descriptors and formulate a novel T-HyperGNN framework.

Definition 1 (Hypergraph Adjacency Tensor [20], [21]): Given a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with N nodes of order M (i.e., $m.c.e(\mathcal{G}) = M$), its adjacency tensor is defined as an Mth-order N-dimensional tensor $\mathcal{A} \in \mathbb{R}^{N^M}$. Specifically, for any hyperedge $e_k = \{v_{k_1}, v_{k_2}, \ldots, v_{k_c}\} \in \mathcal{E}$ with $c = |e_k| \le M$,


Existing HyperGNNs
e_2
e_3
e_1
v_1
v_2
v_3
v_4
v_5
$M = 3$
(a)
(b)
(c)
$\mathcal{A} \in \mathbb{R}^{5\times5\times5}$
Our Approach
(Applicable in both
Spectral and spatial domains)
e_1: $a_{121} = a_{112} = a_{211} = a_{122} = a_{212} = a_{221} = \frac{2}{6}$
e_2: $a_{123} = a_{132} = a_{213} = a_{231} = a_{312} = a_{321} = \frac{3}{6}$
e_3: $a_{345} = a_{354} = a_{435} = a_{453} = a_{534} = a_{543} = \frac{3}{6}$
(d)

	e_1	e_2	e_3
v_1	1	1	0
v_2	1	1	0
v_3	0	1	1
v_4	0	0	1
v_5	0	0	1

Fig. 2. (a) Hypergraph with (b) its clique expansion that is used in spectral HyperGNNs, (c) incidence matrix is utilized in spatial HyperGNNs, and (d) adjacency tensor, where nonzero entries of the adjacency tensor are specified on the right. The adjacency tensor is applicable to both spectral and spatial approaches.

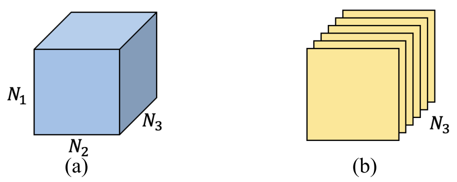


Fig. 3. (a) Third-order tensor $\mathcal{A} \in \mathbb{R}^{N_1\times N_2\times N_3}$. (b) N_3 frontal slices $\mathbf{A}^{[k]} = \mathcal{A}(:,:,k) \in \mathbb{R}^{N_1\times N_2\times 1}$.

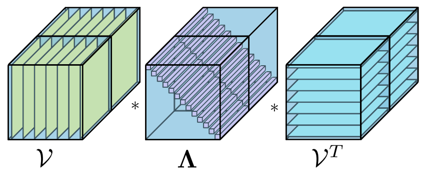


Fig. 4. Visual illustration of the T-eigendecomposition for a 3rd-order Laplacian tensor $\mathcal{L}_s^{\text{norm}}$.

the tensor's corresponding entries are given by

$$a_{p_1p_2\dots p_M} = \frac{c}{\alpha} \tag{1}$$

with

$$\alpha = \sum_{r_1,r_2,\dots,r_c\geq 1,\ \sum_{i=1}^{c} r_i = M} \binom{M}{r_1, r_2, \dots, r_c} \tag{2}$$

where the indices $p_1, p_2, \dots, p_M$ for adjacency entries are chosen from all possible ways of $\{k_1, k_2, \dots, k_c\}$'s permutations with at least one appearance for each element of the hyperedge set, and α is the sum of multinomial coefficients with the additional constraint $r_1, r_2, \dots, r_c \neq 0$, as demonstrated in Fig. 1(d). In addition, a hypergraph adjacency tensor with an order higher than 3 is included in Appendix J (see the Supplementary Material). Note that a hypergraph with $M = 2$ degrades to a simple graph with a binary adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$.

Despite the rich theoretical support [20], [21], [22], the formula introduced above generalizes a fundamental property of the graph adjacency matrix to higher orders, accommodating complex relationships in hyperedges. This key property states that the sum of entries along any $(M-1)$ dimensions, with the remaining one fixed, should yield the degrees of the nodes, i.e., $d_{v_i} = \sum_{j_1,j_2,\dots,j_{M-1}=1}^{N} a_{ij_1j_2\dots j_{M-1}}$. Node degrees represent the number of edges connected to each node. For example, in Fig. 2(d), summing the adjacency tensor along the horizontal dimension for each node yields the node degrees: [2, 2, 2, 1, 1]. The degree tensor $\mathcal{D} \in \mathbb{R}^{N^M}$ is a super-diagonal degree tensor with the degree of node v_i on the corresponding diagonal entry $d_{i\dots i}$. The Laplacian tensor is then defined as $\mathcal{L} = \mathcal{D} - \mathcal{A}$.

For ease of presentation, we first describe the notation for 3rd-order tensors as 3rd-order tensors form the base case of higher order tensors. For a 3rd-order tensor, indices $i \in \{1, 2, \dots, N_1\}$, $j \in \{1, 2, \dots, N_2\}$, and $k \in \{1, 2, \dots, N_3\}$ are used to specify the height, width, and depth direction of the cube in Fig. 3(a). Breaking down a 3rd-order tensor along the third mode, we obtain frontal slices in Fig. 3(b), where the kth frontal slice is $\mathbf{A}^{[k]} = \mathcal{A}(:,:,k) \in \mathbb{R}^{N_1\times N_2\times 1}$. When it comes to Mth-order tensors $\mathcal{A} \in \mathbb{R}^{N^M}$, we can view the last $(M-2)$ orders as flattened frontal slice indices along the third order, that is, $\mathcal{A} \in \mathbb{R}^{N_1\times N_2\times N_3'}$ with $N_3' = N_3N_4\cdots N_M$.

In HGSP [18], a symmetric adjacency and Laplacian hypergraph tensor descriptors are introduced since the tensors $\mathcal{A}$ and $\mathcal{L}$ above are not symmetric under the t-product algebra. Therefore, the operator $\texttt{sym}(\mathcal{A})$ generates a symmetric version $\mathcal{A}_s \in \mathbb{R}^{N\times N\times N_s}$ of $\mathcal{A} \in \mathbb{R}^{N\times N\times N}$, with $N_s = 2N + 1$ according to the symmetrization operation in Appendix A (see the Supplementary Material). The motivations of the symmetrization operation are: 1) to obtain an orthonormal set of the hypergraph spectral space that is defined by the eigendecomposition of the normalized Laplacian $\mathcal{L}_s^{\text{norm}} = \mathcal{V} * \Lambda * \mathcal{V}^T$ as shown in Fig. 4, where $\mathcal{L}_s = \mathcal{I}_s - \mathcal{A}_s^{\text{norm}}$ is the normalized symmetric Laplacian tensor and $\mathcal{I}_s$ is an identity tensor (with the first frontal slice being identity matrix and the other entries being zero) as shown in Appendix C (see the Supplementary Material), and 2) to obtain a symmetric block circulant matrix in the T-product $*$ that is to be introduced in (3).

Definition 2 (T-Product [16], [18]): Let $\mathcal{A}_s \in \mathbb{R}^{N\times N\times N_s}$ and $\mathcal{X}_s \in \mathbb{R}^{N\times D\times N_s}$ be two 3rd-order symmetric tensors, and their T-product $\mathcal{Y}_s \in \mathbb{R}^{N\times D\times N_s}$ is given by

$$\begin{aligned}
&\mathcal{A}_s * \mathcal{X}_s \\
&= \texttt{fold}(\texttt{bcirc}(\mathcal{A}_s)\cdot\texttt{unfold}(\mathcal{X}_s)) \\
&= \texttt{fold}\left(\begin{bmatrix}
\mathbf{0} & \mathbf{A}^{[1]} & \mathbf{A}^{[2]} & \cdots & \mathbf{A}^{[2]} & \mathbf{A}^{[1]} \\
\mathbf{A}^{[1]} & \mathbf{0} & \mathbf{A}^{[1]} & \cdots & \mathbf{A}^{[3]} & \mathbf{A}^{[2]} \\
\mathbf{A}^{[2]} & \mathbf{A}^{[1]} & \mathbf{0} & \cdots & \mathbf{A}^{[4]} & \mathbf{A}^{[3]} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\mathbf{A}^{[2]} & \mathbf{A}^{[3]} & \mathbf{A}^{[4]} & \cdots & \mathbf{0} & \mathbf{A}^{[1]} \\
\mathbf{A}^{[1]} & \mathbf{A}^{[2]} & \mathbf{A}^{[3]} & \cdots & \mathbf{A}^{[1]} & \mathbf{0}
\end{bmatrix}
\begin{bmatrix}
\mathbf{0} \\ \mathbf{X}^{[1]} \\ \mathbf{X}^{[2]} \\ \vdots \\ \mathbf{X}^{[2]} \\ \mathbf{X}^{[1]}
\end{bmatrix}\right)
\end{aligned} \tag{3}$$

where the operator $\texttt{bcirc}(\mathcal{A}_s)$ converts the set of N_s frontal slice matrices (in $\mathbb{R}^{N\times N}$) of the tensor $\mathcal{A}_s$ into a block circulant matrix. Specifically, the first row/column of $\texttt{bcirc}(\mathcal{A}_s)$ is the frontal slices of $\mathcal{A}_s$, i.e., $[\mathbf{0}, \mathbf{A}^{[1]}, \mathbf{A}^{[2]}, \dots, \mathbf{A}^{[N]}, \mathbf{A}^{[N]}, \dots, \mathbf{A}^{[2]}, \mathbf{A}^{[1]}]$ and the next row/column is simply the one-step cyclic shifting of the previous row/column. The operation $\texttt{unfold}(\mathcal{X}_s)$ vertically stacks the set of N_s frontal slice matrices (in $\mathbb{R}^{N\times D}$) of $\mathcal{X}_s$ into an $N_sN \times D$ matrix. The operator $\texttt{fold}()$ is the reverse of the $\texttt{unfold}()$ process so that $\texttt{fold}(\texttt{unfold}(\mathcal{A}_s)) = \mathcal{A}_s$. The t-product of higher order tensors is more involved with recursive computation with 3rd-order base cases, which is relegated to Appendix B (see the Supplementary Material) for technical completeness.

B. Problem Statement

The idea of HyperGNNs is to build neural networks on hypergraphs such that higher order relationships among entities as well as any available node attributes are unlocked. Formally, given a descriptor $\mathcal{S}$ of a hypergraph and the associated node features in $\mathbf{X}$, the goal of HyperGNNs is to identify a representation map $\Phi(\cdot)$ between the feature $\mathbf{X}$ and the target representation $\mathbf{t} = \Phi(\mathbf{X}, \mathcal{S}, \{\mathcal{W}\})$ that incorporates the hypergraph structure, where $\{\mathcal{W}\}$ contains the weight parameters learned by the model. To learn the representation map, we consider a cost function $J(\cdot)$ and a training set $\mathcal{T} = \{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_{|\mathcal{T}|}, t_{|\mathcal{T}|})\}$ with the observed training targets $\mathbf{t} = (t_1, \ldots, t_{|\mathcal{T}|})$. The learned map is then $\Phi(\mathbf{X}, \mathcal{S}, \mathcal{W}^*)$ with

$$\mathcal{W}^* = \underset{\mathcal{W}}{\operatorname{argmin}}\, J(\Phi(\mathbf{X}, \mathcal{S}, \mathcal{W}),\ \mathbf{t}). \tag{4}$$

C. Existing HyperGNNs

The research of HyperGNNs can be briefly categorized into two main approaches: 1) spectral methods that define convolution in the spectral space and 2) spatial methods that aggregate neighboring messages and combine with self-embedding for each node.

1) Matrix-Based Spectral HyperGNNs: The earliest attempt to build HyperGNNs includes HGNN [10] and HCHA [23], which can be considered as spectral HyperGNNs built on the adjacency matrix $\mathbf{A}$ of a hypergraph. From the adjacency matrix, the hypergraph Laplacian is defined to construct the hypergraph spectral space that is formed by the eigendecomposition of the Laplacian matrix. After applying first-order approximation to spectral filters, the spectral convolution is formulated as $\mathbf{Z} = \mathbf{A}^{\text{norm}}\mathbf{X}\mathbf{W}$, where $\mathbf{A}^{\text{norm}} \in \mathbb{R}^{N\times N}$ is a normalized adjacency matrix, $\mathbf{X} \in \mathbb{R}^{N\times D}$ is the feature matrix, and $\mathbf{W} \in \mathbb{R}^{D\times D'}$ is a learnable filter weight matrix. Although $\mathbf{A}$ is a squared matrix, it is geometrically equivalent to the clique expansion, in which a hypergraph is reduced to a simple graph by connecting any two nodes that are in a hyperedge. For instance, the simple graph in Fig. 2(b) is the clique expansion of the hypergraph in Fig. 2(a). With such reduction, the small edge e_1 contained in e_2 is ignored. Thus, the hypergraph expansion is not a one-to-one mapping, which could cause node- and edge-level ambiguities [14]. Other methods, such as HyperGCN [24] and LEGCN [25], are developed following similar ideas with different variants of matrix descriptors.

2) Matrix-Based Spatial HyperGNNs: In contrast to spectral HyperGNNs, spatial HyperGNNs focus on the local connectivity of each node without going to the spectral domain. By defining the incident-edge set of node v as $E_v = \{e \in \mathcal{E} \mid v \in e\}$, UniGNN [19] proposes a spatial message-passing process with two steps

$$\begin{cases} \mathbf{x}_e = \phi_1\left(\{\mathbf{x}_u\}_{u\in e}\right) \\ \mathbf{z}_v = \phi_2\left(\mathbf{x}_v, \{\mathbf{x}_e\}_{e\in E_v}\right) \end{cases} \tag{5}$$

where ϕ_1 and ϕ_2 are two permutation-invariant functions for node-to-edge and edge-to-node aggregations, respectively. Specifically, the first step aggregates information from all nodes that are in each incident edge, thus forming a node-to-edge propagation. The edge embedding $\mathbf{x}_e$ is then combined with the target node embedding $\mathbf{x}_v$ and passes through ϕ_2 to produce a new node embedding $\mathbf{z}_v$. Such a node–edge–node embedding scheme remains to be matrix-based since it is a generalization of $\mathbf{Z} = \mathbf{H}(\mathbf{H}^T\mathbf{X})$, where $\mathbf{H}$ is the incidence matrix. In addition to UniGNN, current methods, including HNHN [11], HyperSAGE [15], and AllSet [13], are all under such node–edge–node propagation paradigm, but with more advanced architectures such as an attention mechanism. Compared to spectral HyperGNNs, spatial message passing does not require the construction of a hypergraph algebraic descriptor and can be applied to previously unseen nodes during testing. However, it remains unclear if an appropriate higher order descriptor (i.e., a tensor) can be employed to accommodate hypergraph structures.

In summary, the following issues remain unsolved for these existing HyperGNNs. First, they are based on matrix descriptors with possible information loss. For example, the adjacency matrix $\mathbf{A}$ corresponds to a clique-expanded simple graph, which could not encode all intrinsic higher order structures. Second, they do not consider possible high-order feature interactions among multiple nodes. Indeed, the salient characteristic of hypergraphs compared to simple graphs is that hyperedges depict the joint effects of a group of nodes. Finally, spectral and spatial HyperGNNs are studied separately in the literature, while a more unified study connecting both approaches would be desirable.

To overcome the aforementioned issues, we propose the tensorial descriptor of the hypergraph structure and further construct the hypergraph interaction tensor by modeling CNIs. Using these two tensors, we design hypergraph spectral convolution under the t-algebra framework and then localize the spectral convolution to form spatial convolution that only propagates to neighbors of each node. Spatial message-passing HyperGNNs are then built upon the compressed adjacency tensor to address tensor complexity for developing computationally efficient algorithms. For the convenience of the reader, the important notations used in this article are listed in the Nomenclature.

III. T-Spectral HyperGNNs

In this section, we present a tensor-based spectral convolution formulation on hypergraphs, utilizing well-established tools in hypergraph signal processing (HGSP) [18]. The T-spectral convolution approach aims to leverage the power of HGSP to develop effective hypergraph convolutional neural networks that can process and analyze higher order geometric data. We first introduce the hypergraph interaction tensor $\mathcal{X}$ by modeling CNIs. Based on the hypergraph tensor representations and the interaction tensor, the hypergraph T-spectral convolution is proposed by leveraging hypergraph spectral filtering, which is defined in Definition 3.

A. Modeling CNIs

The CNI tensor $\mathcal{X}$ is designed as the $(M-1)$-time outer product of feature matrix $\mathbf{X} \in \mathbb{R}^{N\times D}$ along each feature dimension $d = 1,\dots,D$. The dth dimensional interaction among all nodes $(d = 1,\dots,D)$ is given by

$$CNI([\mathbf{x}]_d) = \underbrace{[\mathbf{x}]_d \circ [\mathbf{x}]_d \circ \cdots \circ [\mathbf{x}]_d}_{(M-1)\text{times}} \in \mathbb{R}^{N\times 1\times N^{(M-2)}} \tag{6}$$

where $\circ$ denotes the outer product (also known as elementary tensor product) and $[\mathbf{x}]_d \in \mathbb{R}^N$ represents the dth dimensional feature vector of all N nodes. For example, given $M = 3$, $\text{CNI}([\mathbf{x}]_d) = [\mathbf{x}]_d[\mathbf{x}]_d^T \in \mathbb{R}^{N\times 1\times N}$. Here, we unsqueeze the outer product tensor to generate the additional second mode for the dimension index of different features. Then, by computing $\text{CNI}([\mathbf{x}]_d)$ for all D features and stacking them together along the second-order dimension, we obtain an Mth-order interaction tensor $\mathcal{X} \in \mathbb{R}^{N\times D\times N^{(M-2)}}$. The resulting interaction tensor can be viewed as a collection of D tensors, each depicting node interactions at one feature dimension. The formulation of the CNI tensor has the following unique properties: 1) interactions capture features that cannot be decomposed into sums of subfunctions of node features; 2) interactions are applied across different linked nodes, as opposed to different features; and 3) the order of interactions grows naturally with increasing order of complexity of the hypergraph.

Remark: Although the cross-channel multiplicative interaction has been widely used in recommendation systems (e.g., deep and cross network (DCN) [26] and eXtreme Deep Factorization Machine (xDeepFM) [27]) and high-dimensional regression [28], here we design the interactions to be nonlinear cross-node (as opposed to cross-channel) based on the intrinsic node interactions depicted in hyperedges, which significantly enhances the expressive power of node aggregations by going beyond linear summations.

B. Hypergraph T-Spectral Convolution

The foundation of GNN development traces back to spectral filtering, established in the field of graph signal processing (GSP) [29], [30]. Building upon this idea, we embrace the newly proposed t-HGSP framework [18] to introduce our novel approach: hypergraph T-spectral convolution.

Definition 3 (Hypergraph Spectral Filtering [18], [29]): Given the normalized Laplacian tensor $\mathcal{L}_s^{\text{norm}}$ of a hypergraph as shown in Appendix C (see the Supplementary Material), the spectrum space of the hypergraph is defined by the t-eigendecomposition $\mathcal{L}_s^{\text{norm}} = \mathcal{V} * \Lambda * \mathcal{V}^T$ (see Appendix D for details, see the Supplementary Material). The hypergraph spectral filter $h : \mathbb{R} \to \mathbb{R}$ is a function of the frequency response $h(\mathbf{\Lambda})$, and the spectral filtering operation on hypergraph $\mathcal{G}$ is defined as

$$\mathcal{H}_s \star_{\mathcal{G}} \mathcal{X}_s = \mathcal{V} * h(\Lambda) * \mathcal{V}^T * \mathcal{X}_s \tag{7}$$
$$= h(\mathcal{L}_s^{\text{norm}}) * \mathcal{X}_s \tag{8}$$

with the frequency response

$$h(\Lambda) = \begin{bmatrix} h(\boldsymbol{\lambda}_1) & \cdots & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & \cdots & h(\boldsymbol{\lambda}_N) \end{bmatrix}.$$

Instead of performing eigendecomposition, a more computationally efficient approach is to approximate $h(\Lambda)$ by a truncated expansion of Chebyshev polynomials $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, with $T_0 = 1, T_1 = x$ up to the Kth order, as proposed in ChebyNet [30]. Since the Chebyshev polynomial is defined recursively, which can be achieved by cascading multiple layers of neural networks, following the first-order approximation proposed in graph convolutional network (GCN) [31], we can further let $K = 1$, and the convolution operation is simplified to

$$\mathcal{Z}_s = \mathcal{H}_s \star_{\mathcal{G}} \mathcal{X}_s = \theta_0\mathcal{X}_s + \theta_1\mathcal{V} * (\mathbf{\Lambda} - \mathcal{I}_N) * \mathcal{V}^T * \mathcal{X}_s \tag{9}$$
$$= \theta_0\mathcal{X}_s + \theta_1(\mathcal{L}_s^{\text{norm}} - \mathcal{I}_N) * \mathcal{X}_s \tag{10}$$
$$= \theta_0\mathcal{X}_s - \theta_1\mathcal{A}_s^{\text{norm}} * \mathcal{X}_s. \tag{11}$$

Unifying the two parameters θ_0 and θ_1 as $w = \theta_0 = -\theta_1$, the convolution is further simplified as

$$\mathcal{Z}_s = \mathcal{H}_s \star_{\mathcal{G}} \mathcal{X}_s = w(\mathcal{I}_s + \mathcal{A}_s^{\text{norm}}) * \mathcal{X}_s \tag{12}$$

where $(\mathcal{I}_s + \mathcal{A}^{\text{norm}})$ can be treated as an adjusted hypergraph adjacency tensor with a self-loop of each node. For notation brevity, we use $\mathcal{A}^{\text{norm}}$ to denote normalized adjacency tensors with either self-loop or not. When the hypergraph signal is in D-dimension, i.e., $\mathcal{X}_s \in \mathbb{R}^{N\times D\times N_s^{(M-2)}}$, and the desired convoluted hypergraph signal $\mathcal{Y}_s$ is in D'-dimension, i.e., $\mathcal{Y}_s \in \mathbb{R}^{N\times D'\times N^{(M-2)}}$, the weights will be characterized by a bank of DD' parameters instead of a single value w. In this way, the T-spectral convolution is given by

$$\mathcal{Z}_s = \mathcal{A}_s^{\text{norm}} * \mathcal{X}_s * \mathcal{W}_s \tag{13}$$

where $\mathcal{W}_s \in \mathbb{R}^{D\times D'\times N_s^{(M-2)}}$ is the weight tensor with DD' weights parameterized in the first frontal slice and the remaining frontal slices from 2 throughout $2N+1$ are all zeros for parameter sharing across nodes. The reason for constructing the T-spectral convolution using the t-product $*$ is that the tensor $\mathcal{V}$ forms an orthonormal set, i.e., $\mathcal{V}^T * \mathcal{V} = \mathcal{V} * \mathcal{V}^T = \mathcal{I}_s$, with $\mathcal{I}_s$ being a f-diagonal tensor [16] whose first frontal slice is an identity matrix and other frontal slices are all zeros. In this way, the Fourier transform and the inverse Fourier transform of a given feature tensor are perfectly achieved without approximation or information loss [18].

C. Implementation Details

With the T-spectral convolution defined in (12), a T-spectral convolutional HyperGNN can be built by cascading L layers with

$$\mathcal{X}_s^{(l+1)} = \sigma(\mathcal{A}_s^{\text{norm}} * \mathcal{X}_s^{(l)} * \mathcal{W}_s^{(l)}), \quad l = 0, 1, \dots, L-1 \tag{14}$$

where the initial interaction tensor $\mathcal{X}_s^{(0)}$ is obtained by (6) and $\sigma(\cdot)$ is an activation function. In the final layer of the T-spectral convolutional HyperGNN, after obtaining the tensor $\mathcal{X}_s^{(L)}$, an inverse symmetrization operation is first applied to obtain $\hat{\mathcal{X}} = \mathcal{X}^{(L)}$, and a readout function $f : \mathbb{R}^{N\times C\times N^{(M-2)}} \to \mathbb{R}^{N\times C}$ is utilized to generate the node representations $\mathbf{X}_{\text{out}} \in \mathbb{R}^{N\times C}$, where C is the number of classes

$$\mathbf{X}_{\text{out}} = \sum_{k=1}^{N^{(M-2)}} \hat{\mathbf{X}}^{[k]} \tag{15}$$

with $[\hat{\mathbf{X}}^{[1]}, \hat{\mathbf{X}}^{[2]}, \ldots, \hat{\mathbf{X}}^{[N^{(M-2)}]}] = \texttt{unfold}(\hat{\mathcal{X}})$ being flattened frontal slices of $\hat{\mathcal{X}}$.

D. Complexity Analysis

As the order and the number of nodes of a hypergraph increase, the time and space complexity of the T-spectral convolution becomes a major concern. Indeed, the computation in a one-step t-convolution of the tensors $\mathcal{A}_s^{\text{norm}} * \mathcal{X}_s * \mathcal{W}_s$ in (12) can be shown to be $\mathcal{O}(DN^{2M})$, which is practically difficult given any moderate M. Since circulant matrices are diagonalized by the discrete Fourier transform, as shown in Algorithm 2 in Appendix E (see the Supplementary Material), the t-product can be efficiently computed by recursively applying the fast Fourier transform to both tensors, followed by regular matrix products between flattened tensors and eventually performing inverse fast Fourier transform.

Even though the computation of the t-product could be reduced to $\mathcal{O}(DN^M)$ using Algorithm 2, it is still not sufficiently fast in large hypergraph learning. In addition, considering the space complexity of a Mth-order hypergraph, the memory allocated for the adjacency tensor is $\mathcal{O}(N^M)$. Since tensor-based convolutions require that the full hypergraph adjacency tensor is known during the model training process, a direct implementation is usually not feasible for large hypergraphs. We address these limitations in Section IV.

IV. T-Spatial HyperGNNs

To scale up the T-spectral convolution, two improvements are proposed in this section. First, we localize the T-spectral convolution to form a T-spatial convolution that only propagates to connected neighbors of each node. Two important properties of the T-spatial convolution are highlighted in Propositions 1 and 2. Second, to alleviate the space complexity of tensors, we introduce the compressed adjacency tensor that takes little memory usage. The compressed adjacency tensor consists of two tables: the adjacency value table where the adjacency values are computed based on Theorem 1 and the neighborhood table that records higher order hyperedge and neighborhood of a node based on Definitions 4 and 5, respectively. Given the compressed adjacency tensor, a two-step message-passing framework is proposed, within which the T-spatial convolution is subsumed.

A. T-Spatial Convolution

In the vertex domain, convolution is viewed as a weighted sum of neighboring information. As a result, the main idea of developing spatial convolution is to localize the spectral convolution, that is, only connected nodes should be propagated through during a shifting operation $\mathcal{A}_s^{\text{norm}} * \mathcal{X}_s$. Interestingly, we found that only the first frontal slice (matrix) of the resulting tensor $(\mathcal{A}_s^{\text{norm}} * \mathcal{X}_s)^{[1]} \in \mathbb{R}^{N\times D}$ has local meaning in the vertex domain. To this end, we define the T-spatial convolution as

$$\mathbf{Z}_{\mathcal{G}} = \left(\mathcal{A}_s^{\text{norm}} * \mathcal{X}_s\right)^{[1]}\mathbf{W} \tag{16}$$

where $\mathcal{A}_s^{\text{norm}}$ and $\mathcal{X}_s$ are the corresponding symmetrized tensors, $(\mathcal{A}_s^{\text{norm}} * \mathcal{X}_s)^{(1)}$ is the first frontal slice of the shifted hypergraph signal $\mathcal{A}_s^{\text{norm}} * \mathcal{X}_s$, and $\mathbf{W} \in \mathbb{R}^{D\times D'}$ is a learnable weight matrix.

Also, note that since the circulant operation is muted now, the symmetrization is not needed anymore and the T-spatial aggregation $(\mathcal{A}_s^{\text{norm}} * \mathcal{X}_s)^{[1]}$ can be computed as the sum of the corresponding frontal slice products between $\mathcal{A}^{\text{norm}}$ and $\mathcal{X}$. The resulting feature matrix $\mathbf{Y} \in \mathbb{R}^{N\times D}$ is given by

$$\mathbf{Y} := \left(\mathcal{A}^{\text{norm}} * \mathcal{X}\right)^{[1]} = \sum_{k=1}^{N^{M-2}} \mathbf{A}^{[k]}\mathbf{X}^{[k]}. \tag{17}$$

Equivalently, for an individual node v_i with $1 \le i \le N$ and $1 \le d \le D$, we compute $y_{id} := [\mathbf{Y}]_{i,d}$ as

$$y_{id} = \sum_{j=1}^{N}\sum_{i_3=1}^{N}\cdots\sum_{i_M=1}^{N} a_{iji_3\cdots i_M} x_{jdi_3\cdots i_M} \tag{18}$$

where $x_{jdi_3,\ldots,i_M} = x_{jd}x_{i_3d},\ldots,x_{i_Md}$.

For example, in the 3rd-order case with $M = 3$, the first frontal slice of the shifted signal is computed as $\sum_{k=1}^{N}\mathbf{A}^{[k]}\mathbf{X}^{[k]}$, which is equivalent to computing $\sum_{j=1}^{N}\sum_{k=1}^{N} a_{ijk}x_{jd}x_{kd}$ for each node v_i $(i = 1, \ldots, N)$. Then, if three different nodes v_i, v_j, and v_k are in the same hyperedge, we have $a_{ijk} \neq 0$ and the interaction with the neighboring nodes v_j and v_k are used to compute the shifted signal for v_i; otherwise, $a_{ijk} = 0$ and the respective interaction term makes no contribution for the shifted signal.

Remark: In general, by the adjacency tensor definition and its sparse nature, the entries $a_{iji_3,\ldots,i_M}$ are the indicators to determine whether a corresponding set of nodes is connected to the target node v_i through a hyperedge. Therefore, (18) implies that only the features/signals from neighboring nodes contribute to computing the shifted signal of the target node under the T-spatial convolution, which can lead to efficient computing algorithms to be introduced in Sections IV-B and IV-C. In addition, the outcome of the T-spatial convolution does not depend on the node ordering for adjacency tensor generation. On the other hand, the other frontal slices of $\mathcal{A}_s^{\text{norm}} * \mathcal{X}_s$ (except the first one) would involve more than the neighbors of a target node and may not be computed without prior node ordering information. Therefore, these frontal slices apart from the first one are not included in the T-spatial convolution.

These two desirable properties of the T-spatial aggregation discussed above are summarized in the following propositions.

Proposition 1: The T-spatial aggregation is localized, which propagates only through neighbors of each target node.

Proposition 2: The T-spatial convolution is permutation invariant on the ordering of the nodes.

Proof: See Appendix F (Supplementary Material) for the proof of Propositions 1 and 2. □

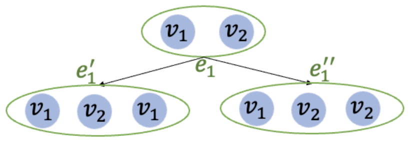


Fig. 5. Spanning the hyperedge e_1 in Fig. 2(a) with $|e_1| = 2 < M = 3$ to Mth-order sub-hyperedges $\text{span}^M(e_1)$.

B. Compressed Adjacency Tensor Representation

While the convolution operation is localized in the spatial domain through the T-spatial convolution, the space and time complexities remain too large for most applications. Our goal next is to formulate an efficient message-passing scheme that aggregates neighboring features according to (18), but without direct tensor loading in the model architecture. To this end, we first propose the compressed adjacency tensor to store the adjacency tensor compactly in this section and then formulate the spatial T-message-passing hypergraph neural network (T-MPHN) based on the compressed adjacency tensor in Section IV-C.

Returning to the hypergraph adjacency tensor introduced in Section II-A, from the hypergraph adjacency tensor example in Fig 2, we can see that the construction of the adjacency tensor can be divided into two sequential steps: 1) spanning every edge into Mth-order hyperedge and 2) permutating indices of each spanned Mth-order hyperedges.

Step 1 (Spanning Every Edge $e \in \mathcal{E}$ Into Mth-Order Hyperedges): Since hyperedges with $|e| = M$ are in Mth-order already, only hyperedges with $|e| < M$ need to be spanned.

Definition 4 (Mth-Order Hyperedge): Given a hypergraph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with the order M, for any hyperedge $e \in \mathcal{E}$, its Mth-order hyperedge set e^M is given by

$$e^M = \begin{cases} \{e\}, & \text{if } |e| = M \\ \text{span}^M(e), & \text{if } |e| < M. \end{cases} \tag{19}$$

Here, $\text{span}^M(e)$ is the set of Mth-order sub-hyperedges spanned from e with $|e| < M$

$$\text{span}^M(e) = \left\{e' \mid \texttt{unique}(e') = e,\ |e'| = M\right\} \tag{20}$$

where $\texttt{unique}(e') = e$ means that the distinct elements in e' are the same as e and $|e'|$ is the number of (possibly nonunique) elements in e'. It is not hard to see that the size of the sub-hyperedge set $|\text{span}^M(e)|$ is exactly the total number of combinations for choosing $(M - |e|)$ elements with replacement from the set e

$$\left|\text{span}^M(e)\right| = \mathcal{C}^R(|e|, (M - |e|)) = \binom{(M-1)}{(|e|-1)}. \tag{21}$$

For example, given the hypergraph of Fig. 2(c), the edge e_1 with $|e_1| = 2 < M = 3$ can be spanned to two 3rd-order sub-hyperedges $e_1' = (v_1, v_2, v_1)$ and $e_1'' = (v_1, v_2, v_2)$ as shown in Fig. 5 and $e_1^M = \text{span}^3(e_1) = \{e_1', e_1''\}$.

Step 2 (Permutating Mth-Order Hyperedges): After obtaining Mth-order hyperedges e^M for every $e \in \mathcal{E}$, we permutate elements contained in e^M (denoted by a sequence permutation function $\pi(\cdot)$), which in turn specifies the set of permuted index sequences corresponding to the adjacency entries associated with hyperedge e. Specifically, given any $(p_1, p_2, \ldots, p_M) \in \pi(e^M)$, the entry value in (1) can be equivalently written as

$$a_{p_1 p_2, \ldots, p_M} = \frac{|e|}{\left|\pi\left(e^M\right)\right|} \tag{22}$$

where the cardinality of permutated Mth-order hyperedge $|\pi(e^M)| = \alpha$ is given in (2). As we can see from (22), two types of information are associated with nonzero adjacency entries: the adjacency value corresponding to the hyperedge e and the indices capturing node connectivities. We then introduce two lookup tables to encode the information of the adjacency tensor: the adjacency value table and the node neighborhood table, which together are called the compressed adjacency tensor, and an illustrative example is shown in Fig. 6(c) and (d).

For the adjacency value table, we first discover that they can be computed efficiently as a function of the edge cardinality $|e|$ and the order M of the hypergraph.

Theorem 1: Given an adjacency tensor of a hypergraph, the adjacency value a_e associated with a hyperedge e is a function of $(|e|, M)$

$$a_e = \frac{|e|}{\alpha}$$

where

$$\alpha = \sum_{i=0}^{|e|} (-1)^i \binom{|e|}{i} (|e| - i)^M. \tag{23}$$

Proof: The proof is given in Appendix G (see the Supplementary Material). □

Given Theorem 1, the adjacency value table is easily constructed, in which the first column lists the cardinalities of hyperedges ranging from 2 (the minimum) to M (the maximum), and the second column refers to the corresponding adjacency value a_e's computed from (23). Note that the computation of the adjacency values a_e's does not rely on specific hyperedges, and hyperedges sharing the same cardinalities have the same adjacency values. Therefore, the adjacency table as shown in Fig. 6(c) is typically very short and can be stored with linear complexity.

Next, for the node neighborhood table, we introduce the concept of Mth-order neighborhood of a node.

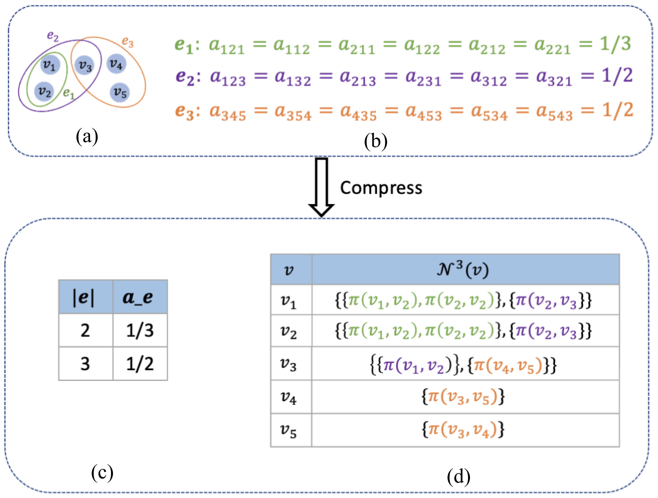
Definition 5 (Mth-Order Neighborhood of a Node): Given a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with order M, for any node $v \in \mathcal{V}$, its Mth-order incidence edge set is

$$E^M(v) := \left\{e^M \mid e \in \mathcal{E},\ v \in e\right\} \tag{24}$$

where e^M is the Mth-order hyperedge set defined in (19). Then, we can define the Mth-order neighborhood of v that basically excludes one target node v in each hyperedge from $E^M(v)$

$$\mathcal{N}^M(v) := \left\{\pi\left(e^M(-v)\right) \mid e^M \in E^M(v)\right\} \tag{25}$$

where $e^M(-v)$ deletes exactly one node of v from each Mth-order hyperedge in e^M and $\pi(\cdot)$ represents the permutation of the remaining nodes.



e_1: $a_{121} = a_{112} = a_{211} = a_{122} = a_{212} = a_{221} = 1/3$
e_2: $a_{123} = a_{132} = a_{213} = a_{231} = a_{312} = a_{321} = 1/2$
e_3: $a_{345} = a_{354} = a_{435} = a_{453} = a_{534} = a_{543} = 1/2$

|e|	a_e
2	1/3
3	1/2

v	$\mathcal{N}^3(v)$
v_1	$\{\{\pi(v_1, v_2), \pi(v_2, v_2)\}, \{\pi(v_2, v_3)\}\}$
v_2	$\{\{\pi(v_1, v_2), \pi(v_2, v_2)\}, \{\pi(v_2, v_3)\}\}$
v_3	$\{\{\pi(v_1, v_2)\}, \{\pi(v_4, v_5)\}\}$
v_4	$\{\pi(v_3, v_5)\}$
v_5	$\{\pi(v_3, v_4)\}$

Fig. 6. (a) Hypergraph. (b) Nonzero adjacency tensor entries for the hypergraph (a). (c) Adjacency value table. (d) Neighborhood table. The parentheses in the neighborhood table represent the nodes forming a hyperedge with the target node in the first column.

Example 1 (Mth-Order Neighborhood of a Node): Consider node v_1 in Fig. 6(a) as an example. The 3rd-order incidence edge set for v_1 is $E^3(v_1) = \{\text{span}^3(e_1), \{e_2\}\} = \{\{(v_1, v_2, v_1), (v_1, v_2, v_2)\}, \{(v_1, v_2, v_3)\}\}$. Correspondingly, it follows that the 3rd-order neighborhood is $\mathcal{N}^3(v_1) = \{\{\pi(v_2, v_1), \pi(v_2, v_2)\}, \{\pi(v_2, v_3)\}\}$. Note that hyperedge (v_1, v_2, v_1) in $\text{span}^3(e_1)$ of $E^3(v_1)$ contains repeated v_1's since it results from the edge spanning, and the subsequent node deletion for generating $\mathcal{N}^M(v_1)$ should only remove one node of v_1.

From the Mth-order neighborhood definition, the neighborhood table [see, e.g., Fig. 6(d)] is constructed with every node as the first column and their Mth-order neighborhood $\mathcal{N}^M(v)$ as the second column so that it represents the hyperedge connectivity information carrying indices of nonzero adjacency entries. By specifying any target node v_i from the first column of the neighborhood table, we can quickly search for nonzero adjacency entries required in computing the shifted signal corresponding to v_i in (18) without $(M-1)$-times looping. For example, as shown in Fig. 6, the nonzero adjacency entries for v_1 with its index fixed at the first mode are $a_{1::} = \{a_{121}, a_{112}, a_{122}, a_{123}, a_{132}\}$, which is consistent with the permutations in $\mathcal{N}^3(v_1)$ from the neighborhood table. The neighborhood table together with the adjacency value table therefore forms the compressed sparse adjacency tensor to provide an efficient representation for higher order hypergraphs.

C. Tensor-Message-Passing HyperGNN

With the compressed adjacency tensor representation, we propose the algorithm called the T-MPHN in this section. The aggregation rule of the T-MPHN is formulated as follows:

$$\mathbf{m}_{\mathcal{N}^M(v)} = \underbrace{\underset{e^M \in E^M(v)}{\text{AGGREGATE}}}_{\text{different edges}} \underbrace{\left(a_e \underset{\pi(\cdot)\in\mathcal{N}^M(v)}{\text{AGGREGATE}} \underset{u\in\pi(\cdot)}{\text{CNI}} \mathbf{x}_u \right)}_{\text{interactions in an } e^M} \quad (26)$$

for any node $v \in \mathcal{V}$. The AGGREGATE denotes permutation invariant aggregation functions such as summation and average, which is chosen to be summation in our implementation, and a_e is the adjacency value computed by Theorem 1. The CNI is the CNI modeled by the product of the neighboring signals along each dimension. Specifically, let $\mathbf{x}_v \in \mathbb{R}^D$ be the input feature associated with node v. Given any sequence of nodes $\mathcal{U} = (u_1, u_2, \ldots, u_{M-1})$, define

$$\underset{u\in\mathcal{U}}{\text{CNI}}\, \mathbf{x}_u := \mathbf{x}_{u_1} \odot \mathbf{x}_{u_2} \odot \cdots \odot \mathbf{x}_{u_{M-1}} \quad (27)$$

to be the Hadamard (element-wise) product of their node features along each feature dimension d $(1 \le d \le D)$.

If the AGGREGATE function is fixed to be summation and a 1-D feature is considered (i.e., a scalar value for each node), then the aggregation rule in (26) can be written as a two-stage message-passing process

$$\begin{cases} \mathbf{m}_{e^M(v)} := \displaystyle\sum_{\pi(\cdot)\in\mathcal{N}^M(v)} \left(\prod_{u\in\pi(\cdot)} \mathbf{x}_u \right) \\ \mathbf{m}_{\mathcal{N}^M(v)} := \displaystyle\sum_{e^M\in E^M(v)} \left(a_e \mathbf{m}_{e^M(v)} \right) \end{cases} \quad (28)$$

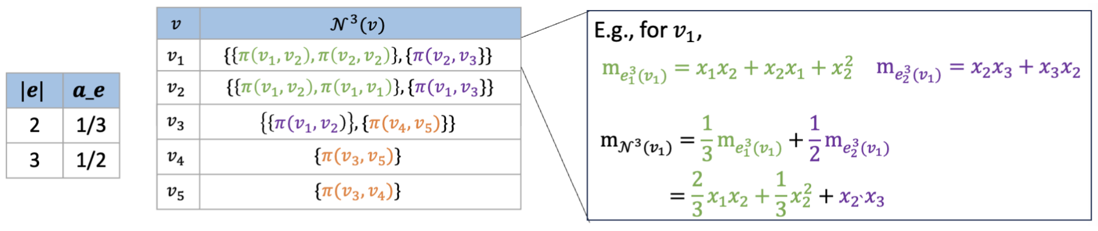
where $\mathbf{m}_{e^M(v)}$ is the Mth-order hyperedge embedding that aggregates CNIs of each permutated sequence of neighborhood nodes from $\pi(e^M(-v))$. $\mathbf{m}_{\mathcal{N}^M(v)}$ is the neighborhood embedding for node v that leverages information from different Mth-order hyperedges using the adjacency value a_e.

To better illustrate the T-message-passing rule, consider the hypergraph example in Fig. 6(a). As illustrated in Fig. 7, if we set node v_1 as the target node, its 3rd-order hyperedge embeddings are $\mathbf{m}_{e_1^3(v_1)} = x_1x_2 + x_2x_1 + x_2^2$, $\mathbf{m}_{e_2^3(v_1)} = x_2x_3 + x_3x_2$ by looking up the neighborhood table in the compressed adjacency tensor. Since the coefficients $a_{e_1} = 1/3$ and $a_{e_2} = 1/2$ can be directly retrieved from the adjacency value table, the neighborhood embedding of v_1 in the example is computed by $\mathbf{m}_{\mathcal{N}^3(v_1)} = (2/3)x_1x_2 + (1/3)x_2^2 + x_2x_3$.

Remark: From (28) and the example, one can easily see the differences between our T-message passing and the matrix-based hypergraph message passing in (5). Using the matrix-based hypergraph message passing, for a target node (e.g., node v_1), its edge embeddings $\mathbf{x}_{e_1} = x_2, \mathbf{x}_{e_2} = x_2 + x_3$, and the neighborhood embedding is further $x_2 + (x_2 + x_3)$. Compared to our higher order neighborhood embedding $\mathbf{m}_{\mathcal{N}^3(v_1)} = (2/3)x_1x_2 + (1/3)x_2^2 + x_2x_3$, the matrix-based hypergraph message-passing does not consider joint multiplicative interactions that are carried in hyperedges and does not assign weights to edge embeddings. In addition, as we can see from (26), the tensor aggregation is a function of the order M, which can be modified at each layer to create hierarchical hypergraph architectures. The significance of these three distinct components in our T-message passing has been well-established in previous works on higher order feature products [32], [33] and multilinear PageRank [20] and has been empirically verified in the ablation study in Section VI-B and the discussion about hyperparameter effects in Section VI-C.

D. Inductive Learning With T-MPHN

Based on the T-message-passing scheme proposed above, we next describe the T-MPHN forwarding algorithm, which is



|e|	a_e
2	1/3
3	1/2

v	$\mathcal{N}^3(v)$
v_1	$\{\{\pi(v_1,v_2),\pi(v_2,v_2)\},\{\pi(v_2,v_3)\}\}$
v_2	$\{\{\pi(v_1,v_2),\pi(v_1,v_1)\},\{\pi(v_1,v_3)\}\}$
v_3	$\{\{\pi(v_1,v_2)\},\{\pi(v_4,v_5)\}\}$
v_4	$\{\pi(v_3,v_5)\}$
v_5	$\{\pi(v_3,v_4)\}$

(a)

E.g., for v_1,

$$\mathrm{m}_{e_1^3(v_1)} = x_1x_2 + x_2x_1 + x_2^2 \quad \mathrm{m}_{e_2^3(v_1)} = x_2x_3 + x_3x_2$$

$$\mathrm{m}_{\mathcal{N}^3(v_1)} = \frac{1}{3}\mathrm{m}_{e_1^3(v_1)} + \frac{1}{2}\mathrm{m}_{e_2^3(v_1)} = \frac{2}{3}x_1x_2 + \frac{1}{3}x_2^2 + x_2x_3$$

(b)

Fig. 7. Illustration of the T-message passing based on the compressed adjacency tensor. (a) Compressed adjacency tensor was obtained from Section IV-B. (b) According to (28), for the given node v_1, the 3rd-order hyperedge embeddings $m_{e_1^3(v_1)}$ and $\mathbf{m}_{e_2^3(v_1)}$ are first computed by summing up the product of neighboring signals (T-message passing) according to the first step in (28). Then, the 3rd-order neighborhood embedding $m_{\mathcal{N}^3(v_1)}$ is the weighted sum of the hyperedge embeddings whose weights are the adjacency values a_e retrieved from the adjacency value table.

Algorithm 1 T-MPHN Forward Propagation

```
Input: Hypergraph $\mathcal{G}(\mathcal{V},\mathcal{E})$; node features $\{\mathbf{x}_v \mid v \in \mathcal{V}\}$; number of layers $L$; hypergraph order $M$; the adjacency value table; the neighborhood table, linear layers $\mathrm{MLP}^{(l)}, l = 1, 2, \ldots, L$; aggregation function AGGREGATE; combine operation COMBINE; nonlinear activation $\sigma$.
Output: Node embeddings $\mathbf{z}_v, \forall v \in \mathcal{V}$.
  $\mathbf{x}_v^{(0)} \leftarrow \mathrm{MLP}(\mathbf{x}_v), \forall v \in \mathcal{V}$;
  for $l = 1, \ldots, L$ do
      for $v \in \mathcal{V}$ do
          for $e^M \in E^M(v)$ do
              $\mathbf{m}^{(l)}_{e^M(v)} \leftarrow \mathrm{AGGREGATE}_{\pi(\cdot)\in\mathcal{N}^M(v)}(\mathrm{CNI}_{u\in\pi(\cdot)}\, \mathbf{x}_u^{(l-1)})$
          end for
          $\mathbf{m}^{(l)}_{\mathcal{N}^M(v)} \leftarrow \mathrm{AGGREGATE}_{e^M\in E^M(v)}(a_e \mathbf{m}^{(l)}_{e^M(v)})$
          $\mathbf{x}_v^{(l)} \leftarrow \sigma(\mathrm{MLP}^{(l)}(\mathrm{COMBINE}(\mathbf{x}_v^{(l-1)}, \mathbf{m}^{(l)}_{\mathcal{N}^M(v)})))$
          $\mathbf{x}_v^{(l)} \leftarrow \mathbf{x}_v^{(l)} / ||\mathbf{x}_v^{(l)}||_2$
      end for
  end for
  $\mathbf{z}_v \leftarrow \mathbf{x}_v^{(L)}, \forall v \in \mathcal{V}$
```

summarized in Algorithm 1. Let $\{\mathbf{x}_v \mid v \in \mathcal{V}\}$ be the input node features. To begin with, we first initialize these node features with one linear layer of regular multilayer perceptron (MLP) and project them into a latent space to obtain the initial hidden embedding features $\{\mathbf{x}_v^{(0)} \mid v \in \mathcal{V}\}$. This step is particularly helpful when the input features have very high dimensions (e.g., one-hot-encoding features) to avoid potential gradient vanishing issues.

The T-MPHN algorithm then iteratively performs L-layer neural networks as follows. For the lth layer ($l = 1, \ldots, L$), let $\{\mathbf{x}_v^{(l-1)} \mid v \in \mathcal{V}\}$ be the hidden embedding features from the previous layer. We first go through each Mth-order hyperedges e^M to compute the hidden edge features $\mathbf{m}^{(l)}_{e^M(v)}$ and then aggregate these edge features to perform the efficient two-step T-message passing to generate the neighborhood features $\mathbf{m}^{(l)}_{\mathcal{N}^M(v)}$. Subsequently, the neighborhood feature is combined with the self-node feature $\mathbf{x}_v^{(l-1)}$ from the previous step and fed into a multilinear perceptron followed by an activation function $\sigma(\cdot)$. Eventually, a normalization step is conducted to generate $\mathbf{x}_v^{(l)}$, which is used as the node's new hidden embedding features for the next layer $l + 1$. The process described above is repeated for L layers and finally leads to the output node embeddings $\mathbf{z}_v$ for all $v \in \mathcal{V}$ from the T-MPHN algorithm.

1) Inductivity of T-MPHN: For a given node, the T-MPHN algorithm can be performed by only knowing its local neighborhood and features. This makes the T-MPHN an inductive learning approach [34] that can be applied to unseen nodes and more general, dynamic hypergraphs. In Section VI-C, three different hypergraphs are constructed for training, validation, and testing, which can be seen as evolving hypergraphs at different times.

E. Design Variations of T-MPHN

Under the T-MPHN framework proposed above, it is conceivable that several variations may be formulated for practical use. We next illustrate some examples of its variations. A comprehensive investigation of other possible variations will be left to future work.

In the T-message passing of (26), one can set the hypergraph order M as a fixed value so that any hyperedge with more than M nodes will be uniformly downsampled to M degree. This downsampling strategy is especially useful for datasets with only a few extremely large edges but many small-sized edges. Furthermore, the order M of the hypergraph at different layers can be set to be different: this variation is motivated by noting that the lth layer of HyperGNNs aggregates information from the lth hop neighbors. As the aggregation propagates to neighbors that are multiple hops away from the central target node, fewer neighboring nodes may be considered. By decreasing the order M as the layer l goes deep, the model performance can often be improved, and we will provide further discussion in Section VI.

In addition to M, one may also change the aggregation function. If a dataset contains "hub" nodes that lie in many hyperedges, a normalization strategy is to set the edge AGGREGATE function in (26) to be the mean function, that is,

$$\mathbf{m}^{(l)}_{\mathcal{N}^{M(v)}} = \frac{1}{d_v} \sum_{e^M \in E^{M(v)}} \left(a_e \sum_{\pi(\cdot)\in\mathcal{N}^{M(v)}} \left(\mathrm{CNI}_{u\in\pi(\cdot)}\, \mathbf{x}_u^{(l-1)} \right) \right)$$

where d_v is the degree of node v that counts the number of edges that v lies in. Other AGGREGATE functions, such as max pooling and long short-term memory (LSTM) [34], may also be considered in accordance with a study's learning task.

F. Complexity Analysis

Unlike the T-spectral convolution that requires the use of the entire sparse adjacency tensor, the T-MPHN algorithm employs the compressed adjacency tensor to design an efficient T-message-passing scheme for a hypergraph to avoid excessive space and time complexity. Let $\delta_v = \max_{v\in\mathcal{V}} d_v$ be the maximum degree of all nodes and $D^{(l-1)}$ be the dimension of the embedding features generated from the previous layer $l-1$. Suppose that M is fixed (which is typically much smaller than N). Then, since the adjacency value table and the neighborhood table are both stored in dictionary format, the space complexity of T-MPHN is $\mathcal{O}(N\delta_v)$ and the time complexity for each layer l is $\mathcal{O}(N\delta_v+ND^{(l-1)}D^{(l)})$, which is linear in N and independent of hypergraph order M. Compared to the polynomial complexities of T-spectral convolution in Section III-D, T-MPHN is scaled to be both space and computationally efficient and practically comparable to the state-of-the-art HyperGNNs such as UniGCN [19] and HNHN [11].

V. Discussion and Connection to Related Work

Here, we first highlight connections between the three proposed HyperGNNs: T-spectral convolutional HyperGNN (T-spectral), T-spatial convolutional HyperGNN (T-spatial), and T-MPHN. Then, we show the relationship between our work and other closely related work under some special cases.

1) Connection Between T-Spectral and T-Spatial Convolutions: As shown in Section IV, the T-spatial convolution is obtained by localizing (or taking the first frontal slice of) the T-spectral convolution. Alternatively, a connection can be viewed from (38) in Algorithm 2 (see the Supplementary Material). Under the hypergraph order $M = 2$, the pre-Fourier transform and the post-Inverse Fourier transform in Algorithm E (see the Supplementary Material) can be omitted since they are applied only to orders higher than 2; the computation of T-spectral convolution then becomes the T-spatial convolution of (16). Therefore, if a hypergraph is reduced to a simple graph ($M = 2$), the T-spectral convolution is the same as the T-spatial convolution.

2) Connection Between T-Spatial Convolution and T-MPHN: The numerical computations of the T-spatial aggregation in (18) and the T-message passing in (26) are quite similar. We summarize the connection between them in Theorem 2.

Theorem 2: Given any node $v_i \in \mathcal{V}$, its aggregated feature $[\mathbf{Y}]_{i\cdot} = (y_{i1}, y_{i2}, \ldots, y_{iD})^T$ in (18) is equivalent to the neighborhood embedding $\mathbf{m}_{\mathcal{N}^M(v_i)}$ computed by (26) up to a node degree normalization factor.

Proof: See Appendix H for proof (Supplementary Material). □

In particular, if the node aggregation function and edge aggregation function in (26) are chosen to be the summation and average, the neighborhood embedding becomes exactly the same as the aggregated feature in (18), that is,

$$\mathbf{m}_{\mathcal{N}^M(v_i)} = \frac{1}{d_{v_i}} \sum_{e^M\in E^M(v)} \left(a_e \sum_{\pi(\cdot)\in\mathcal{N}^M(v)} \underset{u\in\pi(\cdot)}{\mathrm{CNI}}\, \mathbf{x}_u \right) = [\mathbf{Y}]_{i\cdot}$$

While the T-message passing performs a similar higher order aggregation as the T-spatial aggregation, the T-message passing is more general due to the flexibility of choosing the AGGREGATE functions. Aside from the aggregation function, the difference between the T-spatial convolution and the T-MPHN also lies in the way of combining the neighborhood embedding $\mathbf{m}_{\mathcal{N}^M(v_i)}$ and the central node embedding $\mathbf{x}_{v_i}$. In the former approach, if a self-loop-added adjacency tensor is used, the combining operation is restricted to summation; in the T-MPHN, the combining operation is more flexible, and we choose to use concatenation in the experiment.

3) Connection Between T-MPHN and Existing HyperGNNs: As a tensor is a generalization of a matrix, certain matrix-based HyperGNNs built on hypergraph expansions (e.g., HGNN [10] and HCHA [23]) are naturally subsumed in our work. For example, after applying clique expansion to a hypergraph $\mathcal{G}$, we obtained a uniform order-2 hypergraph, and from the definition of the adjacency tensor with $M = 2$, adjacency coefficients are $a_{ij} = 1$ for each edge $e = (i, j)$, which reduces the adjacency tensor to the adjacency matrix. For the hypergraph signal that is defined as an $(M-1)$ times outer product of the original signal $\mathbf{X} \in \mathbb{R}^{N\times D}$, it automatically becomes the same as the original signal with $M = 2$. Furthermore, using our definition of neighborhood with $M = 2$, the adjacency matrix-based neighboring aggregation rule can be written as $\mathbf{m}_{\mathcal{N}^2(v_i)} = \sum_{e^2\in E^2(v_i)} \mathbf{m}_{e^2(v_i)}$ with $\mathbf{m}_{e^2(v_i)} = \sum_{u\in e^2(-v_i)} \mathbf{x}_u$, which is a special case of the T-MPHN.

VI. Experiments

The proposed T-HyperGNNs, including the T-spectral convolution (T-spectral), the T-spatial convolution (T-spatial), and the T-MPHN, are evaluated in this section. In the first experiment, we consider transductive learning in which all nodes are involved in modeling during the training process (except for true labels of testing sets). An ablation study is conducted to show the effectiveness of using the adjacency tensor and the CNI tensor. To demonstrate the scalability and conductivity of the T-MPHN, an inductive setting is applied to four additional datasets, demonstrating broad applications in computer vision, political, and business tasks. We use the accuracy rate to be the metric for all experiments. For each reported accuracy rate, 50 repeated experiments with different seeds are performed to compute the mean and the standard deviation of the accuracy rates. We use the Adam optimizer with a learning rate and the weight decay choosing from $\{0.01, 0.001, 0.0001\}$ and $\{0.005, 0.0005, 0.00005\}$ and tune the hidden dimensions over $\{64, 128, 256, 512\}$ for all methods.

A. Transductive Node Classification

The task for transductive node classification is to predict the label associated with each node by taking the hypergraph structure and node features as input. In this experiment,

TABLE I
SUMMARY STATISTICS OF THE ACADEMIC NETWORK DATASETS

Statistic	Cocitation			Coauthorship	
	Cora	Citeseer	PubMed	Cora	DBLP
$\lvert\mathcal{V}\rvert$	83	87	89	59	65
$\lvert\mathcal{E}\rvert$	42	50	40	40	29
Feature Dimension D	1433	3703	500	1433	1425
Number of Classes	7	6	3	7	6

we consider a transductive setting [34], in which the hypergraph structure is assumed to be the same during the training and testing processes, that is, we assume that the testing node connections are known during model training.

1) Datasets: We use five standard hypergraph datasets in the academic network, which include two co-citation datasets (Cora and DBLP) and three coauthorship datasets (Cora, CiteSeer, and PubMed). The hypergraph structure is obtained by viewing each paper as a node and each co-citation or coauthor relationship as a hyperedge. The node features associated with each paper are the bag-of-words representations summarized from the abstract of each paper, and node labels are classes of papers (e.g., algorithm and computing). The raw datasets [24] are further downsampled to smaller hypergraphs such that the T-spectral and the T-spatial convolution HyeprGNNs can be applied to compare with the proposed T-MPHN. The descriptive statistics of these five hypergraphs are summarized in Table I.

2) Setup and Benchmarks: To classify the labels of testing nodes, we feed the whole hypergraph structure and node features to the model. The training, validation, and testing data are set to be 50%, 25%, and 25% for each complete dataset, respectively. Following the convention of HyperGNNs, we set the number of layers for all HyperGNNs to be 2 to avoid oversmoothing except for the T-spectral HyperGNN. For the T-spectral HyperGNN, we use only one layer because it is considered as a global approach that propagates to all nodes within just one-step T-spectral convolution. In this experiment, we choose regular MLP, clique expansion + GCN (CEGCN), HGNN [10], HyperGCN [24], and HNHN [11] as our benchmarks since these methods are originally designed for transductive settings using matrix representations. Here, CEGCN, HGNN, and HyperGCN utilize hypergraph reduction approaches to define the hypergraph adjacency matrix and Laplacian matrix such that spectral convolutions can be built up, whereas HNHN formulates a two-stage spatial propagation rule using the incidence matrix.

3) Results and Discussion: The testing results of the five academic networks are summarized in Table II. Overall, the tensor-based approaches achieve satisfactory performance compared to all the benchmarks, indicating the importance of effectively utilizing high-order tensor representation for learning hypergraphs. In particular, the T-spectral HyperGNN constructed with the t-product shows the best results on all these data examples except for the PubMed dataset. This observation coincides with our theoretical anticipation that the T-spectral model is the most robust approach as it contains the richest high-order information. Built on the localized T-spectral convolution, the T-spatial approach with only the first frontal slice of the t-product unsurprisingly shows somewhat reduced accuracy rates compared to the T-spectral approach but still achieves competitive results to the benchmarks. The T-MPHN, on the other hand, maintains very competitive results across all the datasets compared to the T-spectral approach (e.g., for the PubMed dataset, the average accuracy rate is even 7.68% higher than that of the T-spectral approach). Comparing these two proposed approaches, we tend to view the T-MPHN as the most competitive method to model various datasets and tasks; such capability is partially attributable to the concatenation of the neighborhood embedding and the central node embedding [i.e., Concat($[\mathbf{x}_v, \mathbf{m}_{\mathcal{N}^M(v)}]$)], which forms a "skip connection" between the input and the output of an aggregation step (see, e.g., GraphSAGE [34]) and, more importantly, the change of hypergraph order M at different layers.

In addition, it is worthwhile to note that the three proposed HyperGNNs themselves already demonstrate an ablation study among the full t-product, the simplified t-product (with only the first frontal slice), and the node-wise message passing with concatenation. Through the comparison between the T-spectral and the T-spatial approaches, we can see that the full t-product captures more information than only its first frontal slice; from the T-spatial approach to the T-MPHN, we can further see that such information loss can be partially compensated from the concatenation of the neighborhood embedding and the central node embedding. To gain additional insights into the model architecture of the T-MPHN, we conduct an ablation study in the next subsection to examine the adjacency value computation and the CNI.

B. Ablation Study for T-MPHN

On the same academic networks, an ablation study is designed by "turning off" the adjacency values in (23) and the CNIs in (27) separately and testing their corresponding performance. We consider three modeling scenarios: 1) the full T-MPHN model; 2) the T-MPHN model with the CNI but not the adjacency values; and 3) the T-MPHN model with the adjacency values but not the CNIs. In the second scenario, when the adjacency values are "turned off," we fill in all ones instead. In the third scenario, when the CNI is "turned off," we replace the Hadamard product of node features with their summation. The results of the ablation study are shown in Fig. 8. We can observe that the performance of the two corrupted T-MPHNs is worsened compared to the full T-MPHN, confirming the effectiveness of the adjacency values and the CNI operation.

C. Inductive Learning on Evolving Hypergraphs

In this experiment, we apply our inductive approach: T-MPHN to four real-world datasets and compare the performance against other inductive approaches developed on hypergraphs. To better adapt to practical circumstances, we assume that the hypergraphs are evolving in which unseen objects are added during testing. This setting is called inductive learning [34], as opposed to the transductive setting in the previous experiments. To create the inductive setting from our static data, we randomly reserve 25% nodes as unseen nodes

TABLE II

AVERAGED TESTING ACCURACY (%, ± STANDARD DEVIATION) ON FIVE ACADEMIC NETWORKS FOR TRANSDUCTIVE NODE CLASSIFICATION. THE TOP THREE RESULTS WITH THE HIGHEST AVERAGE ACCURACY ARE HIGHLIGHTED FOR EACH DATASET

Method	Data	Time Complexity	Cocitation			Coauthorship	
			Cora	Citeseer	Pubmed	Cora	DBLP
MLP	$\mathbf{X}$	$\mathcal{O}(NDD')$	48.23 ± 7.35	65.56 ± 1.48	73.89 ± 5.60	46.11 ± 8.35	76.15 ± 7.26
CEGCN	$(\mathbf{H}, \mathbf{X})$	$\mathcal{O}(N^2D + NDD')$	$\mathbf{72.43 \pm 8.79}$	73.22 ± 7.83	$\mathbf{92.82 \pm 5.39}$	70.26 ± 7.65	92.33 ± 5.16
HGNN	$(\mathbf{H}, \mathbf{X})$	$\mathcal{O}(N^2D + NDD')$	70.59 ± 1.22	73.89 ± 8.98	82.22 ± 1.33	66.94 ± 6.51	93.08 ± 6.39
HyperGCN	$(\mathbf{H}, \mathbf{X})$	$\mathcal{O}(N^2D + NDD' + \|\mathcal{E}\|\delta_e)$	35.29 ± 1.24	61.11 ± 1.53	76.11 ± 1.40	25.79 ± 6.43	25.38 ± 1.29
HNHN	$(\mathbf{H}, \mathbf{X})$	$\mathcal{O}(N\|\mathcal{E}\|D + \|\mathcal{E}\|DD' + NDD')$	69.41 ± 9.04	74.44 ± 9.69	77.22 ± 4.08	$\mathbf{71.39 \pm 5.56}$	93.85 ± 5.76
T-spectral	$(\mathcal{A}, \mathcal{X})$	$\mathcal{O}(N^MD + N^{(M-1)}DD')$	$\mathbf{71.59 \pm 3.43}$	$\mathbf{78.33 \pm 8.03}$	$\mathbf{86.67 \pm 1.18}$	$\mathbf{75.29 \pm 5.59}$	$\mathbf{96.10 \pm 2.16}$
T-spatial	$(\mathcal{A}, \mathcal{X})$	$\mathcal{O}(N^MD + N^{(M-1)}DD')$	69.17 ± 7.58	$\mathbf{76.11 \pm 7.05}$	84.22 ± 3.26	70.00 ± 6.01	$\mathbf{94.62 \pm 4.93}$
T-MPHN	$(\mathcal{A}, \mathcal{X})$	$\mathcal{O}(NDD' + N\delta_v)$	$\mathbf{70.83 \pm 5.59}$	$\mathbf{77.22 \pm 6.44}$	$\mathbf{93.33 \pm 4.48}$	$\mathbf{72.78 \pm 4.44}$	$\mathbf{95.38 \pm 3.10}$

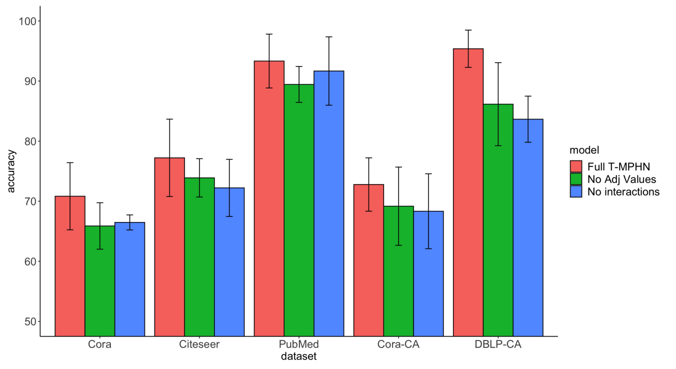


Fig. 8. Averaged accuracy of T-MPHN and its corrupted variations on the five academic networks for the ablation study.

TABLE III

SUMMARY STATISTICS OF 3-D OBJECT RECOGNITION DATASETS

Statistic	ModelNet40	NTU	House	Walmart
$\|\mathcal{V}\|$	12311	2012	1290	18032
$\|\mathcal{E}\|$	24622	4024	341	5798
Feature Dimension D	6144	6144	100	100
Number of Classes	40	67	2	10

for testing, while 50% and 25% nodes are used for regular training and validation, respectively.

1) Datasets: We employ two public 3-D object detection datasets (Princeton ModelNet40 [35] and the National Taiwan University (NTU) [36]) and other two datasets House (Politician) and Walmart (Business). On the two 3-D object detection datasets, each 3-D object is viewed as a node, and the features associated with each node are extracted using GVCNN [37] and MVCNN [38] following the experimental setting of prior work [10]. To generate the hypergraph structures for these two datasets, we follow the setup of [10] by using the K-nearest neighbor (KNN) algorithm with $K = 4$ so that all hyperedges of the constructed hypergraph have size 5. The motivation to construct hyperedges this way is to explicitly reveal higher order correlations among objects, which has been empirically shown to be a helpful strategy compared to pure vision-based models such as PointNet [39] and PointCNN [40]. We summarize the data preprocessing steps described above in Fig. 10 in Appendix K (see the Supplementary Material). The goal of the experiment is to predict the label associated with each node (e.g., window, aircraft, and shelf). For the House dataset [41], each node is a U.S. congressperson and a hyperedge is formed if a group of congresspersons put forth a bill together. Each node is labeled with political party affiliation and the goal is to classify the political party (Democratic or Republican) of each node. The last dataset Walmart [42] contains a hypergraph where nodes are an item at Walmart and hyperedges are sets of co-purchased products. Each node is associated with a department label (e.g., clothing, accessories, and pet) and the goal is to classify the products. Considering that either the House or the Walmart dataset has node features, we follow prior work [13] to set the node features as 100-D Gaussian random vectors with variance 1. The summary statistics of these four datasets are displayed in Table III.

2) Setup and Benchmarks: Since T-spectral and T-spatial HyperGNNs are not applicable to inductive settings, we only implement T-MPHN and compare its performance with the benchmark inductive methods: MLP, clique expansion with GraphSage (CEGraphSAGE) [34], HyperSAGE [15], and UniSAGE [19]. The CEGraphSAGE reduces a hypergraph to a simple graph and applies graph sampling and aggregation to the simple graph. The HyperSAGE defines the intraedge and interedge aggregations through a generalized mean function $M_p = ((1/n)\sum_{i=1}^{n} x_i^p)^{(1/p)}$, and we set $p = 0.01$ as it yields the best performance [15]. On the other hand, UniSAGE proposes a node–edge–node propagation rule using mean and summation as the aggregation functions at the first and the second layers, respectively. For all models, we construct two-layer neural networks.

3) Results: The average accuracy rates along with standard deviations are reported in Table IV. It is apparent from the table that the T-MPHN achieves consistently better results than the other benchmark methods for unseen nodes with comparable time complexity. By comparing the MLP approach against HyperGNNs, we can see that in general, taking into account an additional hypergraph representation improves model performance except for the HyperSAGE approach. A plausible reason for this could be the instability of the complex numbers caused by the generalized mean function. On the other hand, the other two simple yet effective approaches CEGraphSage

TABLE IV

Average Testing Accuracy (%, ± Standard Deviation) on Four Real-World Datasets for Inductive Node Classification. The Best Results With Respect to the Average Accuracy Are Highlighted for Each Dataset

Method	Data	Time Complexity	ModelNet40	NTU	House	Walmart
MLP	$\mathbf{X}$	$\mathcal{O}(NDD')$	88.42 ± 1.41	77.68 ± 4.46	67.25 ± 2.31	46.39 ± 3.14
CEGraphSAGE	$(\mathbf{A}, \mathbf{X})$	$\mathcal{O}(N\delta_v + NDD')$	93.32 ± 1.98	83.21 ± 1.47	70.81 ± 2.43	54.12 ± 2.81
HyperSAGE	$(\mathcal{G}, \mathbf{X})$	$\mathcal{O}(\lvert\mathcal{E}\rvert\delta_e + NDD')$	88.37 ± 2.66	75.34 ± 1.04	59.75 ± 2.09	46.64 ± 2.43
UniSAGE	$(\mathcal{G}, \mathbf{X})$	$\mathcal{O}(N\delta_v + NDD')$	92.62 ± 2.19	81.05 ± 0.82	68.64 ± 2.17	63.20 ± 1.47
T-MPHN	$(\mathcal{A}, \mathcal{X})$	$\mathcal{O}(N\delta_v + NDD')$	$\mathbf{96.69 \pm 3.22}$	$\mathbf{86.34 \pm 2.17}$	$\mathbf{72.98 \pm 2.39}$	$\mathbf{74.42 \pm 2.13}$

and UniSAGE both achieve comparable results on all datasets. A closer comparison between T-MPHN and the benchmarks shows that, on the Walmart dataset, the performance difference between the T-MPHN and the second-best approach (UniSAGE) is the largest, with the T-MPHN achieving a remarkable 17.75% accuracy improvement. This could be due to the full higher order relationship exploitation in T-MPHN. Since the features of the Walmart dataset are generated randomly, the hypergraph structure is especially important. Under such circumstances, the advantage of T-MPHN is significant.

4) Effects of Hyperparameters: While there are various hyperparameters tuned in the training process, the orders of hypergraph at each layer of the T-MPHN can be flexibly treated as hyperparameters. We find that decreasing hypergraph orders (e.g., $5 \rightarrow 3$) are generally desirable in practice. This can be viewed as a regularization of the oversmoothing problem. The first layer spreading at the first-hop neighbors of target nodes is naturally the most important one that requires a higher order, while the second layer aggregating the second-hop neighbors could use a lower order.

D. Running Times and Memory Consumption

The running times and memory consumption of all methods are available in Appendix L (see the Supplementary Material). Consistent with the time and space complexity analysis, the experiment results validate our expectations regarding the complexity hierarchy of the T-spectral, T-spatial, and T-MPHN models. Specifically, on the academic networks, T-spectral shows the highest consumption with up to 191 s/epoch and about 5000 MB of memory. T-spatial represents a mid-level complexity, requiring around 70 s and 2000 MB. Most notably, T-MPHN demonstrates a significant reduction in both time and memory usage, clocking only 0.06 s/epoch and consuming 700 MB, which is comparable with simpler models such as GCN and MLP. This significant improvement in efficiency is primarily attributed to T-MPHN's utilization of tensor sparsity. By leveraging this property, T-MPHN not only maintains the higher order tensor expensiveness but also minimizes additional time and memory requirements.

However, we observed that on larger inductive datasets such as ModelNet40, T-MPHN's consumption metrics are slightly higher, with both time and memory usage doubling in comparison to similar-complexity hypergraph networks. This increase is largely due to the dense connectivity inherent in the hypergraph structure generated via the KNN algorithm. A deeper examination revealed that the majority of T-MPHN's computation time is allocated to forward computation, with backpropagation being comparatively faster. Within the forward process, the most time-consuming step involves solving the combinatorial problem described in (25), which is essential for constructing the Mth-order neighborhood of nodes. To address this bottleneck, one of our future research directions includes operating out the neighborhood aggregation as a preprocessing step to further improve the training efficiency.

VII. Conclusion and Future Work

In this article, we introduce tensor representations of hypergraphs and present a general T-HyperGNN framework that consists of T-spectral convolutional HyperGNN, T-spatial convolutional HyperGNN, and T-MPHN. To the best of our knowledge, this is the first work using tensor representations in HyperGNNs. The advantages of this framework are threefold.

1) The proposed models benefit from hypergraph tensor descriptors. These descriptors preserve full higher order relationships without using any hypergraph reduction techniques, leading to loss-free hypergraph exploitation. In addition, the tensor representations are applicable in both spectral and spatial domains, providing a unified framework for hypergraph representation learning.
2) The T-spectral convolution fills up the sparse literature in spectral HyperGNNs by leveraging spectral filtering in HGSP.
3) The CNI tensors consider polynomial interactions of features, which enlarge receptive fields of traditional linear aggregation schemes, capturing intrinsic higher order relationships in hypergraphs.

However, with direct tensor representations, the time and space complexities are too large for some real-world applications. To address this limitation, we further propose the compressed adjacency tensor and the T-MPHN, which can efficiently handle large hypergraphs containing thousands of vertices as confirmed by the extensive numerical experiments. The empirical results on the nine real-world datasets show a very competitive performance of the proposed HyperGNNs in comparison to the other state-of-the-art benchmarks. Some issues and open questions of the T-HyperGNN framework remain to be addressed in future work:

1) Hypergraph structures studied in this work are undirected, unweighted, static (without time variation), and complete. It is interesting to consider directed, weighted, dynamic [43], and incomplete [44], [45] hypergraphs.

2) T-HyperGNNs achieve satisfactory experimental results after careful hyperparameter tuning via grid search. However, it is of great interest to see how hyperparameter adaption techniques [46] can help to optimize hyperparameters in T-HyperGNNs.
3) The current experiments have been primarily focused on node classification tasks. It would be valuable to examine the performance of T-HyperGNNs for other tasks such as hyperlink prediction [47] and hypergraph drawing [48], and other industrial applications, e.g., recommendation [5], cloud services [49], healthcare [8], [50], and language comprehension [51].
4) While we propose the compressed adjacency tensor to formulate T-MPHN, there are different approaches to reducing the complexities of the high-dimensional sparse tensors in neural networks, such as automatic sparse learning [52]. We will investigate these approaches in future work.
5) T-spectral and T-spatial convolutional HyperGNNs are based on the tensor t-product $*$ that is related to the discrete Fourier transform. Other unitary transforms, such as discrete cosine transform and Haar transform, can be considered to establish new tensor products and, thus, new hypergraph convolutions [53].

References

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Mar. 2020.

[2] S. Jia, S. Jiang, S. Zhang, M. Xu, and X. Jia, "Graph-in-graph convolutional network for hyperspectral image classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 1, pp. 1157–1171, Jan. 2024.

[3] D. Wang, B. Du, and L. Zhang, "Spectral–spatial global graph reasoning for hyperspectral image classification," *IEEE Trans. Neural Netw. Learn. Syst.*, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10114988

[4] C. Seo, K.-J. Jeong, S. Lim, and W.-Y. Shin, "SiReN: Sign-aware recommendation using graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, 2022. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9781816

[5] V. La Gatta, V. Moscato, M. Pennone, M. Postiglione, and G. Sperlí, "Music recommendation via hypergraph embedding," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 10, pp. 7887–7899, Oct. 2023.

[6] S. Munikoti, D. Agarwal, L. Das, M. Halappanavar, and B. Natarajan, "Challenges and opportunities in deep reinforcement learning with graph neural networks: A comprehensive review of algorithms and applications," *IEEE Trans. Neural Netw. Learn. Syst.*, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10161704

[7] M. M. Wolf, A. M. Klinvex, and D. M. Dunlavy, "Advantages to modeling relational data using hypergraphs versus graphs," in *Proc. IEEE High Perform. Extreme Comput. Conf. (HPEC)*, Sep. 2016, pp. 1–7.

[8] D. A. Nguyen, C. H. Nguyen, and H. Mamitsuka, "Central-smoothing hypergraph neural networks for predicting drug–drug interactions," *IEEE Trans. Neural Netw. Learn. Syst.*, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10091150

[9] C. Zheng, X. Fan, C. Wang, and J. Qi, "GMAN: A graph multi-attention network for traffic prediction," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, vol. 34, no. 1, pp. 1234–1241.

[10] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 3558–3565.

[11] Y. Dong, W. Sawin, and Y. Bengio, "HNHN: Hypergraph networks with hyperedge neurons," 2020, *arXiv:2006.12278*.

[12] M. T. Schaub, Y. Zhu, J.-B. Seby, T. M. Roddenberry, and S. Segarra, "Signal processing on higher-order networks: Livin' on the edge… And beyond," *Signal Process.*, vol. 187, Oct. 2021, Art. no. 108149.

[13] E. Chien, C. Pan, J. Peng, and O. Milenkovic, "You are AllSet: A multiset function framework for hypergraph neural networks," 2021, *arXiv:2106.13264*.

[14] C. Wan, M. Zhang, W. Hao, S. Cao, P. Li, and C. Zhang, "Principled hyperedge prediction with structural spectral features and neural networks," 2021, *arXiv:2106.04292*.

[15] D. Arya, D. K. Gupta, S. Rudinac, and M. Worring, "HyperSAGE: Generalizing inductive representation learning on hypergraphs," 2020, *arXiv:2010.04558*.

[16] M. E. Kilmer, K. Braman, N. Hao, and R. C. Hoover, "Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging," *SIAM J. Matrix Anal. Appl.*, vol. 34, no. 1, pp. 148–172, 2013.

[17] M. E. Kilmer and C. D. Martin, "Factorization strategies for third-order tensors," *Linear Algebra Appl.*, vol. 435, no. 3, pp. 641–658, Aug. 2011.

[18] K. Pena-Pena, D. L. Lau, and G. R. Arce, "T-HGSP: Hypergraph signal processing using t-product tensor decompositions," *IEEE Trans. Signal Inf. Process. over Netw.*, vol. 9, pp. 329–345, 2023.

[19] J. Huang and J. Yang, "UniGNN: A unified framework for graph and hypergraph neural networks," 2021, *arXiv:2105.00956*.

[20] D. F. Gleich, L.-H. Lim, and Y. Yu, "Multilinear PageRank," *SIAM J. Matrix Anal. Appl.*, vol. 36, no. 4, pp. 1507–1541, Jan. 2015.

[21] A. Banerjee, A. Char, and B. Mondal, "Spectra of general hypergraphs," *Linear Algebra Appl.*, vol. 518, pp. 14–30, Apr. 2017.

[22] S. Zhang, Z. Ding, and S. Cui, "Introducing hypergraph signal processing: Theoretical foundation and practical applications," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 639–660, Jan. 2020.

[23] S. Bai, F. Zhang, and P. H. S. Torr, "Hypergraph convolution and hypergraph attention," *Pattern Recognit.*, vol. 110, Feb. 2021, Art. no. 107637.

[24] N. Yadati, M. Nimishakavi, P. Yadav, V. Nitin, A. Louis, and P. Talukdar, "HyperGCN: A new method for training graph convolutional networks on hypergraphs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 1511–1522.

[25] C. Yang, R. Wang, S. Yao, and T. Abdelzaher, "Semi-supervised hypergraph node classification on hypergraph line expansion," 2020, *arXiv:2005.04843*.

[26] R. Wang, B. Fu, G. Fu, and M. Wang, "Deep & cross network for ad click predictions," in *Proc. ADKDD*, Aug. 2017, pp. 1–7.

[27] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, "XDeepFM: Combining explicit and implicit feature interactions for recommender systems," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 1754–1763.

[28] C. Ye and Y. Yang, "High-dimensional adaptive minimax sparse estimation with interactions," *IEEE Trans. Inf. Theory*, vol. 65, no. 9, pp. 5367–5379, Sep. 2019.

[29] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," 2013, *arXiv:1312.6203*.

[30] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29, 2016, pp. 3844–3852.

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

[32] A. Novikov, M. Trofimov, and I. Oseledets, "Exponential machines," 2016, *arXiv:1605.03795*.

[33] C. Hua, G. Rabusseau, and J. Tang, "High-order pooling for graph neural networks with tensor decomposition," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 6021–6033.

[34] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1024–1034.

[35] Z. Wu et al., "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1912–1920.

[36] D.-Y. Chen, X.-P. Tian, Y.-T. Shen, and M. Ouhyoung, "On visual similarity based 3D model retrieval," *Comput. Graph. Forum*, vol. 22, no. 3, pp. 223–232, Sep. 2003.

[37] Y. Feng, Z. Zhang, X. Zhao, R. Ji, and Y. Gao, "GVCNN: Group-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 264–272.

[38] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 945–953.

[39] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 652–660.

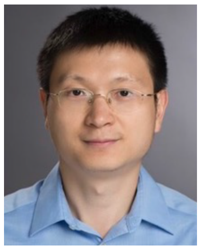
[40] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on X-transformed points," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 820–830.
[41] P. S. Chodrow, N. Veldt, and A. R. Benson, "Generative hypergraph clustering: From blockmodels to modularity," *Sci. Adv.*, vol. 7, no. 28, Jul. 2021. [Online]. Available: https://www.science.org/doi/10.1126/sciadv.abh1303
[42] I. Amburg, N. Veldt, and A. Benson, "Clustering in graphs and hypergraphs with categorical edge labels," in *Proc. Web Conf.*, Apr. 2020, pp. 706–717.
[43] X. Luo, H. Wu, Z. Wang, J. Wang, and D. Meng, "A novel approach to large-scale dynamically weighted directed network representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 12, pp. 9756–9773, Dec. 2022.
[44] X. Luo, H. Wu, and Z. Li, "Neulft: A novel approach to nonlinear canonical polyadic decomposition on high-dimensional incomplete tensors," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 6, pp. 6148–6166, Jun. 2023.
[45] D. Wu, Y. He, and X. Luo, "A graph-incorporated latent factor analysis model for high-dimensional and sparse data," *IEEE Trans. Emerg. Topics Comput.*, vol. 11, no. 4, pp. 907–917, Oct./Dec. 2023.
[46] X. Luo, Y. Yuan, S. Chen, N. Zeng, and Z. Wang, "Position-transitional particle swarm optimization-incorporated latent factor analysis," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 8, pp. 3958–3970, Aug. 2022.
[47] C. Chen and Y.-Y. Liu, "A survey on hyperlink prediction," *IEEE Trans. Neural Netw. Learn. Syst.*, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10163497
[48] M. Tiezzi, G. Ciravegna, and M. Gori, "Graph neural networks for graph drawing," *IEEE Trans. Neural Netw. Learn. Syst.*, 2022. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9810169
[49] F. Bi, T. He, Y. Xie, and X. Luo, "Two-stream graph convolutional network-incorporated latent feature analysis," *IEEE Trans. Services Comput.*, vol. 16, no. 4, pp. 3027–3042, Jul./Aug. 2023.
[50] X.-A. Bi, Y. Wang, S. Luo, K. Chen, Z. Xing, and L. Xu, "Hypergraph structural information aggregation generative adversarial networks for diagnosis and pathogenetic factors identification of Alzheimer's disease with imaging genetic data," *IEEE Trans. Neural Netw. Learn. Syst.*, 2022. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9925992
[51] X. Sun, F. Yao, and C. Ding, "Modeling high-order relationships: Brain-inspired hypergraph-induced multimodal-multitask framework for semantic comprehension," *IEEE Trans. Neural Netw. Learn. Syst.*, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10068184
[52] Z. Tang et al., "Automatic sparse connectivity learning for neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 10, pp. 7350–7364, Oct. 2023.
[53] M. Li, Z. Ma, Y. G. Wang, and X. Zhuang, "Fast Haar transforms for graph neural networks," *Neural Netw.*, vol. 128, pp. 188–198, Aug. 2020.

Fuli Wang (Student Member, IEEE) received the B.Sc. degree in finance from Dongbei University of Finance and Economics, Dalian, China, in 2018, and the M.Sc. degree in statistics from the University of Minnesota, Minneapolis, MN, USA, in 2020. She is currently pursuing the Ph.D. degree in financial services analytics with the University of Delaware, Newark, DE, USA.

Her research interests include graph neural networks, graph signal processing, anomaly detection, and applications in finance and business.

Karelia Pena-Pena (Student Member, IEEE) received the B.Sc. degree in electrical engineering from the Universidad de Los Andes, Mérida, Venezuela, in 2017, and the M.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Delaware, Newark, DE, USA, in 2020 and 2023, respectively.

Her research interests include graph signal processing, natural language processing, computer vision, machine learning, and optimization.



Wei Qian received the Ph.D. degree in statistics from the University of Minnesota, Minneapolis, MN, USA, in 2014.

He joined the School of Mathematics and Statistics, Rochester Institute of Technology, as an Assistant Professor in 2014 and then moved to the Department of Applied Economics and Statistics, University of Delaware, Newark, DE, USA, in 2017, where he has been an Associate Professor since 2021. His research interests include high-dimensional statistics, model selection, dimension reduction, statistical computing, deep learning, reinforcement learning, and data science applications.

Dr. Qian currently serves as a fellow for JP Morgan Chase (JPMC), an affiliated member of the Institute of Financial Services Analytics, and the Graduate Director of Statistics Programs at the University of Delaware.

Gonzalo R. Arce (Life Fellow, IEEE) is currently the Charles Black Evans Distinguished Professor of Electrical and Computer Engineering and a J. P. Morgan-Chase Senior Faculty Fellow with the Institute of Financial Services Analytics, University of Delaware, Newark, DE, USA. He held the 2010 and 2017 Fulbright-Nokia Distinguished Chair of Information and Communications Technologies with Aalto University, Espoo, Finland. He holds 25 U.S. patents and is the coauthor of four books. His research interests include computational imaging, data science, and machine learning.

Prof. Arce was an elected fellow of OPTICA, Society of Photo-Optical Instrumentation Engineers (SPIE), Asia-Pacific Artificial Intelligence Association (AAIA), and the National Academy of Inventors (NAI). He received the NSF Research Initiation Award.