

T-HyperGNNs: Hypergraph Neural Networks Via Tensor Representations

This paper was downloaded from TechRxiv (<https://www.techrxiv.org>).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

31-01-2023 / 05-02-2023

CITATION

Wang, Fuli; Pena-Pena, Karelia; Qia, Wei; Arce, Gonzalo (2023): T-HyperGNNs: Hypergraph Neural Networks Via Tensor Representations. TechRxiv. Preprint. <https://doi.org/10.36227/techrxiv.21984797.v1>

DOI

[10.36227/techrxiv.21984797.v1](https://doi.org/10.36227/techrxiv.21984797.v1)

T-HyperGNNs: Hypergraph Neural Networks Via Tensor Representations

Fuli Wang, *Member, IEEE*, Karelia Pena-Pena, *Member, IEEE*, Wei Qian, 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 paper, we depart from hypergraph matrix representations and present a new tensor-HyperGNN framework (T-HyperGNN) with cross-node interactions. 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 cross-node interaction layer. These advantages of our T-HyperGNNs are demonstrated in a wide range of real-world hypergraph datasets.

Index Terms—Hypergraphs, neural networks, tensors, convolution, message passing

I. INTRODUCTION

MACHINE learning on graphs has drawn much attention in the last few years as graphs can represent non-Euclidean relations in data. Graph neural networks (GNNs), in particular, have shown promise in various domains, such as social networks [1], [2], computer vision [3], [4], knowledge graphs [5], [6], and anomaly detection [7]. 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, regular GNNs become insufficient to capture all useful features [8]. For example, biomedical

reactions often contain more than two substances [9], a co-authorship network can involve more than two authors for each paper [10], and traffic flows may be determined by more than two locations [11]. 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, c)).

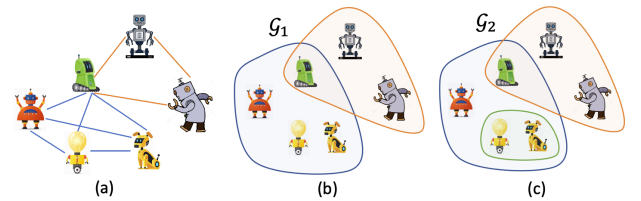


Fig. 1. Robot collaboration network represented by (a) a simple graph and (b) a hypergraph \mathcal{G}_1 and (c) another hypergraph \mathcal{G}_2 . In (a), each cooperation relationship is denoted by a line connecting exactly two entities; whereas in (b) and (c), each hyperedge denoted by a colored ellipse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

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 [12] and star expansion [13], among several others [14]. Since the graph convolution operation is originally derived in the spectral domain [12], we call them spectral 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 [15], [16]. 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 HNNH [13] and HyperSAGE [17], 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 matrix-based hypergraph representations. In addition, neither the spectral nor the spatial HyperGNNs use higher-order interactions among nodes while decomposing hypergraph information mainly through matrix representations.

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) [18] were introduced to better understand hypergraph operations such as signal shifting and spectral filtering, thus offering powerful tools to formulate spectral convolutions [19], [20]. Given these tensor representations

F. Wang is with the Institute for Financial Services Analytics, University of Delaware, Newark, DE, 19716, USA

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

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

This work was supported by the National Science Foundation under grants 1815992, 1816003, and 1916376, the AFOSR award FA9550-22-1-0362 and by the Institute of Financial Services Analytics, sponsored by JP Morgan Chase & Co.

Manuscript received January 19, 2023; revised August 16, 2023.

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? (3) Is it possible to generalize common graph neural network architectures, such as spectral convolution, spatial convolution, and message passing under the tensorial setting? 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 framework. For simplicity, We call this new framework **T-HyperGNNs**. The contribution of this paper includes the following main aspects:

- We design HyperGNNs via tensor representations to make use of higher-dimensional data in hypergraph representation learning. We encode hypergraph structures in adjacency tensors and model mutual interaction among nodes via cross-node interaction tensors, which allow a HyperGNN to learn higher-order functions beyond node-wise summation.
- From hypergraph tensor representations and cross-node interaction tensors, we formulate T-spectral convolutions under the t-product scheme that connects to loss-free hypergraph Fourier transforms [20]. Since the T-spectral convolution is a global operation, we localize the T-spectral convolution to form the T-spatial convolution.
- To address the scalability and inductivity of tensor-based convolutions, we further propose tensor message-passing hypergraph neural networks (T-MPHNs) by storing and computing the adjacency tensor in a compressed manner (which is referred to as the compressed adjacency tensor).
- The proposed T-HyperGNNs show promising performance in comparison to state-of-the-art benchmarks over a wide range of datasets. The T-MPHN, in particular, is capable of processing large hypergraphs with efficient space and computational complexity comparable to matrix representation-based HyperGNNs.

The rest of this paper is organized as follows. We introduce the necessary background and related work in Section II. We then define the cross-node interaction 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 tensor message passing neural network (T-MPHN). Connections between our three methods and other existing HyperGNNs are illustrated in Section V. The numerical experiments are summarized in Section VI. And a brief conclusion is given in Section VII.

II. BACKGROUND AND RELATED WORK

A. Hypergraph and Algebraic Descriptors

A hypergraph \mathcal{G} is defined as a pair of two sets $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, \dots, v_N\}$ denotes the set of N nodes (or vertices) and $\mathcal{E} = \{e_1, e_2, \dots, e_K\}$ is the set of K hyperedges whose elements e_k ($k = 1, 2, \dots, K$) 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 $\mathbf{X} \in \mathbb{R}^{N \times D}$ of a hypergraph.

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 representation, a hypergraph is 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 straightforward, it is a rectangular operator without a dimension-preserving property. 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)) that causes distortion of hypergraph structures [15], [16].

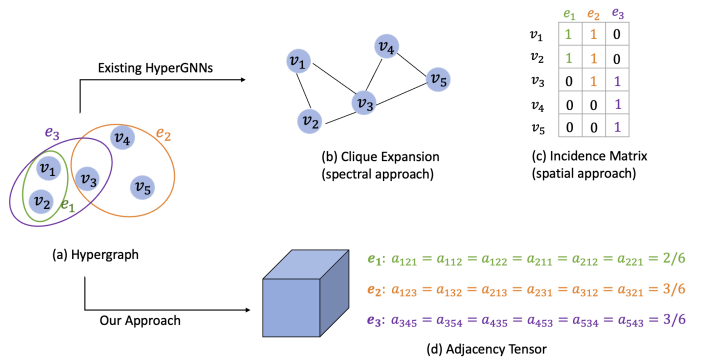


Fig. 2. (a) A hypergraph with (b) its clique expansion that is used in spectral HyperGNNs, (c) incidence matrix is utilized in spatial HyperGNNs, and (d) adjacency tensor, where nonzero entries of the adjacency tensor are specified on the right-hand side.

In this work, we use tensor-based descriptors to propose a novel tensor-hypergraph neural network (T-HyperGNN) framework. Given a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with N nodes of M^{th} -order (that is, $m.c.e(\mathcal{G}) = M$), its **adjacency tensor** is defined as an M^{th} -order N -dimensional tensor $\mathcal{A} \in \mathbb{R}^{N^M}$. Specifically, for any hyperedge $e_k = \{v_{k_1}, v_{k_2}, \dots, v_{k_c}\} \in \mathcal{E}$ with $c = |e_k| \leq M$, the tensor's corresponding entries are given by

$$a_{p_1 p_2 \dots p_M} = \frac{c}{\alpha}, \quad (1)$$

with

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

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 $r_1, r_2, \dots, r_c \neq 0$. In addition, other entries not associated with any hyperedge are all zeros. The example below demonstrates this definition. We will revisit and explain this definition in Section IV when we define the compressed adjacency tensor.

Example 2.1: Given the hypergraph in Fig. 2 (a), its three hyperedges e_1, e_2, e_3 are represented by the adjacency cube in

Fig. 2 (d) with nonzero entries specified on the right-hand side. For e_1 that contains nodes v_1 and v_2 , the adjacency indices are assigned according to their length-3 permutations $\{(121), (112), (211), (122), (212), (221)\}$, where 1 and 2 appear for at least once. Then, the values of adjacency coefficients are computed as $2/6$, where the numerator $c = 2$ is the degree of e_1 , and the denominator $\alpha = 6$ is the number of index permutations. For the other two hyperedges e_2, e_3 with $c = |e_2| = |e_3| = M = 3$, the indices of their corresponding adjacency entries are direct permutations of their node indices (e.g., $\{(345), (354), (435), (453), (534), (543)\}$ for edge e_3). And the numerical value is computed by the quotient of hyperedge cardinality and the number of index permutations, giving the coefficient value $3/6$.

B. Problem Definition

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 $\mathcal{T} = \{(\mathbf{x}_1, t_1), \dots, (\mathbf{x}_{|\mathcal{T}|}, t_{|\mathcal{T}|})\}$ with the observed training targets $\mathbf{t} = (t_1, \dots, t_{|\mathcal{T}|})$. The learned map is then $\Phi(\mathbf{X}, \mathcal{S}, \mathcal{W}^*)$ with

$$\mathcal{W}^* = \underset{\mathcal{W}}{\operatorname{argmin}} J(\Phi(\mathbf{X}, \mathcal{S}, \mathcal{W}), \mathbf{t}). \quad (3)$$

The cost function can be chosen based on downstream tasks (e.g., node classification [21]).

C. Related Work

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.

Matrix-based Spectral HyperGNNs. The earliest attempt to build HyperGNNs includes HGNN [12] and HCHA [22], 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. After applying spectral filters, the spectral convolution is formulated as $\mathbf{Z} = \mathbf{A}^{norm} \mathbf{X} \mathbf{W}$, where $\mathbf{A}^{norm} \in \mathbb{R}^{N \times N}$ 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-level and edge-level ambiguities [16]. Other methods such as HyperGCN [10]

and LEGCN [23] are developed following similar ideas with different variants of matrix descriptors.

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 [24] proposes a spatial message-passing process with two steps:

$$\begin{cases} \mathbf{x}_e = \phi_1(\{\mathbf{x}_u\}_{u \in e}) \\ \mathbf{z}_v = \phi_2(\mathbf{x}_v, \{\mathbf{x}_e\}_{e \in E_v}) \end{cases}, \quad (4)$$

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 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 HNNH [13], HyperSAGE [17], and AllSet [15] 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, spectral HyperGNNs require a dimension-preserving hypergraph descriptor in order to define the spectral space, and current methods are mostly focused on matrix representations that correspond to hypergraph reductions. On the other hand, spatial HyperGNNs with the node-edge-node aggregation process stem from spectral convolution but are implemented in a two-step manner to adopt deep learning techniques. 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 take into account possible high-order feature interactions among multiple nodes. Indeed, the salient characteristic of hypergraphs compared to simple graphs is that hyperedges depict joint effects of a group of nodes. Lastly, 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 signal tensor by modeling cross-node interactions. 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 (T-MPHNs) are then built upon

the compressed adjacency tensor to address tensor complexity for developing computationally efficient algorithms.

D. Tensor Notations and Operations

For ease of presentation, we first describe the notation for 3rd order tensors as 3rd order tensors are the base case of higher-order tensors. For a 3rd order tensor, indices $i \in \{1, 2, \dots, N_1\}, j \in \{1, 2, \dots, N_2\}, k \in \{1, 2, \dots, N_3\}$ 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). The k^{th} frontal slice is $\mathbf{A}^{(k)} = \mathbf{A}(:, :, k) \in \mathbb{R}^{N_1 \times N_2 \times 1}$. When it comes

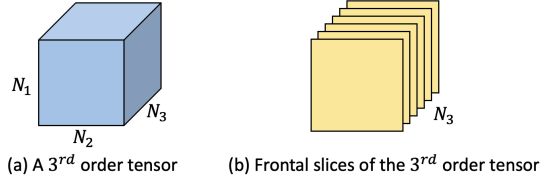


Fig. 3. (a) Third-order tensor $\mathcal{A} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$; (b) N_3 frontal slices $\mathbf{A}^{(k)} = \mathcal{A}(:, :, k) \in \mathbb{R}^{N_1 \times N_2 \times 1}$.

to M^{th} -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 $N'_3 = N_3 N_4 \cdots N_M$.

III. T-SPECTRAL CONVOLUTION ON HYPERGRAPHS

In this section, we introduce the hypergraph interaction tensor by modeling cross-node interactions. Then based on the hypergraph adjacency tensor and the hypergraph interaction tensor, we propose the hypergraph T-spectral convolution using t-products.

A. Modeling Cross-node Interactions

To begin with, we present a cross-node interaction (CNI) tensor to model higher-order interactions among nodes. The CNI is designed as the $(M-1)$ -time outer product of features along each feature dimension. Given the feature (or signal) matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ as the input, with N being the number of nodes in a hypergraph and D being the dimension of features for each node, the d -th dimensional interaction among all nodes ($d = 1, \dots, D$) is given by

$$CNI([\mathbf{x}]_d) = \underbrace{[\mathbf{x}]_d \circ [\mathbf{x}]_d \circ \cdots \circ [\mathbf{x}]_d}_{(M-1) \text{ times}} \in \mathbb{R}^{N \times 1 \times N^{(M-2)}}, \quad (5)$$

where \circ denotes the outer product (also known as elementary tensor product), and $[\mathbf{x}]_d \in \mathbb{R}^N$ represents the d -th dimensional feature vector of all N nodes. For example, given $M = 3$, $CNI([\mathbf{x}]_d) = [\mathbf{x}]_d [\mathbf{x}]_d^T \in \mathbb{R}^{N \times 1 \times N}$. Here we unsqueeze the outer-product tensor to generate the additional second mode for the dimension index of different features. Then by computing $CNI([\mathbf{x}]_d)$ for all D features and stacking them together along the second-order dimension, we obtain an M^{th} -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 cross-node interaction

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; 3) The order of interactions grows naturally with increasing order of complexity of the hypergraph.

Although widely used in many applications such as recommendation systems (e.g., Deep & Cross Network (DCN) [25], eXtreme Deep Factorization Machine (xDeepFM) [26]), interactions are mostly defined to be cross-channel or cross-attribute for a node. Cross-channel interactions are also well-known in high dimensional regression [27]. Here we design the interactions to be cross-node (as opposed to cross-channel) based on the intrinsic node interactions depicted in hyperedges, which could contain additional information across linked nodes beyond linear summations of individual nodes.

B. Hypergraph T-spectral Convolution

The hypergraph T-spectral convolution is inspired by the tensor operations in the hypergraph signal processing framework known as t-HGSP [20], where the hypergraph spectrum is defined via t-product decompositions [19]. With the adjacency tensor \mathcal{A} and the cross-node interaction tensor \mathcal{X} , we first enlarge them to their symmetric version $\mathcal{A}_s \in \mathbb{R}^{N \times N \times N_s^{(M-2)}}$ and $\mathcal{X}_s \in \mathbb{R}^{N \times D \times N_s^{(M-2)}}$, where $N_s = (2N + 1)$ according to the symmetrization operation in Appendix B. The motivation of symmetrizing tensors is to obtain a symmetric block circulant matrix like $bcirc(\mathcal{A}_s)$ (to be introduced in Eq. (9)) so as to allow proper alignment of the adjacency tensor and the CNI feature tensor. After the symmetrization, to further ensure bounded spectra of the adjacency tensor, we normalize adjacency entries of \mathcal{A}_s by scaling them using the degrees of relevant nodes to generate \mathcal{A}_s^{norm} . Since the symmetrization and the normalization are both known operations for tensors, for brevity, we leave their detailed description to Appendix B and Appendix C, respectively. The **T-spectral convolution** is then formulated as

$$\mathcal{Z}_s = \mathcal{A}_s^{norm} * \mathcal{X}_s * \mathcal{W}_s, \quad (6)$$

where $\mathcal{W}_s \in \mathbb{R}^{D \times D' \times N_s^{(M-2)}}$ is a learnable weight tensor with DD' weights parameterized in the first frontal slice and all the remaining frontal slices being zeros. The multiplication operation $*$ denotes the tensor t-product [20]. Specifically, for the 3rd order case ($M = 3$), given $\mathcal{A}_s \in \mathbb{R}^{N \times N \times N_s}$ and $\mathcal{X}_s \in \mathbb{R}^{N \times D \times N_s}$, we have

$$\mathcal{A}_s * \mathcal{X}_s \quad (7)$$

$$= fold(bcirc(\mathcal{A}_s) \cdot unfold(\mathcal{X}_s)) \quad (8)$$

$$= fold \left(\begin{bmatrix} \mathbf{0} & \mathbf{A}^{(1)} & \mathbf{A}^{(2)} & \cdots & \mathbf{A}^{(2)} & \mathbf{A}^{(1)} \\ \mathbf{A}^{(1)} & \mathbf{0} & \mathbf{A}^{(1)} & \cdots & \mathbf{A}^{(3)} & \mathbf{A}^{(2)} \\ \mathbf{A}^{(2)} & \mathbf{A}^{(1)} & \mathbf{0} & \cdots & \mathbf{A}^{(4)} & \mathbf{A}^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{A}^{(2)} & \mathbf{A}^{(3)} & \mathbf{A}^{(4)} & \cdots & \mathbf{0} & \mathbf{A}^{(1)} \\ \mathbf{A}^{(1)} & \mathbf{A}^{(2)} & \mathbf{A}^{(3)} & \cdots & \mathbf{A}^{(1)} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{X}^{(1)} \\ \mathbf{X}^{(2)} \\ \vdots \\ \mathbf{X}^{(2)} \\ \mathbf{X}^{(1)} \end{bmatrix} \right), \quad (9)$$

where the operator $bcirc(\mathcal{A}_s)$ 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 of $bcirc(\mathcal{A}_s)$ 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 is simply the one-step cyclic shifting of the previous row. The operation $unfold(\mathcal{X}_s)$ stacks vertically the set of N_s frontal slice matrices (in $\mathbb{R}^{N \times D}$) of \mathcal{X}_s into a $N_s N \times D$ matrix. The operator $fold()$ is the reverse of the $unfold()$ process so that $fold(unfold(\mathcal{A}_s)) = \mathcal{A}_s$. The t-product of higher order tensors is more involved with recursive computation with 3rd order base cases. To maintain presentation brevity here, the details of this known t-product procedure are relegated to Appendix A for technical completeness.

The reason for constructing the T-spectral convolution as we defined above in Eq. (6) is partly due to the connection between the t-product and the Fourier transform [28]. Since circulant matrices are diagonalized by the discrete Fourier transform, as shown in Algorithm 1, the t-product above can be efficiently computed by recursively applying the Fast Fourier Transform to both tensors, followed by conducting regular matrix product between flattened tensors and eventually performing Inverse Fast Fourier Transform. The derivation of the

Algorithm 1 T-spectral Convolution

Input: Input: Shifting Operator $\mathcal{A}_s^{norm} \in \mathbb{R}^{N \times N \times N_s^{(M-2)}}$; hypergraph signal tensor $\mathcal{X}_s \in \mathbb{R}^{N \times D \times N_s^{(M-2)}}$; weight matrix $\mathbf{W} \in \mathbb{R}^{D \times D'}$.

Output: New hypergraph signal $\mathcal{Z}_s \in \mathbb{R}^{N \times N \times N_s^{(M-2)}}$.

for $p = 3, \dots, M$ **do**

$\mathcal{A}_s^{norm} \leftarrow \text{fft}(\mathcal{A}_s^{norm}, \text{mode} = p)$,

$\mathcal{X}_s \leftarrow \text{fft}(\mathcal{X}_s, \text{mode} = p)$

end for

for $k = 1, 2, \dots, N_s^{(M-2)}$ **do**

$$\mathcal{Z}_s \leftarrow \mathcal{A}_s^{norm}(:, :, k) \cdot \mathcal{X}_s(:, :, k) \cdot \mathbf{W} \quad (10)$$

end for

for $p = M, \dots, 3$ **do**

$\mathcal{Z}_s \leftarrow \text{ifft}(\mathcal{Z}_s, \text{mode} = p)$

end for

T-spectral convolution is drawn from the t-eigendecomposition of the Laplacian tensor, and we include the technical details in Appendix H. After constructing the spectral space from the t-eigendecomposition of the Laplacian tensor, a filtering function is applied to the frequency of the hypergraph, i.e., the eigen-tuples of the Laplacian tensor. When the filtering function is defined as the commonly-used first-order Chebyshev polynomial [29], [30], the T-spectral convolution in Eq. (6) is obtained.

C. 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 $\mathcal{A}_s * \mathcal{X}_s * \mathcal{W}_s$ in Eq. (6) can be shown to be $\mathcal{O}(DN^{2M})$, which is practically difficult

given any moderate M . Even though the computation of the t-product could be reduced to $\mathcal{O}(DN^M)$ using Algorithm 1, it is still not sufficiently fast in large hypergraph learning. In addition, considering the space complexity of a M^{th} -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 model training process, a direct implementation is usually not feasible for large hypergraphs. We address these limitations in the next section.

IV. T-SPATIAL HYPERGRAPH NEURAL NETWORKS

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 nodes. Second, to alleviate the space complexity of tensors, we introduce the compressed adjacency tensor that takes little memory usage. Based on 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 are propagated through during a shifting operation.

To this end, recall that $\mathcal{A}^{norm} \in \mathbb{R}^{N \times N \times N^{(M-2)}}$ is the (normalized) adjacency tensor defined from Eq. (1), and $\mathcal{X} \in \mathbb{R}^{N \times D \times N^{(M-2)}}$ is the CNL signal tensor as defined in Eq. (5) from the feature matrix \mathbf{X} . Here, we view the last $(M-2)$ orders of these tensors as indices along the third order so that matrices $\mathbf{A}^{(k)} \in \mathbb{R}^{N \times N}$ and $\mathbf{X}^{(k)} \in \mathbb{R}^{N \times D}$ represent the frontal slices ($k = 1, 2, \dots, N^{(M-2)}$) of \mathcal{A}^{norm} and \mathcal{X} , respectively. After applying the 3rd-order symmetrization to \mathcal{A}^{norm} and \mathcal{X} according to Eq. (32) in Appendix B, the **T-spatial convolution** is defined as

$$\mathbf{Z}_G = (\mathcal{A}_s^{norm} * \mathcal{X}_s)^{(1)} \mathbf{W}, \quad (11)$$

where \mathcal{A}_s^{norm} and \mathcal{X}_s are the corresponding symmetrized tensors, $(\mathcal{A}_s^{norm} * \mathcal{X}_s)^{(1)}$ is the first frontal slice of the shifted hypergraph signal $\mathcal{A}_s^{norm} * \mathcal{X}_s$, and $\mathbf{W} \in \mathbb{R}^{D \times D'}$ is a learnable weight matrix. In this sense, the form of T-spatial convolution defined above can be viewed as a localized version of a T-spectral convolution as it keeps only the first frontal slice of the form of Eq. (6).

Also, note that $(\mathcal{A}_s^{norm} * \mathcal{X}_s)^{(1)}$ can be computed as the sum of the corresponding frontal-slice products between \mathcal{A}^{norm} and \mathcal{X} , that is,

$$\mathbf{Y} := (\mathcal{A}_s^{norm} * \mathcal{X}_s)^{(1)} = \sum_{k=1}^{N^{M-2}} \mathbf{A}^{(k)} \mathbf{X}^{(k)}. \quad (12)$$

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

$$y_{id} = \sum_{j=1}^N \sum_{i_3=1}^N \cdots \sum_{i_M=1}^N a_{ji i_3 \dots i_M} x_{j d i_3 \dots i_M}, \quad (13)$$

where $x_{jd i_3 \dots i_M} = x_{jd} x_{i_3 d} \dots x_{i_M d}$.

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 ($i = 1, \dots, N$). Then if three different nodes v_i, v_j, 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.

In general, by the adjacency tensor definition and its sparse nature, the entries $a_{i j_1 j_2 \dots j_{M-1}}$ are the indicators to determine whether a corresponding set of nodes is connected to the target node v_i through a hyperedge. Therefore, Eq. (13) 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 the next subsections.

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 $\mathcal{A}_s^{norm} * \mathcal{X}_s$ (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 convolution discussed above are summarized in the following propositions.

Proposition 4.1: The T-spatial convolution is localized that propagates only through neighbors of each target node.

Proposition 4.2: The T-spatial convolution is permutation invariant on the ordering of the nodes.

Proof. See Appendix D for the proof of Proposition 4.1 and 4.2. \square

B. Compressed Adjacency Tensor Representation

If the use of the T-spatial convolution had to require complete construction and loading of large tensors, the time and space complexity would remain too large for most applications. In the following, to avoid direct tensor constructions, we decompose the adjacency tensor into two tables: the adjacency value table and the neighborhood table. These two tables will play an important role in formulating a **T-spatial message passing** algorithmic framework called **T-MPHN** in Section IV-C. Since the T-MPHN algorithm is designed from the T-spatial convolution based on the adjacency tensor and the cross-node interaction tensor, it remains to be a tensor-based approach. Nonetheless, T-MPHN can be shown to require much less computing time than that of using the T-spatial convolution by taking advantage of the sparsity of the adjacency tensor to store adjacency values and node connectivity in a compressed manner.

Returning to the hypergraph adjacency tensor introduced in Section II-A, from Example 2.1, we can see that the construction of the adjacency tensor can be divided into two sequential steps: 1) Spanning every edge into M^{th} -order hyperedge; 2) Permutating indices of each spanned M^{th} -order hyperedges.

Step 1. Spanning every edge $e \in \mathcal{E}$ into M^{th} -order hyperedges: Since hyperedges with $|e| = M$ are in M^{th} -order already, only hyperedges with $|e| < M$ need to be spanned.

Definition 1 (M^{th} -order Hyperedge): Given a hypergraph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with the order M , for any hyperedge $e \in \mathcal{E}$, its M^{th} -order hyperedge set e^M is given by

$$e^M = \begin{cases} \{e\}, & \text{if } |e| = M, \\ \text{span}^M(e), & \text{if } |e| < M. \end{cases} \quad (14)$$

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

$$\text{span}^M(e) = \{e' \mid \text{unique}(e') = e, |e'| = M\}, \quad (15)$$

where $\text{unique}(e') = e$ means the distinct elements in e' is the same as e , and $|e'|$ is the number of (possibly non-unique) 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 :

$$|\text{span}^M(e)| = \mathcal{C}^R(|e|, (M - |e|)) = \binom{(M - 1) + |e|}{|e|}. \quad (16)$$

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 $e'_{11} = (v_1, v_2, v_1)$ and $e'_{12} = (v_1, v_2, v_2)$ as shown in Fig. 4.

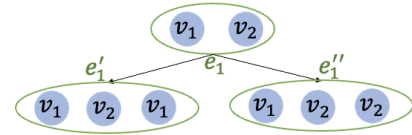


Fig. 4. Spanning the hyperedge e_1 in Fig. 2(a) with $|e_1| = 2 < M = 3$ to M^{th} -order sub-hyperedges.

Step 2. Permutating M^{th} -order hyperedges: After obtaining M^{th} -order hyperedges e^M for every $e \in \mathcal{E}$, we permute 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 $(p_1, p_2, \dots, p_M) \in \pi(e^M)$, the entry value in Eq. (1) can be equivalently written as

$$a_{p_1 p_2 \dots p_M} = \frac{|e|}{|\pi(e^M)|}, \quad (17)$$

where the cardinality of permuted M^{th} -order hyperedge $|\pi(e^M)| = \alpha$ is given in Eq. (2). As we can see from Eq. (17), two types of information are associated with nonzero adjacency entries: the adjacency value corresponding to the hyperedge e , and the indices capturing node connectivity. We then introduce two lookup tables to encode the information of the adjacency tensor: the adjacency value table and the node neighborhood table. These tables represent the compressed adjacency tensor, and an illustrative example is shown in Fig. 5.

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 4.1: 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

$$\alpha = \sum_{i=0}^{|e|} (-1)^i \binom{|e|}{i} (|e| - i)^M. \quad (18)$$

Proof. The proof is given in Appendix E. \square

Given Theorem 4.1, 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 Eq. (18). 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. So the adjacency table as shown in Fig. 5 (c) is typically very short.

Next for the node neighborhood table, we introduce the concept of M^{th} -order neighborhood of a node.

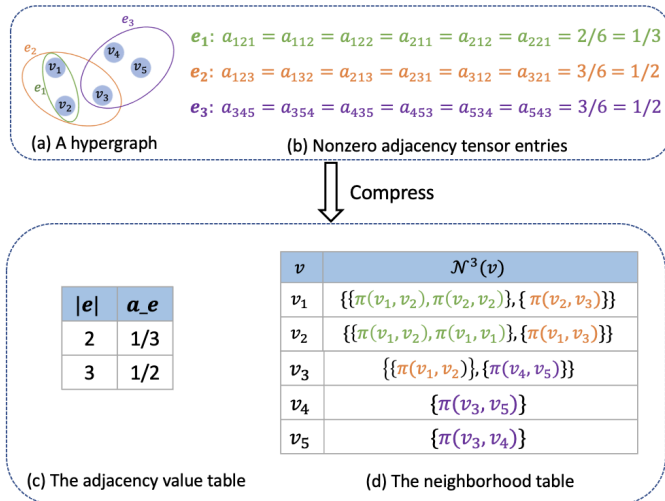


Fig. 5. (a) A hypergraph; (b) The nonzero adjacency tensor entries for the hypergraph (a); (c) the adjacency value table; (d) the neighborhood table. The parentheses in the neighborhood table represent the nodes forming a hyperedge with the target node in the first column.

Definition 2 (M^{th} -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 M^{th} -order incidence edge set is

$$E^M(v) := \{e^M \mid e \in \mathcal{E}, v \in e\}, \quad (19)$$

where e^M is the M^{th} -order hyperedge set defined in Eq. (14). Then we can define the M^{th} -order neighborhood of v that basically excludes one target node v in each hyperedge from $E^M(v)$:

$$\mathcal{N}^M(v) := \{\pi(e^M(-v)) \mid e^M \in E^M(v)\}, \quad (20)$$

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

Consider node v_1 in Fig. 2(c) 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, the M^{th} -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 M^{th} -order neighborhood definition, the neighborhood table (see, e.g., Fig. 5(c)) is constructed with every node as the first column and their M^{th} -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 Eq. (13). For example, as shown in Fig. 5, the nonzero adjacency entries for v_1 with its index fixed at the first node 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 hypergraph.

C. Inductive Learning with T-MPHN

With the compressed adjacency tensor representation, we propose the algorithm called the tensor message-passing hypergraph neural network (T-MPHN) in this subsection. Given any node $v \in \mathcal{V}$, let $\mathbf{x}_v \in \mathbb{R}^D$ be the input feature associated with node v . Given any ordered sequence of nodes $\mathcal{U} = (u_1, u_2, \dots, u_{M-1})$, define

$$\text{CNI } \mathbf{x}_{\mathcal{U}} := \mathbf{x}_{u_1} \odot \mathbf{x}_{u_2} \odot \dots \odot \mathbf{x}_{u_{M-1}} \quad (21)$$

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

From the node-wise perspective, we then use the M^{th} -order neighborhood $\mathcal{N}^M(v)$ defined from Eq. (19) and Eq. (20) to compute the neighborhood embedding $\mathbf{m}_{\mathcal{N}^M(v)}$ of node v

$$\mathbf{m}_{\mathcal{N}^M(v)} := \text{AGGREGATE}_{e^M \in E^M(v)}(a_e \mathbf{m}_{e^M(v)}), \quad (22)$$

where AGGREGATE denotes permutation invariant aggregation functions such as summation and average, a_e is the adjacency value computed by Theorem 4.1, and

$$\mathbf{m}_{e^M(v)} := \text{AGGREGATE}_{\{\mathcal{U} \in \pi(\cdot) \mid \pi(\cdot) \in \pi(e^M(-v))\}} (\text{CNI } \mathbf{x}_{\mathcal{U}}) \quad (23)$$

is the edge embedding for M^{th} -order hyperedge set e^M by aggregating cross-node interactions of each permuted sequence of neighborhood nodes from $\pi(e^M(-v))$. As the edge embeddings from Eq. (23) are aggregated for all hyperedges of a node with weights by adjacency values a_e , the two-step process proposed in Eq. (22) essentially aggregates all the cross-node interactions generated by the ordered node sequences from $\mathcal{N}^M(v)$ to infer the neighborhood embedding.

Using the hypergraph example of Fig. 2 for illustration, consider the case of $D = 1$ such that $x_i = \mathbf{x}_{v_i}$ ($i = 1, 2, \dots, N$). Then if we set node v_1 in Fig. 2(c) as the target node and let AGGREGATE be the summation operation, we obtain $\mathbf{m}_{e_1^3(v_1)} = (x_1x_2 + x_2x_1 + x_2^2)$, $\mathbf{m}_{e_2^3(v_1)} = (x_2x_3 + x_3x_2)$ by looking up the neighborhood table in Fig. 5(c). Since the coefficients $a_{e_1} = 1/3$ and $a_{e_2} = 1/2$ can be directly retrieved from the adjacency value table in Fig. 5(b), the neighborhood embedding of v_1 in the example is computed by $\mathbf{m}_{\mathcal{N}^3(v_1)} = \frac{1}{3}(x_1x_2 + x_2x_1 + x_2^2) + \frac{1}{2}(x_2x_3 + x_3x_2)$.

As we see from the above example, the neighboring aggregation first finds the connected nodes to a target node, and then sums up their cross-node interactions within corresponding hyperedges, which follows the same procedure as the shifting operation in the T-spatial convolution. Therefore, we summarize the connection between the T-spatial convolution and the t-message passing in the following theorem.

Theorem 4.2: Given any node $v_i \in \mathcal{V}$, its shifted signal $[\mathbf{Y}]_{i\cdot} = (y_{i1}, y_{i2}, \dots, y_{iD})^T$ by Eq. (13) is equivalent to the neighborhood embedding $\mathbf{m}_{\mathcal{N}^M(v_i)}$ computed by Eq. (22) up to a tensor normalization factor.

Proof. See Appendix F for proof. \square

In particular, if the adjacency tensor is normalized by scaling the entries with the degrees of relevant nodes (see Eq. (34) in Appendix C) and the AGGREGATE operations are defined to be average and summation in Eq. (22) and Eq. (23), respectively, then the neighborhood embedding becomes exactly the same as the shifted signal, that is, $\mathbf{m}_{\mathcal{N}^M(v_i)} = [\mathbf{Y}]_{i\cdot}$.

Algorithm 2 T-MPHN Forward Propagation

Input: Hypergraph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; node features $\{\mathbf{x}_v | v \in \mathcal{V}\}$; number of layers L ; hypergraph order M ; the adjacency value table; the neighborhood table, linear layers $\text{MLP}^{(l)}$, $l = 1, 2, \dots, L$; aggregation function AGGREGATE; combine operation COMBINE; nonlinear activation σ .

Output: Node embeddings $\mathbf{z}_v, \forall v \in \mathcal{V}$.

```

 $\mathbf{x}_v^{(0)} \leftarrow \text{MLP}(\mathbf{x}_v), \forall v \in \mathcal{V};$ 
for  $l = 1, \dots, L$  do
  for  $v \in \mathcal{V}$  do
    for  $e^M \in E^M(v)$  do
       $\mathbf{m}_{e^M(v)}^{(l)} \leftarrow \text{AGGREGATE}_{\{\mathcal{U} \in \pi(\cdot) | \pi(\cdot) \in \pi(e^M(-v))\}}^{(l-1)} (\text{CNI } \mathbf{x}_u^{(l-1)})_{u \in \mathcal{U}}$ 
    end for
     $\mathbf{m}_{\mathcal{N}^M(v)}^{(l)} \leftarrow \text{AGGREGATE}_{e^M \in E^M(v)}(a_e \mathbf{m}_{e^M(v)}^{(l)})$ 
     $\mathbf{x}_v^{(l)} \leftarrow \sigma(\text{MLP}^{(l)}(\text{COMBINE}(\mathbf{x}_v^{(l-1)}, \mathbf{m}_{\mathcal{N}^M(v)}^{(l)})))$ 
     $\mathbf{x}_v^{(l)} \leftarrow \mathbf{x}_v^{(l)} / \|\mathbf{x}_v^{(l)}\|_2$ 
  end for
end for
 $\mathbf{z}_v \leftarrow \mathbf{x}_v^{(L)}, \forall v \in \mathcal{V}$ 

```

Based on the computing scheme proposed above for the shifted signals, we next describe the T-MPHN algorithm, which is summarized in Algorithm 2. Let $\{\mathbf{x}_v | v \in \mathcal{V}\}$ be the input node features. To begin with, we first initialize

these node feature with one linear layer of regular multilayer perceptron (MLP) and project them into a latent space to obtain the initial hidden embedding features $\{\mathbf{x}_v^{(0)} | v \in \mathcal{V}\}$. 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 performs multi-layer operations as follows. Given the current layer l ($l = 1, \dots, L$), let $\{\mathbf{x}_v^{(l-1)} | v \in \mathcal{V}\}$ be the hidden embedding features from the previous layer, and let $\tilde{\mathbf{X}}^{(l-1)} = (\mathbf{x}_{v_1}^{(l-1)}, \dots, \mathbf{x}_{v_N}^{(l-1)})^T$ be the corresponding design matrix. For a given target node $v \in \mathcal{V}$, we first perform the efficient two-step aggregations to generate shifted hidden features $\mathbf{m}_{\mathcal{N}^M(v)}^{(l)}$ as shown in Eq. (24), where $\mathbf{m}_{e^M(v)}^{(l)}$ are computed by the aggregation scheme of Eq. (23) using the previous embedding features in $\tilde{\mathbf{X}}^{(l-1)}$.

Subsequently, the step in Eq. (25) essentially integrates the proposed T-spatial convolution by concatenating $\mathbf{x}_v^{(l-1)}$ with $\mathbf{m}_{\mathcal{N}^M(v)}^{(l)}$ to obtain an augmented node-specific vector followed by one regular linear layer ($\text{MLP}^{(l)}$) operation. Indeed, by Theorem 4.2, this step can be equivalently viewed as a weighted linear combination between the transformed hidden features $\tilde{\mathbf{X}}^{(l-1)}$ by a simple linear layer and the T-spatial convolution from the CNI signals of $\tilde{\mathbf{X}}^{(l-1)}$ by the convolution operation of Eq. (11).

Lastly, the resulting hidden features of node v are fed into a nonlinear (e.g., RELU) activation followed by a normalization step in Eq. (26) 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.

D. 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. Comprehensive investigation of other possible variations will be left to future work.

In the aggregation of Eq. (22), one can set the hypergraph order M as a fixed value so that any hyperedge with more than M nodes will be uniformly down-sampled to M degree. This down-sampling 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 l -th layer of HyperGNNs aggregates information from the l -th hop neighbors. As the aggregation propagates to neighbors that are multiple hops away from the central target node, less 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 Sec. 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 AG-

GREGATE function in Eq. (22) to be the mean function. That is,

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

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 LSTM [31] may also be considered in accordance to a study's learning task.

E. 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 aggregation scheme for hypergraph to avoid excessive space and time complexity. Let $d_m = \max_{v \in \mathcal{V}} d_v$ be the maximum degree of all nodes and let $D^{(l-1)}$ be the dimension of the embedding features generated from the previous layer $l-1$. Suppose 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 $\mathcal{O}(Nd_m)$ and the time complexity for each layer l is $\mathcal{O}(Nd_m D^{(l-1)})$. Therefore, rather than having the polynomial order of N^M for the T-spectral convolution as shown in Section III-C, both the space and time complexities of T-MPHN are only linearly increasing with N , which is practically comparable to the state-of-the-art HyperGNNs such as UniGCN [24] and HNHN [13].

V. CONNECTION TO RELATED WORK

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

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 Eq. (10) in Algorithm 1: under the hypergraph order $M=2$, the pre-Fourier transform and the post-Inverse Fourier transform in Algorithm 1 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 Eq. (11). Therefore, if a hypergraph is reduced to a simple graph ($M=2$), the T-spectral convolution is the same as the T-spatial convolution.

Connection between T-spatial convolution and T-MPHN. In Theorem 4.2, we shown that the neighborhood embedding $\mathbf{m}_{\mathcal{N}^M(v_i)}$ is equivalent to the shifted signal $[\mathbf{Y}]_i$ in the T-spatial convolution. Aside from the algorithmic perspective, the difference of 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.

Connection between T-MPHN and other related work.

As tensor is a generalization of matrix, certain matrix-based HyperGNNs built on hypergraph expansions 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 the $(M-1)$ 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 $\mathbf{m}_{\mathcal{N}^2(v_i)} = \sum_{e^2 \in E^2(v_i)} \mathbf{m}_{e^2(v_i)}$ and $\mathbf{m}_{e^2(v_i)} = \sum_{u \in e^2(-v_i)} \mathbf{x}_u$, which are simplified from the two aggregation steps in Eq. (22) and Eq. (23).

VI. EXPERIMENTS

The proposed T-HyperGNNs including the T-spectral convolution (T-spectral), the T-spatial convolution (T-spatial), and the T-Message-passing (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 cross-node interaction tensor. To demonstrate the scalability and conductivity of the T-MPHN, an inductive setting is applied to a 3D object recognition problem, in which the newly-added unseen nodes are evaluated during the testing process. We use the accuracy rate to be the metric. For each reported accuracy rate, 10 random data splits and 5 different parameter initialization (a total of 50 repetitions) 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 $\{0.01, 0.001\}$ and $\{0.005, 0.0005\}$, 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, we consider a transductive setting [31], in which the hypergraph structure is assumed to be the same during the training and testing processes. That is, we assume the testing node connections are known during model training.

Datasets. We use five standard hypergraph datasets in the academic network, which include two co-citation datasets (Cora and DBLP) and three co-authorship datasets (Cora, CiteSeer and PubMed). The hypergraph structure is obtained by viewing each paper as a node and each co-citation or co-author 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, computing, etc). The raw datasets [10] are further downsampled to smaller hypergraphs such that the T-spectral and the T-spatial convolution HyperGNNs can be applied to compare with the proposed T-MPHN. The descriptive statistics of these five hypergraphs are summarized in Table I.

TABLE I
SUMMARY STATISTICS OF THE ACADEMIC NETWORK DATASETS

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

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 over-smoothing 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 multi-layer perceptron (MLP), HGNN [12], HyperGCN [10], and HNHN [13] as our benchmarks since these methods are originally designed for transductive settings. Here 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.

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 more capable one to model various datasets and tasks; such capability is partially attributable to the concatenation of the neighborhood embedding and the central node embedding (that is, $\text{Concat}([\mathbf{x}_v, \mathbf{m}_{\mathcal{N}^M(v)}])$,

which forms a “skip-connection” between the input and the output of an aggregation step (see, e.g., GraphSAGE [31]).

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 cross-node interaction.

B. Ablation Study for T-MPHN

On the same academic networks, an ablation study is designed by “turning off” the adjacency values in Eq. (18) and the cross-node interactions in Eq. (21) separately and testing their corresponding performance. We consider three modeling scenarios: 1) the full T-MPHN model; 2) the T-MPHN model with the cross-node interaction but not the adjacency values; 3) the T-MPHN model with the adjacency values but not the cross-node interactions. In the second scenario, when the adjacency values are “turned off”, we fill in all ones instead. In the third scenario, when the cross-node interaction is “turned off”, we replace the Hadamard product of node features with their summation. The results of the ablation study are shown in Fig. 6. We can observe that the performance of the two

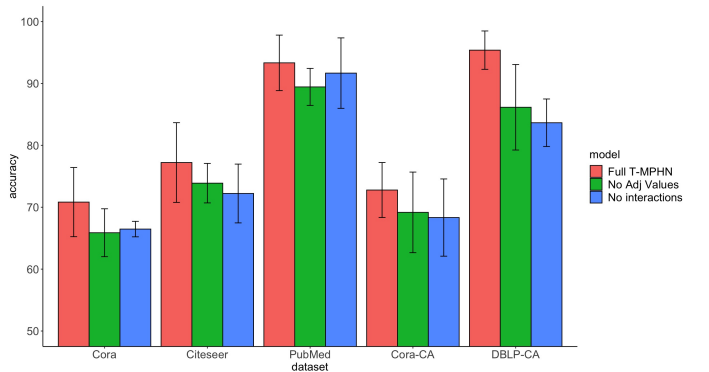


Fig. 6. Averaged accuracy of T-MPHN and its corrupted variations on the five academic networks for the ablation study.

corrupted T-MPHNs is worsened compared to the full T-MPHN, confirming its need of using the adjacency values and the cross-node interaction operation.

C. Inductive 3D Object Recognition

In this experiment, we apply T-MPHN to one of the important tasks in computer vision: 3D object recognition. The goal of 3D object recognition is to classify 3D objects into different categories. To better adapt practical circumstances, we assume the 3D object datasets are evolving in which

TABLE II
AVERAGED TESTING ACCURACY (% , \pm STANDARD DEVIATION) ON FIVE ACADEMIC NETWORKS FOR TRANSDUCTIVE NODE CLASSIFICATION. THE TOP THREE RESULTS ARE HIGHLIGHTED FOR EACH DATASET.

Method	Cocitation		Pubmed	Coauthorship	
	Cora	Citeseer		Cora	DBLP
MLP	48.23 \pm 7.35	65.56 \pm 1.48	73.89 \pm 5.60	46.11 \pm 8.35	76.15 \pm 7.26
HGNN	70.59 \pm 1.22	73.89 \pm 8.98	82.22 \pm 1.33	66.94 \pm 6.51	93.08 \pm 6.39
HyperGCN	35.29 \pm 1.24	61.11 \pm 1.53	76.11 \pm 1.40	25.79 \pm 6.43	25.38 \pm 1.29
HNHN	69.41 \pm 9.04	74.44 \pm 9.69	77.22 \pm 4.08	71.39 \pm 5.56	93.85 \pm 5.76
T-spectral	71.59 \pm 3.43	78.33 \pm 8.03	86.67 \pm 1.18	75.29 \pm 5.59	96.10 \pm 3.16
T-spatial	69.17 \pm 7.58	76.11 \pm 7.05	84.22 \pm 3.26	70.00 \pm 6.01	94.62 \pm 2.93
T-MPHN	70.83 \pm 5.59	77.22 \pm 6.44	93.33 \pm 4.48	72.78 \pm 4.44	95.38 \pm 2.10

unseen objects are added during testing. This setting is called inductive learning [31], as opposed to the transductive setting in the previous experiments. To create the inductive setting from our static data, we follow HyperSAGE [17] to randomly reserve 40% nodes as unseen nodes for testing, while 20% and 40% nodes are used for regular training and validation, respectively.

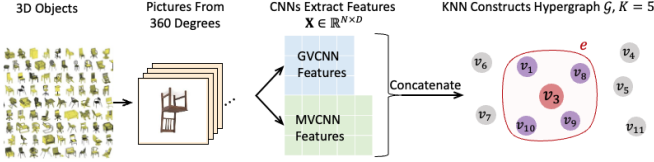


Fig. 7. The preprocessing steps for the 3D object recognition task.

Datasets. We employ two public datasets known as the Princeton ModelNet40 dataset [32] and the National Taiwan University (NTU) 3D model dataset [33]. On these two datasets, each 3D object is viewed as a node, and the features associated with each node are extracted using Group-View Convolutional Neural Network (GVCNN) [34] and Multi-View Convolutional Neural Network (MVCNN) [35] following the experimental setting of [12]. The resulting feature dimensions from the MVCNN and the GVCNN are 4096 and 2048, respectively, and we concatenate them to form the input features for our study. To form the hypergraph structures for these two datasets, we also follow the setup of [12] by using the K-nearest neighbor algorithm with $K = 5$ so that all hyperedges of the constructed hypergraph have size 5. We summarize the data preprocessing steps described above in Fig. 7. The goal of the experiment is to predict the label associated with each node (e.g., window, aircraft, shelf, etc). The statistics of the two 3D object recognition datasets are summarized in Table III.

TABLE III
SUMMARY STATISTICS OF 3D OBJECT RECOGNITION DATASETS

Statistic	ModelNet40	NTU
$ \mathcal{V} $	12311	2012
$ \mathcal{E} $	24622	4024
Feature Dimension D	6144	6144
Number of Classes	40	67

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, HyperSAGE [17], and UniSAGE [24]. The HyperSAGE defines the intra-edge and inter-edge aggregations through a generalized mean function $M_p = (\frac{1}{n} \sum_{i=1}^n x_i^p)^{\frac{1}{p}}$, and we adopt $p = 1$ for its best results. 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 layer, respectively. For all models, we construct 2-layer neural networks.

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 both seen and unseen nodes. A closer comparison between seen and unseen samples shows that generalizing a trained model to unseen nodes is not an easy task as all models show reduced accuracy rates. For example, the HyperSAGE gives even lower accuracy than that of the MLP for unseen nodes. By comparing the reduced accuracy percentages from seen nodes to unseen nodes, we also observe the desirable result that the T-MPHN shows the smallest reduction among the four methods.

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 over-smoothing 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.

VII. CONCLUSION AND FUTURE WORK

In this paper, we introduce tensor representations of hypergraphs and derive hypergraph T-spectral convolution by the tensor t-product. While hypergraph neural networks can be built on the T-spectral convolution, the time and space complexities are too large for some real-world applications. To alleviate the time complexity, we localize the T-spectral convolution to the T-spatial convolution by taking only the first frontal slice of the T-spectral convolution. Furthermore,

TABLE IV

ACCURACY (% , \pm STANDARD DEVIATION) ON TWO 3D OBJECT RECOGNITION DATASETS. THE REDUCED PERCENT MEANS THE ACCURACY PERCENTAGE REDUCTION FROM SEEN NODES TO UNSEEN NODES. THE BEST RESULTS ARE HIGHLIGHTED FOR EACH DATASET.

Method	ModelNet40			NTU		
	Seen	Unseen	Reduced (%)	Seen	Unseen	Reduced (%)
MLP	96.13 \pm 2.17	88.42 \pm 1.41	8.72	94.51 \pm 4.70	77.68 \pm 4.46	17.81
HyperSAGE	97.55 \pm 2.35	88.37 \pm 2.66	10.39	97.33 \pm 3.58	75.34 \pm 1.04	22.59
UniSAGE	100.00 \pm 0.00	92.62 \pm 2.19	7.38	96.60 \pm 1.43	81.05 \pm 0.82	16.10
T-MPHN	100.00 \pm 0.00	96.69 \pm 3.22	3.31	100.00 \pm 0.00	86.34 \pm 2.17	13.66

to avoid direct tensor usage and enable inductive learning, we introduce the compressed adjacency tensor and proposing the tensor-message passing hypergraph neural network (T-MPHN), which can efficiently handle large hypergraphs containing thousands of vertices as confirmed by the extensive numerical experiment. The empirical results on the five academic networks showed very competitive performance of our proposed HyperGNNs in comparison to the other state-of-the-art benchmarks. In addition, we show the effectiveness of tensor representations and high-order interaction as the key components of T-MPHN by an ablation study. Some promising extensions of our proposal include adding attention learning module [22] with probabilistic models and establishing the underlying equivalency with hypergraph denoising problem, and we leave these interesting topics to future studies.

REFERENCES

- [1] Y. Wang, P. Li, C. Bai, and J. Leskovec, "Tedic: Neural modeling of behavioral patterns in dynamic social interaction networks," pp. 693–705, 2021.
- [2] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," *Advances in neural information processing systems*, vol. 31, 2018.
- [3] S. Zhang, S. Cui, and Z. Ding, "Hypergraph-based image processing," pp. 216–220, 2020.
- [4] A. Zeng, X. Sun, L. Yang, N. Zhao, M. Liu, and Q. Xu, "Learning skeletal graph neural networks for hard 3d pose estimation," pp. 11 436–11 445, 2021.
- [5] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "Kgat: Knowledge graph attention network for recommendation," pp. 950–958, 2019.
- [6] H. Ren, H. Dai, B. Dai, *et al.*, "Smore: Knowledge graph completion and multi-hop reasoning in massive knowledge graphs," pp. 1472–1482, 2022.
- [7] Y.-Y. Chang, P. Li, R. Sasic, M. Afifi, M. Schweighauser, and J. Leskovec, "F-fade: Frequency factorization for anomaly detection in edge streams," pp. 589–597, 2021.
- [8] M. M. Wolf, A. M. Klinvex, and D. M. Dunlavy, "Advantages to modeling relational data using hypergraphs versus graphs," *2016 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, 2016.
- [9] X. Yue, Z. Wang, J. Huang, *et al.*, "Graph embedding on biomedical networks: Methods, applications and evaluations," *Bioinformatics*, vol. 36, no. 4, pp. 1241–1251, 2020.
- [10] N. Yadati, M. Nimishakavi, P. Yadav, V. Nitin, A. Louis, and P. Talukdar, "Hypergen: A new method for training graph convolutional networks on hypergraphs," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [11] C. Zheng, X. Fan, C. Wang, and J. Qi, "Gman: A graph multi-attention network for traffic prediction," vol. 34, no. 01, pp. 1234–1241, 2020.
- [12] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 3558–3565, 2019.
- [13] Y. Dong, W. Sawin, and Y. Bengio, "Hnhn: Hypergraph networks with hyperedge neurons," *arXiv preprint arXiv:2006.12278*, 2020.
- [14] 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 Processing*, vol. 187, p. 108 149, 2021.
- [15] E. Chien, C. Pan, J. Peng, and O. Milenkovic, "You are allset: A multiset function framework for hypergraph neural networks," *arXiv preprint arXiv:2106.13264*, 2021.
- [16] C. Wan, M. Zhang, W. Hao, S. Cao, P. Li, and C. Zhang, "Principled hyperedge prediction with structural spectral features and neural networks," *arXiv preprint arXiv:2106.04292*, 2021.
- [17] D. Arya, D. K. Gupta, S. Rudinac, and M. Worring, "Hypersage: Generalizing inductive representation learning on hypergraphs," *arXiv preprint arXiv:2010.04558*, 2020.
- [18] 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 Journal on Matrix Analysis and Applications*, vol. 34, no. 1, pp. 148–172, 2013.
- [19] M. E. Kilmer and C. D. Martin, "Factorization strategies for third-order tensors," *Linear Algebra and its Applications*, vol. 435, no. 3, pp. 641–658, 2011.
- [20] K. Pena-Pena, D. Lau, and G. Arce, "Thgsp: Hypergraph signal processing using t-product tensor decompositions," *Submitted to IEEE transactions on signal processing, also in TechRxiv.*, 2022.
- [21] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," *Proceedings of the 22nd*

- ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864, 2016.
- [22] S. Bai, F. Zhang, and P. H. Torr, “Hypergraph convolution and hypergraph attention,” *Pattern Recognition*, vol. 110, p. 107637, 2021.
- [23] C. Yang, R. Wang, S. Yao, and T. Abdelzaher, “Hypergraph learning with line expansion,” *arXiv preprint arXiv:2005.04843*, 2020.
- [24] J. Huang and J. Yang, “Unignn: A unified framework for graph and hypergraph neural networks,” *arXiv preprint arXiv:2105.00956*, 2021.
- [25] R. Wang, B. Fu, G. Fu, and M. Wang, “Deep & cross network for ad click predictions,” in *Proceedings of the ADKDD’17*, 2017, pp. 1–7.
- [26] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, “Xdeepfm: Combining explicit and implicit feature interactions for recommender systems,” pp. 1754–1763, 2018.
- [27] C. Ye and Y. Yang, “High-dimensional adaptive minimax sparse estimation with interactions,” *IEEE Transactions on Information Theory*, vol. 65, no. 9, pp. 5367–5379, 2019.
- [28] M. E. Kilmer, C. D. Martin, and L. Perrone, “A third-order generalization of the matrix svd as a product of third-order tensors,” *Tufts University, Department of Computer Science, Tech. Rep. TR-2008-4*, 2008.
- [29] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, “Modeling relational data with graph convolutional networks,” *European semantic web conference*, pp. 593–607, 2018.
- [30] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” *Advances in neural information processing systems*, vol. 29, 2016.
- [31] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” *Advances in neural information processing systems*, vol. 30, 2017.
- [32] Z. Wu, S. Song, A. Khosla, *et al.*, “3d shapenets: A deep representation for volumetric shapes,” *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.
- [33] D.-Y. Chen, X.-P. Tian, Y.-T. Shen, and M. Ouhyoung, “On visual similarity based 3d model retrieval,” vol. 22, no. 3, pp. 223–232, 2003.
- [34] Y. Feng, Z. Zhang, X. Zhao, R. Ji, and Y. Gao, “Gvcnn: Group-view convolutional neural networks for 3d shape recognition,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 264–272, 2018.
- [35] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, “Multi-view convolutional neural networks for 3d shape recognition,” pp. 945–953, 2015.
- [36] C. D. Martin, R. Shafer, and B. LaRue, “An order-p tensor factorization with applications in imaging,” *SIAM Journal on Scientific Computing*, vol. 35, no. 1, A474–A490, 2013.
- [37] A. Banerjee, A. Char, and B. Mondal, “Spectra of general hypergraphs,” *Linear Algebra and its Applications*, vol. 518, pp. 14–30, 2017.
- [38] B. C. Rennie and A. J. Dobson, “On stirling numbers of the second kind,” *Journal of Combinatorial Theory*, vol. 7, no. 2, pp. 116–121, 1969.
- [39] S. Mallat, *A wavelet tour of signal processing*. Elsevier, 1999.
- [40] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” *Advances in neural information processing systems*, vol. 29, 2016.
- [41] F. Gama, E. Isufi, G. Leus, and A. Ribeiro, “Graphs, convolutions, and neural networks: From graph filters to graph neural networks,” *IEEE Signal Processing Magazine*, vol. 37, no. 6, pp. 128–138, 2020.
- [42] F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi, “Graph neural networks with convolutional arma filters,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.