

# 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)

# Supplementary Material: 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*

## APPENDIX A DEFINITION OF T-PRODUCT

**Definition 3 (Tensor T-product [18]):** The T-product of two 3<sup>rd</sup> order tensors  $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$  and  $\mathcal{Y} \in \mathbb{R}^{N_2 \times N_4 \times N_3}$  is the tensor  $\mathcal{Z} \in \mathbb{R}^{N_1 \times N_4 \times N_3}$  computed as

$$\begin{aligned} \mathcal{Z} &= \mathcal{X} * \mathcal{Y} = \text{fold}(\text{bcirc}(\mathcal{X}) \cdot \text{unfold}(\mathcal{Y})) \\ &= \text{fold} \left( \begin{bmatrix} \mathbf{X}^{(1)} & \mathbf{X}^{(N_3)} & \dots & \mathbf{X}^{(2)} \\ \mathbf{X}^{(2)} & \mathbf{X}^{(1)} & \dots & \mathbf{X}^{(3)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{X}^{(N_3)} & \mathbf{X}^{(N_3-1)} & \dots & \mathbf{X}^{(1)} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{Y}^{(1)} \\ \mathbf{Y}^{(2)} \\ \vdots \\ \mathbf{Y}^{(N_3)} \end{bmatrix} \right), \end{aligned} \quad (27)$$

$$(28)$$

where the operator  $\text{bcirc}(\mathcal{X})$  converts the set of frontal slices of the tensor  $\mathcal{X}$  into a block circulant matrix and  $\text{unfold}(\mathcal{Y})$  stacks vertically the set of frontal slices of  $\mathcal{Y}$  into a  $N_2 N_3 \times N_4$  matrix. The operator  $\text{fold}(\cdot)$  reverses this process,  $\text{fold}(\text{unfold}(\mathcal{X})) = \mathcal{X}$ . Since circulant matrices are diagonalized by the discrete Fourier transform, the T-product can be computed efficiently in the Fourier domain as explained in detail in [18]. Using MATLAB notation, let  $\hat{\mathcal{X}} := \text{fft}(\mathcal{X}, [], 3)$  denote the tensor obtained by applying the fast Fourier transform (FFT) along each tubal scalar of  $\mathcal{X}$ . Thus, the T-product of  $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$  and  $\mathcal{Y} \in \mathbb{R}^{N_2 \times N_4 \times N_3}$  can be alternatively computed as

$$\mathcal{Z} = \mathcal{X} * \mathcal{Y} := \text{ifft}(\{\hat{\mathbf{X}}^{(k)} \hat{\mathbf{Y}}^{(k)}\}_{k=1}^{N_3}, [], 3), \quad (29)$$

where  $\text{ifft}$  is the inverse FFT. The T-product can be easily extended to high-order tensors in a recursive manner[36]. For the  $M^{\text{th}}$ -order tensors  $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times N_3 \times \dots \times N_M}$  and  $\mathcal{Y} \in \mathbb{R}^{N_2 \times L \times N_3 \times \dots \times N_M}$ , their T-product  $\mathcal{Z} \in \mathbb{R}^{N_1 \times L \times N_3 \times \dots \times N_M}$  is computed recursively as

$$\begin{aligned} \mathcal{Z} &= \mathcal{X} * \mathcal{Y} = \text{fold}(\text{bcirc}(\mathcal{X}) \cdot \text{unfold}(\mathcal{Y})) \\ &= \text{fold} \left( \begin{bmatrix} \mathcal{X}^{(1)} & \mathcal{X}^{(N_M)} & \dots & \mathcal{X}^{(2)} \\ \mathcal{X}^{(2)} & \mathcal{X}^{(1)} & \dots & \mathcal{X}^{(3)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{X}^{(N_M)} & \mathcal{X}^{(N_M-1)} & \dots & \mathcal{X}^{(1)} \end{bmatrix} * \begin{bmatrix} \mathcal{Y}^{(1)} \\ \mathcal{Y}^{(2)} \\ \vdots \\ \mathcal{Y}^{(N_M)} \end{bmatrix} \right), \end{aligned} \quad (30)$$

$$(31)$$

where  $\mathcal{X}^{(k)}$  and  $\mathcal{Y}^{(k)}$  are  $(M-1)$ -order tensors formed from flattening the  $M^{\text{th}}$  order, and it can be done recursively to the following  $(M-1)^{\text{th}}$ ,  $(M-2)^{\text{th}}$ , and so on. Each of these successive flatten operations thus involves a T-product of one order less until the base case of the 3<sup>rd</sup>-order tensors.

## APPENDIX B SYMMETRIZATION OF TENSORS.

In order to obtain symmetric block circulant matrices in the t-product, we need to symmetrize the adjacency tensor and the hypergraph signal tensor. Therefore, we define a symmetrization operator  $\text{sym}(\mathcal{A})$  that generates a symmetric version  $\mathcal{A}_s \in \mathbb{R}^{N \times N \times (2N+1)}$  of  $\mathcal{A} \in \mathbb{R}^{N \times N \times N}$ , by adding a matrix of zeros  $\mathbf{0}_{N \times N}$  as the first frontal slice, dividing by 2, and reflecting the frontal slices of  $\mathcal{A}$  along the third dimension as

$$\mathcal{A}_s = \text{sym}(\mathcal{A}) = \text{fold} \left( \begin{bmatrix} \mathbf{0}_{N \times N} \\ \frac{1}{2} \mathbf{A}^{(1)} \\ \frac{1}{2} \mathbf{A}^{(2)} \\ \vdots \\ \frac{1}{2} \mathbf{A}^{(2)} \\ \frac{1}{2} \mathbf{A}^{(1)} \end{bmatrix} \right). \quad (32)$$

Now, if we let  $N_s = 2N + 1$ , for a higher-order tensor  $\mathcal{A} \in \mathbb{R}^{N^M}$ , its symmetric version is a  $M^{\text{th}}$ -order tensor  $\mathcal{A}_s \in \mathbb{R}^{N \times N \times N_s \times \dots \times N_s}$  obtained by recursively appending a  $(M-1)$ -th-order tensor of zeros  $\mathcal{O} \in \mathbb{R}^{N^{(M-1)}}$  at the front, dividing by 2, and reflecting the  $(M-1)$ -th-order tensors  $\mathcal{A}^{(l)}$  along the last dimension as

$$\mathcal{A}_s = \text{sym}(\mathcal{A}) = \text{fold} \left( \begin{bmatrix} \mathcal{O} \\ \frac{1}{2} \text{sym}(\mathcal{A}^{(1)}) \\ \frac{1}{2} \text{sym}(\mathcal{A}^{(2)}) \\ \vdots \\ \frac{1}{2} \text{sym}(\mathcal{A}^{(2)}) \\ \frac{1}{2} \text{sym}(\mathcal{A}^{(1)}) \end{bmatrix} \right). \quad (33)$$

When applied to the hypergraph signal tensor and the weight tensor, we obtain  $\mathcal{X}_s$  and  $\mathcal{W}_s$ , respectively. Notice that this operation is reversible.

## APPENDIX C DEFINITION OF NORMALIZED ADJACENCY TENSOR

The normalized adjacency matrix of a graph generally can be defined in non-symmetric or symmetric manner. Similar to the matrix setting, we define the normalized adjacency tensor in two ways.

**Definition 4 (Normalized adjacency tensor [37]):** For a hypergraph  $\mathcal{G}$  without any isolated vertex, the non-symmetric

normalized adjacency tensor  $\mathcal{A}^{norm} \in \mathbb{R}^{N^M}$  of the hypergraph is a stochastic tensor defined as: for any hyperedge  $e_k = \{v_{k_1}, v_{k_2}, \dots, v_{k_c}\} \in \mathcal{V}(\mathcal{H})$  of cardinality  $c \leq M$ ,

$$\tilde{a}_{p_1 p_2 \dots p_M} = \frac{c}{\alpha} \frac{1}{d(v_{p_1})}, \quad (34)$$

where  $\alpha$  and the indices are defined in Eq. (1), and  $d(v_{p_1})$  is the degree of vertex  $p_1$ . Alternatively, the normalized adjacency tensor can also be defined by multiplying  $M$ -th square root along each mode, i.e., for any hyperedge  $e_k = \{v_{k_1}, v_{k_2}, \dots, v_{k_c}\} \in \mathcal{V}(\mathcal{H})$  of cardinality  $c \leq M$ ,

$$\tilde{a}_{p_1 p_2 \dots p_M} = \frac{c}{\alpha} \prod_{j=1}^M \frac{1}{\sqrt[M]{d(v_{p_j})}}. \quad (35)$$

These two normalized adjacency tensors have the same eigenvalues [37], so for notation simplicity, we refer to the definition in Eq. (34) as normalized adjacency tensors and use the notation  $\mathcal{A}^{norm}$  to denote the normalized adjacency tensors (and  $\mathcal{L}^{norm}$  as the normalized Laplacian tensors).

#### APPENDIX D

##### POOF OF PROPOSITION 1 AND 2.

*Proof.* As pointed out in Eq. (13), from a node-wise perspective, the aggregation or shifting in the T-spatial convolution  $\mathbf{Y} := (\mathcal{A}_s^{norm} * \mathcal{X}_s)^{(1)}$  is equivalent to summing up all the connected cross-node interactions, which only involves in neighbors of the target node. By Proposition 1 and Eq. (13), we know that the T-spatial convolution only aggregates neighboring information for each node, which means that no matter what is the ordering of nodes, the output of the T-spatial convolution is the same for each node. In other words, even without the ordering of nodes, the T-spatial convolution can be performed by the two-step message-passing rule showed in Eq. (22) and Eq. (23), which confirms that the T-spatial convolution is permutation invariant.  $\square$

#### APPENDIX E

##### PROOF OF THEOREM.4.1

*Proof.* The proof is directly inspired by the *Stirling numbers of the second kind* [38], which states the number of ways to partition a set of  $N$  labelled objects into  $k$  nonempty unlabelled subsets. Denote the Stirling numbers of the second kind as  $S(N, k)$ , the explicit formula is given by

$$S(N, k) = \frac{1}{k!} \sum_{i=0}^k (-1)^i \binom{k}{i} (k-i)^N. \quad (36)$$

In the definition of the adjacency tensor, we have a set of  $M$  labelled modes, and would like to partition it into  $|e|$  nonempty labelled subsets, so from the Stirling numbers of the second kind, we obtain

$$\alpha = S(M, |e|) \times k! = \sum_{i=0}^{|e|} (-1)^i \binom{|e|}{i} (|e| - i)^M. \quad (37)$$

$\square$

#### APPENDIX F

##### PROOF OF THEOREM 4.2

*Proof.* From the shifting operation in the T-spatial convolution (Eq. (13)), we can see there are three steps associated. First, considering the sparsity of the adjacency tensor, it is clear that only nonzero values of  $a_{i j i_3 \dots i_M}$  will be added, this is equivalent to finding the  $M^{\text{th}}$ -order neighborhood of a node as we defined in Def. 2. Then for different hyperedges, we have different adjacency values that can be computed based on Theorem 4.1. Thirdly, the cross-node product  $x_{j d i_3 \dots i_M}$  is defined through the CNI in the message passing (see Eq. (21)). All three steps together finish the proof.  $\square$

#### APPENDIX G

##### TENSOR TRANSPOSE

The transpose of a tensor, under the t-algebra, can be defined following the symmetrization above. Similarly, the transpose is defined recursively with the three-order tensor as the base case. The transpose of a three-order tensor  $\mathcal{A} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$ , denoted by  $\mathcal{A}^T \in \mathbb{R}^{N_2 \times N_1 \times N_3}$ , is obtained by transposing each of the frontal slices of  $\mathcal{A}$ , and then reversing the order of the transposed frontal slices from 2 through  $N_3$ . For a  $M^{\text{th}}$ -order tensor  $\mathcal{A} \in \mathbb{R}^{N_1 \times N_2 \times \dots \times N_M}$ , its transpose  $\mathcal{A}^T \in \mathbb{R}^{N_2 \times N_1 \times \dots \times N_M}$  is obtained by recursively transposing each sub-order tensor  $\mathcal{A}^{(m)}$  for  $m = 1, 2, \dots, M$  and then reversing the order of the  $\mathcal{A}^{(m)}$ 's from  $m = 2$  to  $m = M$ , i.e.,

$$\mathcal{A}^T = \text{fold} \left( \begin{bmatrix} \mathcal{A}^{(1)T} \\ \mathcal{A}^{(M)T} \\ \vdots \\ \mathcal{A}^{(2)T} \end{bmatrix} \right). \quad (38)$$

As a result,  $\mathcal{A} = \mathcal{A}^T$  if  $\mathcal{A}$  is a symmetrized tensor.

#### APPENDIX H

##### DEVIATION OF HYPERGRAPH SPECTRAL CONVOLUTION

In the main body, we introduced the T-spectral convolution directly and ignored the relationship between convolutions and the hypergraph Fourier space, in which the convolution theorem [39] is originally defined. Since designing various spectral filters such as polynomial[29], [30], [40], and ARMA[41], [42] leads to different approaches, which widens the design space of convolutional hypergraph neural networks. Here, in this appendix, we show how the T-spectral convolution is derived from the spectral space.

##### A. Construct Spectral Space

The eigendecomposition of hypergraph Laplacian serves as the basis of hypergraph spectral space. We define the hypergraph Laplacian as a difference operator.

**Definition 5 (Laplacian Tensor):** Given a hypergraph  $\mathcal{G}$  with  $N$  nodes and  $m.c.e(\mathcal{G}) = M$ , the Laplacian tensor is defined as the

$$\mathcal{L} = \mathcal{D} - \mathcal{A} \in \mathbb{R}^{N^M}, \quad (39)$$

where  $\mathcal{A}$  is the adjacency tensor and  $\mathcal{D}$  is a super-diagonal degree tensor with degree of node  $v_i$  on the

$\square$

corresponding diagonal entry  $d_{i\dots i}$ . Particularly,  $d_{i\dots i} = \sum_{j_1, j_2, \dots, j_{M-1}=1}^N a_{ij_1 j_2 \dots j_{M-1}}$ . To ensure a bounded spectrum of the Laplacian tensor, the normalized hypergraph Laplacian tensor is further defined as

$$\mathcal{L}^{norm} = \mathcal{I} - \mathcal{A}^{norm}, \quad (40)$$

where  $\mathcal{I}$  is a super-diagonal identity tensor with all diagonal entries as 1, and  $\mathcal{A}^{norm}$  is the normalized adjacency tensor.

Moving on now to consider the decomposition of the Laplacian tensor, we carefully choose the T-eigendecomposition that offers better insights perfectly analogous to the characteristics of the eigendecomposition in the traditional graph spectral convolution theorem [30]. Given the normalized Laplacian tensor  $\mathcal{L}^{norm} \in \mathbb{R}^{N^M}$  of an  $M^{\text{th}}$  order hypergraph, we first modify it to its symmetric version  $\mathcal{L}_s^{norm} \in \mathbb{R}^{N \times N \times (2N+1)^{(M-2)}}$  according to the symmetries operation in Appendix B. For notation simplicity, we let  $N_s = (2N+1)$ . It follows that the T-eigendecomposition of  $\mathcal{L}_s^{norm}$  is expressed as

$$\mathcal{L}_s^{norm} = \mathcal{V} * \mathbf{\Lambda} * \mathcal{V}^T, \quad (41)$$

where  $\mathcal{V} \in \mathbb{R}^{N \times N \times N_s^{(M-2)}}$  is an orthogonal tensor[36], and  $\mathbf{\Lambda} \in \mathbb{R}^{N \times N \times N_s^{(M-2)}}$  is a f-diagonal tensor whose frontal slices are diagonal matrices[36]. A visualization of T-eigendecomposition for a 3<sup>rd</sup>-order Laplacian tensor is shown in Fig. 8. The diagonal components in  $\mathbf{\Lambda}$  are in a decreasing

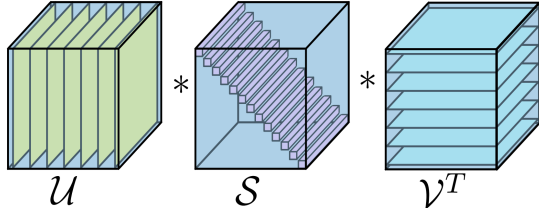


Fig. 8. Visual illustration of the T-eigendecomposition for a 3<sup>rd</sup>-order Laplacian Tensor.

order [36], which generalizes the concept of frequency of graphs to hypergraphs.

### B. Hypergraph Spectral Convolution

Based on convolution theorem [39], the hypergraph spectral convolution between two hypergraph signals is then defined as the element-wise product of their Fourier transforms.

**Definition 6 (Hypergraph convolution):** The hypergraph spectral convolution  $\star_G$  between a filter  $\mathcal{H}_s \in \mathbb{R}^{N \times 1 \times N_s^{(M-2)}}$  and a hypergraph signal  $\mathcal{X}_s \in \mathbb{R}^{N \times 1 \times N_s^{(M-2)}}$  is defined as

$$\mathcal{H}_s \star_G \mathcal{X}_s = \mathcal{V} * ((\mathcal{V}^T * \mathcal{H}_s) \odot (\mathcal{V}^T * \mathcal{X}_s)), \quad (42)$$

where  $*$  is the T-product defined in Appendix A,  $\odot$  is the element-wise Hadamard product, and  $\mathcal{V}$  is the decomposed orthogonal tensor for  $\mathcal{L}_s^{norm}$ . Let  $\hat{\mathcal{H}}_s = \mathcal{U}^T * \mathcal{H}_s \in \mathbb{R}^{N \times 1 \times N_s^{(M-2)}}$  be the Fourier transform of the filter  $\mathcal{H}_s$ , then the hypergraph convolution is equivalent to

$$\mathcal{H}_s \star_G \mathcal{X}_s = \mathcal{V} * (\hat{\mathcal{H}}_s \odot (\mathcal{V}^T * \mathcal{X}_s)) \quad (43)$$

$$= \mathcal{V} * (\text{diag}(\hat{\mathcal{H}}_s) * \mathcal{V}^T * \mathcal{X}_s), \quad (44)$$

where  $\text{diag}(\hat{\mathcal{H}}_s) = \begin{bmatrix} \hat{h}_1 & \dots & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & \dots & \hat{h}_N \end{bmatrix} \in \mathbb{R}^{N \times N \times N_s^{(M-2)}}$ , and

$\hat{h}_i \in \mathbb{R}^{1 \times 1 \times N_s^{(M-2)}}$ 's are tuples of  $\hat{\mathcal{H}}_s$ . The Fourier transformed filter  $\hat{\mathcal{H}}$  itself can be viewed as a non-parametric spectral filter. However, a filter created in this non-parametric manner has little to no dependence on the graph's structure and might not satisfy many of the convolution's desired properties. Such filters, for instance, may propagate to any node arbitrarily. A general and natural practice is to apply filtering on frequency components of the Laplace, leading to hypergraph spectral convolution.

**Definition 7 (Hypergraph Spectral Convolution):** Given the frequency representation  $\mathbf{\Lambda}$  of a hypergraph, parameterize the filter  $h : \mathbb{R} \rightarrow \mathbb{R}$  as  $h(\mathbf{\Lambda})$ , then the hypergraph spectral convolution is defined as

$$\mathcal{H}_s \star_G \mathcal{X}_s = \mathcal{V} * h(\mathbf{\Lambda}) * \mathcal{V}^T * \mathcal{X}_s \quad (45)$$

$$= h(\mathcal{L}_s^{norm}) * \mathcal{X}_s \quad (46)$$

with the frequency response  $h(\mathbf{\Lambda}) = \begin{bmatrix} h(\lambda_1) & \dots & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & \dots & h(\lambda_N) \end{bmatrix}$ .

By constructing hypergraph spectral convolution via a frequency filter, the resulting convolution commutes with the Laplacian tensor, which is localized in space. [40]. In this way, different filters  $h(\mathbf{\Lambda})$  can be designed according to specific tasks.

### C. Connection to Spectral T-HGCN

The formulation of our T-spectral convolution is indeed derived from a recursive polynomial parametrization of frequencies, in particular, Chebyshev polynomial. The reason to use recursive polynomial has two folds: 1) recursive formulation is computationally efficient; 2) such recursion can be naturally modeled by cascading layers of neural networks, which is especially appropriate for developing hypergraph convolutional neural network. Recall that the recursive Chebyshev polynomial of order  $k$  is  $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ , with  $T_0 = 1, T_1 = x$ . A spectral filter thus can be designed as

$$h_\theta(\mathbf{\Lambda}) = \sum_{k=0}^{K-1} T_k(\tilde{\mathbf{\Lambda}}), \quad (47)$$

where  $\theta_k$  is the filter weight for the  $k^{\text{th}}$  order of the Chebyshev polynomial, and  $T_k(\tilde{\mathbf{\Lambda}}) \in \mathbb{R}^{N \times N \times N_s^{(M-2)}}$  is the Chebyshev polynomial of order  $k$  evaluated at  $\tilde{\mathbf{\Lambda}} = 2/\lambda_{\max} \mathbf{\Lambda} - \mathcal{I}_{N_s}$ .  $\lambda_{\max}$  is the maximum value of eigentuples of  $\mathcal{L}^{norm}$ , and  $\mathcal{I}_{N_s} \in \mathbb{R}^{N \times N \times N_s^{(M-2)}}$  is the symmetric identity tensor. It was proved in [37] that the normalized adjacency tensor  $\mathcal{L}^{norm}$  is with the largest eigenvalue  $\lambda_{\max} = 2$ , such that  $\hat{\mathbf{\Lambda}} = \mathbf{\Lambda} - \mathcal{I}_{N_s}$  has eigenvalues within the range  $[-1, 1]$ . Applying the truncated order  $K$  expansion of the Chebyshev polynomial to the spectral convolution in Eq. (45), we then obtain

$$\mathcal{H}_s \star_G \mathcal{X}_s = \sum_{k=0}^K \theta_k \mathcal{V} * T_k(\hat{\mathbf{\Lambda}}) * \mathcal{V}^T * \mathcal{X}_s. \quad (48)$$

Since orders higher than 1 can be performed by stacking neural network layers, we follow GCN [29] and restrict the order of the Chebyshev polynomial to  $K = 1$ , leading to the spectral convolution

$$\mathcal{H}_s \star_{\mathcal{G}} \mathcal{X}_s = \theta_0 \mathcal{X}_s + \theta_1 \mathcal{V} * (\mathbf{I} - \mathcal{I}_{N_s}) * \mathcal{V}^T * \mathcal{X}_s \quad (49)$$

$$= \theta_0 \mathcal{X}_s + \theta_1 (\mathcal{L}^{norm} - \mathcal{I}_{N_s}) * \mathcal{X}_s \quad (50)$$

$$= \theta_0 \mathcal{X}_s - \theta_1 \mathcal{A}_s^{norm} * \mathcal{X}_s. \quad (51)$$

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

$$\mathcal{H}_s \star_{\mathcal{G}} \mathcal{X}_s = w(\mathcal{I}_{N_s} + \mathcal{A}_s^{norm}) * \mathcal{X}_s. \quad (52)$$

where the  $(\mathcal{I}_{N_s} + \mathcal{A}_s^{norm})$  can be treated as an adjusted hypergraph adjacency tensor with self loop of each node. Depending how important are the self features for the downstream task, one can also remove the self loop as  $\mathcal{H}_s \star_{\mathcal{G}} \mathcal{X}_s = w \mathcal{A}_s^{norm} * \mathcal{X}_s$ , which can be interpreted as inferring central nodes using their own neighbors, and the operation  $\mathcal{A}_s^{norm} * \mathcal{X}_s$  is in deed a hypergraph signal shifting. 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 spectral convolution is given by  $\mathcal{Y}_s = \mathcal{A}_s^{norm} * \mathcal{X}_s * \mathcal{W}_s$ , 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, which is exactly the T-spectral construction in Eq. (6).