Convolutional Graph-Tensor Net for Graph Data Completion

Xiao-Yang Liu[1] , **Ming Zhu**[2,3]
[1]Columbia University
[2]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences [3]All-In Tech Inc.
xl427@columbia.edu, zhumingpassional@gmail.com

Abstract

Graph data completion is a fundamentally important issue as data generally has a graph structure, e.g., social networks, recommendation systems, and the Internet of Things. We consider a graph where each node has a data matrix, represented as a *graph-tensor* by stacking the data matrices in the third dimension. In this paper, we propose a *Convolutional Graph-Tensor Net* (*Conv GT-Net*) for the graph data completion problem, which uses deep neural networks to learn the general transform of graph-tensors. The experimental results on the ego-Facebook data sets show that the proposed *Conv GT-Net* achieves significant improvements on both completion accuracy (50% higher) and completion speed (3.6x $\sim$ 8.1x faster) over the existing algorithms.

1 Introduction

Data with graph structures are common in real-world applications, e.g., user profiles in social networks, user-item matrices in recommendation systems, and sensory data in the Internet of Things [Liu and Wang, 2017b; Liu *et al.*, 2016]. Fig. 1 illustrates an example of user data with a graph structure in social networks or recommendation systems. Each user has a data matrix, and a *graph-tensor* is obtained by stacking the data matrices in the third dimension. Due to the limitations of the data collection/measurement process [Liu and Wang, 2017b], it is common that only a subset of data matrices are observed while the other data matrices are totally unobserved. The key problem is how to complete a graph-tensor with missing data matrices.

Several existing works are applied to the graph data completion problem. Tensor Alt-Min [Liu *et al.*, 2019] and tensor nuclear-norm minimization with the alternative direction method of multipliers (TNN-ADMM) [Zhang *et al.*, 2014] are proposed based on the low-rank structure of the transform-based tensor model, which can be applied to the graph-tensor completion problem. [Sun *et al.*, 2018] propose an iterative imputation algorithm for the graph-tensor completion problem, however, it is inadequate due to hundreds of iterations and the time-consuming shrinkage processes. The major challenge of the graph-tensor completion problem is to recover the missing data matrices by exploiting the graph topology.

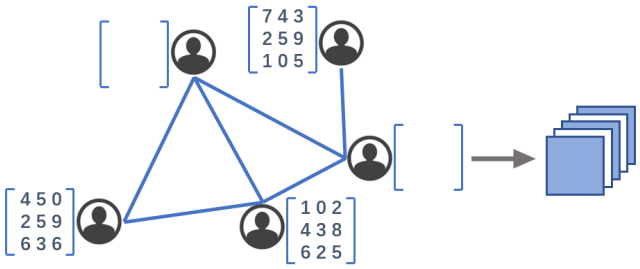

Figure 1: Example of user data with graph structure. A graph-tensor can be obtained by stacking user profile matrices in the third dimension.

2 Graph-Tensor Model and Problem Formulation

We represent a third-order tensor as $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, where the (i,j)-th tube is $\mathcal{X}(i,j,:)$ and the k-th frontal slice is $\mathcal{X}(:,:,k)$, or $\mathcal{X}^{(k)}$ for simplicity. We use $[n]$ to denote the set $\{1,...,n\}$. The Frobenius norm of a third-order tensor $\mathcal{X}$ is defined as $\|\mathcal{X}\|_F = \sqrt{\sum_{i=1}^{n_1}\sum_{j=1}^{n_2}\sum_{k=1}^{n_3}|\mathcal{X}(i,j,k)|^2}$.

2.1 Graph-Tensor Model

Consider an undirected graph $G = (V, E)$, where V denotes a set of nodes with $|V| = n_3$, and E denotes a set of edges. We assume that each node has a data matrix $\mathcal{X}^{(k)} \in \mathbb{R}^{n_1 \times n_2}$. After stacking the data matrices along the third dimension, we obtain a *graph-tensor* $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$.

Let $\boldsymbol{A} \in \mathbb{R}^{n_3 \times n_3}$ be the symmetric adjacency matrix of graph G, where the diagonal entries are zeros. The corresponding non-negative diagonal degree matrix $\boldsymbol{D} \in \mathbb{R}^{n_3 \times n_3}$ is $\boldsymbol{D}(k,k) = \sum_{s=1}^{n_3} \boldsymbol{A}(k,s)$, for $k \in [n_3]$. The graph Laplacian matrix of G is a real symmetric matrix $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A} \in \mathbb{R}^{n_3 \times n_3}$, and the normalized graph Laplacian matrix of G is $\boldsymbol{L}_n = \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}} \in \mathbb{R}^{n_3 \times n_3}$, where $\boldsymbol{D}^{-\frac{1}{2}}$ is calculated in an element-wise manner on the diagonal entries $\boldsymbol{D}(k,k)$ for $k \in [n_3]$.

Let the eigenvalue decomposition of $\boldsymbol{L}$ be $\boldsymbol{L} = \boldsymbol{Q}\boldsymbol{E}\boldsymbol{Q}^H$, where $\boldsymbol{Q} \in \mathbb{R}^{n_3 \times n_3}$ consists of the eigenvectors, $\boldsymbol{E} \in$

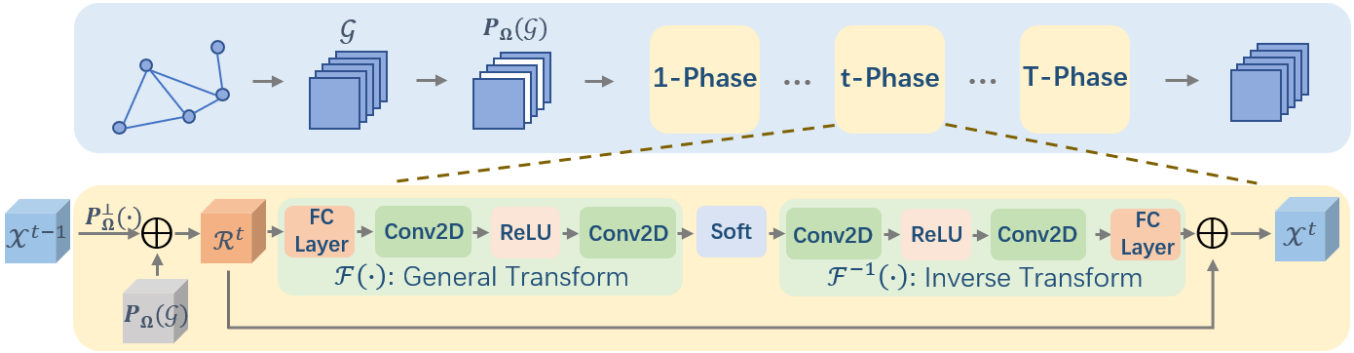

Figure 2: The structure of *Conv GT-Net*. The upper part is the data flow in *Conv GT-Net*, containing T phases. The bottom part is the detailed structure of one phase. Gray arrows denote the data flow.

$\mathbb{R}^{n_3\times n_3}$ is a diagonal matrix with the corresponding eigenvalues, and H is the Hermitian transpose. The graph transform matrix $\boldsymbol{U} \in \mathbb{R}^{n_3\times n_3}$ is a unitary matrix defined as $\boldsymbol{UL} = \boldsymbol{EU}$, and $\boldsymbol{U} = \boldsymbol{Q}^H$.

Any linear transform can be the transform $\mathcal{L}$ in [Liu and Wang, 2017a]. In this paper, we use the graph Fourier transform as the transform $\mathcal{L}$, which is performed by the matrix $\boldsymbol{U}$. Let $\widetilde{\mathcal{X}}$ denote the graph-tensor in the spectral domain for $\mathcal{X}$. The graph Fourier transform and the inverse graph Fourier transform can be defined as follows:

$$\widetilde{\mathcal{X}}^{(k)} = \sum_{s\in[n_3]} \boldsymbol{U}(k,s)\mathcal{X}^{(s)}, \text{ for } k \in [n_3], \tag{1}$$

$$\mathcal{X}^{(k)} = \sum_{s\in[n_3]} \boldsymbol{U}^{-1}(k,s)\widetilde{\mathcal{X}}^{(s)}, \text{ for } k \in [n_3], \tag{2}$$

where $\boldsymbol{U}^{-1}$ is the inverse matrix of $\boldsymbol{U}$.

2.2 Problem Formulation

We assume that the graph-tensor in the spectral domian is low rank, which has been supported in many research works, such as the spectral space singular value distribution [Sun *et al.*, 2018]. We assume that a subset of data matrices of a graph-tensor are observed. For example, for a graph-tensor $\mathcal{G} \in \mathbb{R}^{n_1\times n_2\times n_3}$ with n_3 nodes, the observed data matrices are $P_\Omega(\mathcal{G})$, where $P_\Omega : \mathbb{R}^{n_1\times n_2\times n_3} \to \mathbb{R}^{n_1\times n_2\times n_3}$ is the projection of the tensor to a partial observation by only retaining entries in the index set $\Omega \in [n_3]$. Let $\mathcal{X}$ be a graph-tensor which has the same partial observation as $P_\Omega(\mathcal{G})$. For a graph-tensor $\mathcal{X} \in \mathbb{R}^{n_1\times n_2\times n_3}$, the $\mathcal{L}$-SVD [Liu and Wang, 2017a] is $\mathcal{X} = \mathcal{U} \bullet \mathcal{S} \bullet \mathcal{V}^\dagger$. We define its graph-tensor nuclear-norm as $\|\mathcal{X}\|_{\text{gTNN}} = \langle \mathcal{S}, \mathcal{I}\rangle = \sum_{i=1}^r |\widetilde{\mathcal{S}}(i,i,1)|$, where $r = \text{rank}_{\mathcal{L}}(\mathcal{X})$.

To recover the data matrices of the unobserved nodes, the graph-tensor completion problem is formulated as:

$$\begin{aligned} &\min_{\widetilde{\mathcal{X}}} \|\widetilde{\mathcal{X}}\|_{\text{gTNN}}, \\ &\text{s.t. } \mathcal{P}_\Omega(\mathcal{X}) = \mathcal{P}_\Omega(\mathcal{G}). \end{aligned} \tag{3}$$

Let $\mathcal{P}_\Omega^\perp$ be the complement of $\mathcal{P}_\Omega$, and $\boldsymbol{A} = \boldsymbol{V_1}\Sigma\boldsymbol{V_2}^H$ be the singular value decomposition. Let singular-soft$(\cdot)$ denote the diagonal soft-thresholding operator on the singular value vector of the matrix: singular-soft$(\boldsymbol{A}, \lambda) = \boldsymbol{V_1}(\Sigma - \lambda\boldsymbol{I})_+\boldsymbol{V_2}^H$ with $(\cdot)_+$ keeping only positive values.

Problem (3) can be solved via the iterative imputation algorithm in [Sun *et al.*, 2018]. Its key steps are as follows for $t \geq 1$ and $k \in [n_3]$:

$$\mathcal{R}^t = \mathcal{P}_\Omega(\mathcal{G}) + \mathcal{P}_\Omega^\perp(\mathcal{X}^{t-1}), \tag{4}$$

$$\widetilde{\mathcal{R}}^{t(k)} = \sum_{s=1}^{n_3} \boldsymbol{U}(k,s)\mathcal{R}^{t(s)}, \tag{5}$$

$$\widetilde{\mathcal{X}}^{t(k)} = \text{singular-soft}(\widetilde{\mathcal{R}}^{t(k)}, \boldsymbol{\lambda}(k)), \tag{6}$$

$$\mathcal{X}^{t(k)} = \sum_{s=1}^{n_3} \boldsymbol{U}^{-1}(k,s)\widetilde{\mathcal{X}}^{t(s)}, \tag{7}$$

where $\mathcal{X}^0 = \boldsymbol{0}$.

3 Convolutional Graph-Tensor Net

We propose a *Convolutional Graph-Tensor Net* (*Conv GT-Net*) for the graph data completion problem in this section. The basic idea is to map the imputation algorithm steps in (4)-(7) into deep neural networks with a fixed number of phases, where each phase corresponds to one iteration. We assume that there are T phases in the *Conv GT-Net* with the same parameters. Fig. 2 shows the structure of *Conv GT-Net*.

1) Imputation. Step (4) aims to keep the observed nodes the same as their ground truth. We use the same operation in (4) as the imputation results in each phase of *Conv GT-Net*.

2) General Transform. Step (5) uses graph Fourier transform in (1) to obtain the low-rank data in the spectral domain. Considering the great representation power of the convolutional neural networks, we adopt one fully connected (FC) layer, two 2D convolution layers and one ReLU activation layer as the general transform structure, denoted by $\mathcal{F}(\cdot) = \text{Conv2D}(\text{ReLU}(\text{Conv2D}(\text{FC}(\cdot))))$. For simplicity, we use 3×3 filters in convolution layers. $\mathcal{F}(\cdot)$ aims to learn the general transform of the graph-tensor. The graph-tensor $\mathcal{R}^t$ in the spectral domain is denoted by $\mathcal{F}(\mathcal{R}^t)$.

3) Soft-Thresholding Operator. We replace (6) by the soft-thresholding operator in [Donoho, 1995]. Unlike the diagonal soft-thresholding operator, the soft-thresholding operator is defined as soft$(a,\lambda) = \max(|a| - \lambda, 0)\cdot \text{sign(a)}$,

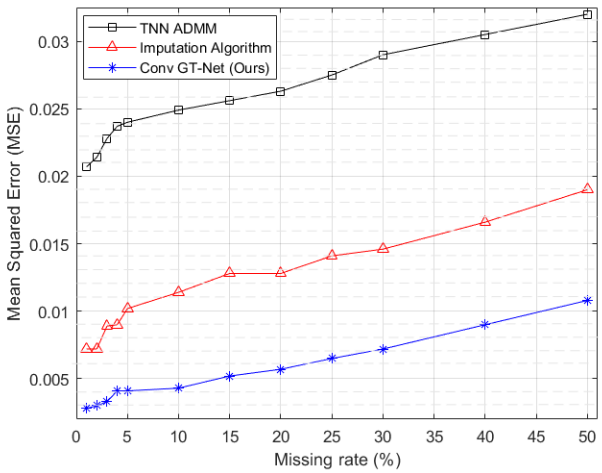

Figure 3: Experimental results of the unweighted graph-tensor. Completion accuracy vs. missing rate.

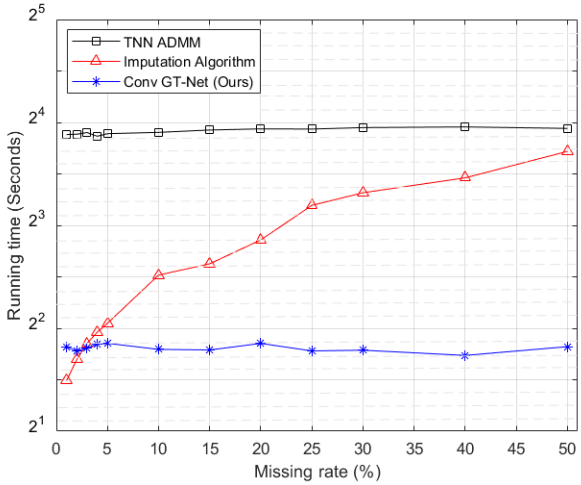

Figure 4: Experimental results of the unweighted graph-tensor. Running time vs. missing rate.

where sign$(\cdot)$ is the sign of a. It is an element-wise operator on each entry when applied to a matrix or tensor. Deep neural networks have great learning ability, so we set the soft-thresholding parameter set $\boldsymbol{\lambda} \in [T]$ to be learnable instead of using the pre-set ones in [Sun *et al.*, 2018]. The graph-tensor after the soft-thresholding operator is denoted by soft$(\mathcal{F}(\mathcal{R}^t), \boldsymbol{\lambda})$.

4) Inverse Transform. We introduce an inverse transform structure $\mathcal{F}^{-1}(\cdot)$ as the inversion of the general transform to replace step (7), i.e. $\mathcal{F}^{-1}(\mathcal{F}(\mathcal{X})) = \mathcal{X}$. Specifically, $\mathcal{F}^{-1}(\cdot)$ adopts the symmetric structure of $\mathcal{F}(\cdot)$ and $\mathcal{F}^{-1}(\cdot) = \text{FC}(\text{Conv2D}(\text{ReLU}(\text{Conv2D}(\cdot))))$. The filters in $\mathcal{F}^{-1}(\cdot)$ share the same size as those in $\mathcal{F}(\cdot)$. The graph-tensor after all the above operations is $\mathcal{F}^{-1}(\text{soft}(\mathcal{F}(\mathcal{R}^t), \boldsymbol{\lambda}))$.

Shortcut Structure. Inspired by the residual network [He *et al.*, 2016], we add a shortcut structure before $\mathcal{X}^t$ to make the network more robust:

$$\mathcal{X}^t = \mathcal{R}^t + \mathcal{F}^{-1}(\text{soft}(\mathcal{F}(\mathcal{R}^t), \boldsymbol{\lambda}). \tag{8}$$

Loss Function. Two items should be considered in *Conv GT-Net*: the fidelity of the reconstructed data, and the accuracy of the inverse function. Loss function $\mathcal{L}$ is the weighted summation of two terms:

$$\mathcal{L} = \alpha \mathcal{L}_{\text{fidelity}} + \beta \mathcal{L}_{\text{inversion}}, \tag{9}$$

$$\mathcal{L}_{\text{fidelity}} = \|P_{\Omega}^{\perp}(\mathcal{X}^T) - P_{\Omega}^{\perp}(\mathcal{G})\|_F^2, \tag{10}$$

$$\mathcal{L}_{\text{inversion}} = \frac{1}{T}\sum_{t=1}^{T} \|\mathcal{F}^{-1}(\mathcal{F}(\mathcal{X}^t)) - \mathcal{X}^t\|_F^2. \tag{11}$$

Referring to [Zhang and Ghanem, 2018] and considering the case in our work, we set $\alpha = 1$ and $\beta = 0.0001$.

4 Performance Evaluation

The comparison algorithms are TNN-ADMM [Zhang *et al.*, 2014] and imputation algorithm [Sun *et al.*, 2018], which do not need training and are implemented using MATLAB. *Conv GT-Net* is trained in TensorFlow on a server with an NVIDIA Telsa V100 GPU (16GB device memory). For a fair comparison, we perform all the experiments on a laptop with an i7-8750H CPU (2.20GHz) and 16GB memory. The phase number T is 10, the learning rate is 0.0001, and the running epoch is 500. Adam optimizer [Kingma and Ba, 2014] is used to minimize the loss function.

We take the graph topology of the ego-Facebook data sets from SNAP [Leskovec and Krevl, 2014], which contain an unweighted and undirected graph with 4,039 nodes and 88,234 edges. We pick 100 nodes (No. 896-995) and their corresponding edges to form a graph topology. To evaluate the performance in graph data with weighted edges, we randomly assign a weight (1-20) to each edge to form another graph topology. We use the feature data of the ego-Facebook data sets, select the nodes whose feature vector has 576 features and resize the feature vectors into 24×24 feature matrices. We regard the feature matrices as the data in the spectral domain and perform the inverse graph Fourier transform to get the graph-tensor in the time domain. Consequently, the graph-tensor has a size of $24 \times 24 \times 100$, and we select nodes and their feature data to form 32 graph-tensors as testing data and another 900 graph-tensors as training data.

We take mean squared error (MSE) and running time as metrics. Fig. 3 shows the completion accuracy of the unweighted graph-tensors. We observe that *Conv GT-Net* provides an average of 50% higher completion accuracy than the imputation algorithm, let alone TNN-ADMM. This is mainly due to the strong low-rank assumption of the imputation algorithm since the real data is usually not strictly low-rank in the spectral domain. However, *Conv GT-Net* can learn a general transform for real data, which alleviates the dependency on the low-rank assumption. TNN-ADMM provides the lowest accuracy since it omits the graph topology.

Fig. 4 shows the running time of the unweighted graph-tensors. We observe that *Conv GT-Net* runs about 3 to 8 times faster than the imputation algorithm and about 4 times

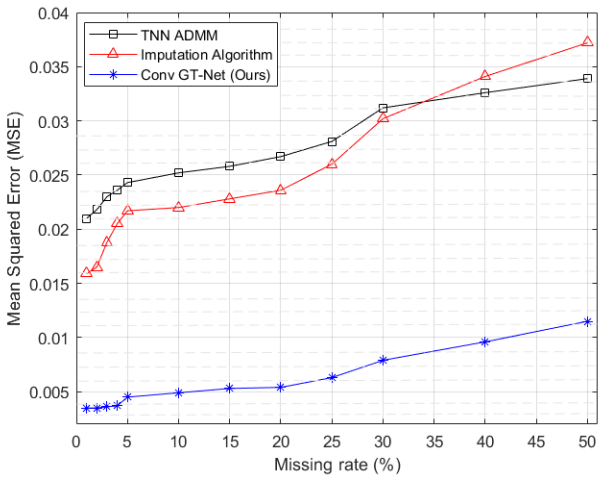

Figure 5: Experimental results of the weighted graph-tensor. Completion accuracy vs. missing rate.

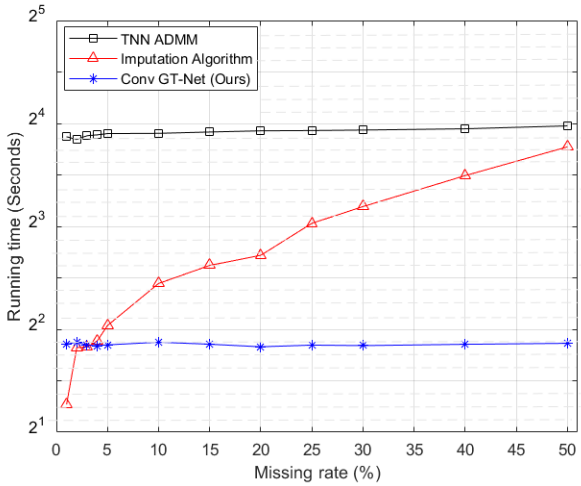

Figure 6: Experimental results of the weighted graph-tensor. Running time vs. missing rate.

faster than TNN-ADMM. It is worth noting that for different missing rates, *Conv GT-Net* and TNN-ADMM keep a constant running speed, whereas the imputation algorithm [Sun *et al.*, 2018] needs more running time as the missing rate increases. This is because *Conv GT-Net* adopts a fixed structure for a certain size of graph-tensors and TNN-ADMM omits the graph topology, while the imputation algorithm needs more iterations to converge as the missing rate increases.

Fig. 5 and Fig. 6 show the completion accuracy and running time of the weighted graph-tensor, respectively. We notice that *Conv GT-Net* and TNN-ADMM are less influenced by the weight change of the graph while the imputation algorithm is more influenced. This is because the imputation algorithm performs graph Fourier transform which takes the weights information into consideration.

5 Conclusion

In this paper, we proposed *Conv GT-Net* to solve the graph-tensor completion problem using deep neural networks. It learns a general transform of a graph-tensor. The experimental results show that *Conv GT-Net* provides higher completion accuracy and runs faster than the other algorithms.

Acknowledgment

Ming Zhu was supported by National Natural Science Foundations of China (Grant No. 61902387).

References

[Donoho, 1995] David L. Donoho. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, May 1995.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE CVPR*, pages pp. 770–778, June 2016.

[Kingma and Ba, 2014] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Leskovec and Krevl, 2014] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, 2014.

[Liu and Wang, 2017a] Xiao Yang Liu and Xiaodong Wang. Fourth-order tensors with multidimensional discrete transforms. 2017.

[Liu and Wang, 2017b] Xiao-Yang Liu and Xiaodong Wang. LS-decomposition for robust recovery of sensory big data. *IEEE Transactions on Big Data*, 4(4):542–555, 2017.

[Liu *et al.*, 2016] Xiao-Yang Liu, Shuchin Aeron, Vaneet Aggarwal, Xiaodong Wang, and Min-You Wu. Tensor completion via adaptive sampling of tensor fibers: Application to efficient indoor rf fingerprinting. In *IEEE ICASSP*, pages 2529–2533, 2016.

[Liu *et al.*, 2019] Xiao-Yang Liu, Shuchin Aeron, Vaneet Aggarwal, and Xiaodong Wang. Low-tubal-rank tensor completion using alternating minimization. In *IEEE Transactions on Information Theory*, 2019.

[Sun *et al.*, 2018] Qingyun Sun, Mengyuan Yan, David Donoho, and stephen boyd. Convolutional imputation of matrix networks. In *ICML*, volume 80, pages 4818–4827, 2018.

[Zhang and Ghanem, 2018] Jian Zhang and Bernard Ghanem. ISTA-Net: Interpretable optimization-inspired deep network for image compressive sensing. In *IEEE CVPR*, pages 1828–1837, June 2018.

[Zhang *et al.*, 2014] Zemin Zhang, Gregory Ely, Shuchin Aeron, Ning Hao, and Misha Kilmer. Novel methods for multilinear data completion and de-noising based on tensor-svd. In *IEEE CVPR*, pages 3842–3849, 2014.