CoSTCo: A Neural Tensor Completion Model for Sparse Tensors

Hanpeng Liu
University of Southern California
Los Angeles, California
hanpengl@usc.edu

Yaguang Li
University of Southern California
Los Angeles, California
yaguang@usc.edu

Michael Tsang
University of Southern California
Los Angeles, California
tsangm@usc.edu

Yan Liu
University of Southern California
Los Angeles, California
yanliu.cs@usc.edu

ABSTRACT

Low-rank tensor factorization has been widely used for many real world tensor completion problems. While most existing factorization models assume a multilinearity relationship between tensor entries and their corresponding factors, real world tensors tend to have more complex interactions than multilinearity. In many recent works, it is observed that multilinear models perform worse than nonlinear models. We identify one potential reason for this inferior performance: the nonlinearity inside data obfuscates the underlying low-rank structure such that the tensor *seems* to be a high-rank tensor. Solving this problem requires a model to simultaneously capture the complex interactions and preserve the low-rank structure. In addition, the model should be scalable and robust to missing observations in order to learn from large yet sparse real world tensors. We propose a novel convolutional neural network (CNN) based model, named CoSTCo (**Co**nvolutional **S**parse **T**ensor **Co**mpletion). Our model leverages the expressive power of CNN to model the complex interactions inside tensors and its parameter sharing scheme to preserve the desired low-rank structure. CoSTCo is scalable as it does not involve computation- or memory- heavy tasks such as Kronecker product. We conduct extensive experiments on several real world large sparse tensors and the experimental results show that our model clearly outperforms both linear and nonlinear state-of-the-art tensor completion methods.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; **Neural networks**; **Factorization methods**.

KEYWORDS

Sparse Tensor Completion, Nonlinear Tensor factorization, Convolutional neural network

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330881

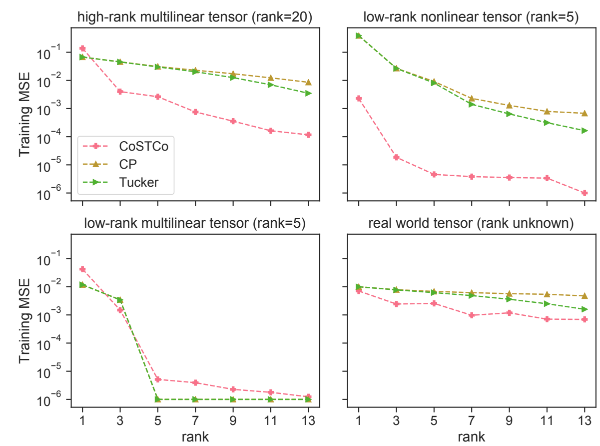


Figure 1: Rank-vs-loss curves of four tensor completion experiments with fully observations. The curves of CP/Tucker methods on the low-rank nonlinear tensor is different to those on low-rank multilinear tensors but similar to those on high-rank multilinear tensors[1].

ACM Reference Format:
Hanpeng Liu, Yaguang Li, Michael Tsang, and Yan Liu. 2019. CoSTCo: A Neural Tensor Completion Model for Sparse Tensors. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3292500.3330881

1 INTRODUCTION

A tensor is an effective way of representing multidimensional real world data. Due to the issue of sparsity or incompleteness in many real world tensors, *tensor completion*, which predicts the missing entries of a partially observed tensor, has gained wide interest and serves as a fundamental problem of many machine learning applications, such as recommendation [11, 33], image in-painting [24, 28], spatial-temporal analysis [43], and health data analysis [15–17]. Low-rank tensor factorization is a popular method for tensor completion. This method assumes a compact underlying structure in the tensor such that its entries can be reconstructed from low-rank factor matrices through multilinear multiplication. Thus we can

[1] We clip the values for better visualization so that the minimum value is 10^{-6}.

complete a tensor by first finding latent factors through observed entries and then reconstructing the tensor with these factors[2]. CP (CANDECOMP/PARAFAC) [14] and Tucker [40] are two popular low-rank tensor factorization models with many variants being developed.

Recently, a series of works [8, 11, 15, 16, 42, 48] have shown that nonlinear factorization models have superior performances over multilinear models for sparse tensor completion. It is believed that real world tensors tend to have interactions that are more complex than the multilinear multiplication assumed by the CP and Tucker models. We hypothesize that multilinear models have inferior performance because the nonlinearity in data obfuscates the low-rank property, i.e. a low-rank nonlinear tensor seems to be a high-rank one for multilinear models. This obfuscation prevents the low-rank model from fitting the data well. To examine this hypothesis, we train factorization models to learn the latent factors of fully observed tensors and measure the reconstruction losses. Figure 1 shows the results of three synthetic CP tensors[3] and one real world tensor. Both CP and Tucker find good latent factors and achieve very small reconstruction error on the low-rank multilinear tensor. However, in the case of a low-rank *nonlinear* tensor, both CP and Tucker model behave similarly to those of a high-rank multilinear tensor. The real world tensor is much more complex and the loss curves of CP and Tucker decrease slowly. In comparison, our proposed model `CoSTCo` fits the nonlinear tensor well at a low rank and the loss curve of our model decreases faster compared to CP and Tucker.

To capture complex interactions within a tensor, the model must have strong expressive power to capture the underlying nonlinearity. In addition, the model needs to be parameter efficient to avoid overfitting to the sparse and limited training data and thus to achieve good generalization performance.

In this paper, we propose a novel nonlinear low-rank model, called `CoSTCo` (**Co**nvolutional **S**parse **T**ensor **Co**mpletion), to do tensor completion for large sparse tensors that have complex underlying structures. With the expressive power of CNNs, our model is capable of learning a complex nonlinear interaction function among factors. The parameter sharing scheme of CNN imposes an inductive bias on `CoSTCo` which enforces it to learn a parameter-efficient representation of the nonlinear function and informative factor vectors. The scheme helps the model to preserve the low-rank structure and makes the model robust to sparse inputs, which is crucial for generalization. Like other neural networks, `CoSTCo` can be efficiently trained with Stochastic Gradient Descent (SGD). The model does not require computation/memory heavy operations in the training process, such as Kronecker product or Gram matrix. Furthermore, `CoSTCo` scales well with the mini-batch training scheme of modern neural nets which avoids explicit storage of the whole tensor. We also draw connections from recent advances [7, 13] in deep learning theory and show that a slightly simplified variant of `CoSTCo` falls into the category of shallow neural nets [7], which has been proven to have fast convergence rate and bounded generalization error.

[2]We use factors and factor matrices interchangeably wherever the context is clear.
[3]The nonlinear tensor is generated by replacing the multiplication funciton inside original CP factorization with a non-linear function.

We evaluate the proposed model, `CoSTCo`, with extensive experiments on real-world sparse tensors. Our model consistently outperforms both linear and nonlinear state-of-the-art tensor completion models. Experiments also show that `CoSTCo` scales well with large tensors and is able to learn meaningful representations while being robust to data sparsity. Besides superior tensor completion performance, `CoSTCo` also outperforms state-of-the-art methods for downstream tasks, such as Point-of-interest (POI) recommendation and spatiotemporal forecasting. We also show `CoSTCo` can learn meaningful factors that fit intuitions and prior knowledge. In summary, our main contributions are as follows:

- We propose a novel nonlinear model for tensor completion based on convolutional neural network that can learn nonlinear interactions, leverage a low-rank inductive bias, and scale well with large tensors.
- We conduct extensive tensor completion experiments on six real world tensors and show that our model has superior performance over state-of-the-art multilinear/nonlinear tensor completion algorithms.
- We show that our method is capable of outperforming state-of-the-art methods for downstream tasks, i.e., point-of-interest recommendation and spatiotemporal forecasting.

2 RELATED WORK

Low-rank tensor completion. CANDECOMP/PARAFAC factorization (CP) [14] and Tucker factorization [40] are classical tensor factorization methods, which many low-rank tensor completion algorithms are based on, such as convex-relaxed tensor completion [12, 28, 35], completion models for non-negative tensors [36], scalable algorithms for completing incomplete large tensors [1, 4, 18], completions for streaming tensors [38], completions that can utilize extra information or auxiliary tensors [17, 30]. Different from these multilinear models, our model is designed to do nonlinear sparse tensor completion.

Nonlinear tensor factorization. Recently, nonlinear tensor factorization has gained attention due to its effectiveness at modeling real world complex tensors. In [15, 16], the authors introduced a series of kernel based CP models that can capture nonlinear relationships. Unlike our `CoSTCo`, these models are not designed for the sparse tensor completion task and require full observations of the target tensor. In [22], the authors developed nonlinear matrix factorization algorithm based on Gaussian processes. In [11], the authors proposed to use Gaussian kernels to do tensor factorization. In [42], the authors proposed an InfTucker model with nonlinearity by extending probabilistic Tucker model [5] with a nonlinear covariance function. The InfTucker model cannot scale to large tensors because of the Kronecker product involved in the training process. A series of scalable models based on InfTucker are proposed in [47, 48] by paralleling the computation of the Kronecker product or by avoiding the Kronecker product altogether. Compared to this line of works, our model neither requires a pre-specified kernel function nor involves computational expensive optimization procedures.

Neural network matrix/tensor completion. Our `CoSTCo` model is distinguished from other neural approaches on 1) its use of convolution to accurately and efficiently perform tensor completion and 2)

Table 1: Common symbols used in the paper.

Symbol	Description
N	tensor order
d_i	tensor size in the i-th dimension
$\mathcal{T} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$	a tensor of order-N and shape $d_1 \times \cdots \times d_N$
$\mathcal{T}_{i_1,\ldots,i_N}$	tensor entry value at index $(i_1, \ldots, i_N)$
x, $\mathbf{x}$, $\mathbf{X}$	a scalar, a vector, a matrix
$\mathbf{X}_{i,:}$, $\mathbf{X}_{:,j}$	i-th row, j-th column of matrix $\mathbf{X}$
$\mathcal{A} * \mathcal{B}$	element-wise tensor multiplication
$\|\mathcal{T}\|_F^2$	Frobenius norm of tensor $\mathcal{T}$
$J_{\mathcal{T}}$	the set of all entry indices of the tensor $\mathcal{T}$

its capability of representing general tensors that are order-2, order-3, or higher. For example, many works [8, 27, 41] have proposed to replace the linear/multilinear operations in matrix/tensor factorization with multi-layer perceptrons (MLP). However, MLP-based methods tend to overfit to sparse training data as we will show later in the experiments section. Another work [19] developed a matrix completion model which uses CNN to extract features from text documents for get better semantic features. However, this model didn't use CNN for the factorization/completion. [37] combined neural nets and tensors for knowledge graph reasoning. Compared to these models, our model is more general and not restricted for order-3 knowledge graph tensors.

3 PRELIMINARY

3.1 Notations

Table 1 shows commonly used symbols in the paper following the notations of [21]. We use $\mathbf{e}_d(i) \in \mathbb{R}^d$ to denote a one-hot vector of length-d, where its i-th entry is 1 and all others are 0. For example, $\mathbf{e}_5(1) = [1 \quad 0 \quad 0 \quad 0 \quad 0]^T$. For an order-$N$ tensor $\mathcal{T} \in \mathbb{R}^{d_1 \times \ldots \times d_N}$, we use a vector $\mathbf{e}_{d_1 \ldots d_N}(i, \ldots, i_N)$

$$\mathbf{e}_{d_1 \ldots d_N}(i_1, \ldots, i_N) = \begin{bmatrix} \mathbf{e}_{d_1}(i_1) \\ \vdots \\ \mathbf{e}_{d_N}(i_N) \end{bmatrix} \in \mathbb{R}^{d_1 + \cdots + d_N}, \tag{1}$$

as the generalized one-hot embedding of the index tuple $(i_1, \ldots, i_N)$ of $\mathcal{T}$, called as **indexing vectors**. We use $\mathbf{e}(i_1, \ldots, i_N)$ as the abbreviation of $\mathbf{e}_{d_1 \ldots d_N}(i_1, \ldots, i_N)$ whenever appropriate.

3.2 Tensor Factorization

For simplicity and without loss of generality, we discuss formulations using order-3 tensors. In general, a tensor factorization model learns the low rank factor matrices from observed entries, and reconstructs the tensor $\mathcal{T}'$ from them. The goal is to find accurate factor matrices such that the reconstructed tensor $\hat{\mathcal{T}}$ is close to the target tensor $\mathcal{T}$, i.e., its Frobenius norm, $||\hat{\mathcal{T}} - \mathcal{T}||_F^2$, is minimized. The Frobenius norm of a tensor is defined as

$$\|\mathcal{T}\|_F^2 = \sum_{i_1, i_2, i_3} \mathcal{T}_{i_1, i_2, i_3}{}^2. \tag{2}$$

Two major factorization models are CP model and Tucker model.

CP model. Given an order-3 tensor $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$, a rank-$r$ CP model factorizes it into three factor matrices: $\mathbf{U}^1 \in \mathbb{R}^{r \times d_1}, \mathbf{U}^2 \in \mathbb{R}^{r \times d_2}, \mathbf{U}^3 \in \mathbb{R}^{r \times d_3}$, so that the predicted tensor entry is

$$\hat{\mathcal{T}}_{i,j,k} = \sum_{t=1}^{r} \mathbf{U}^1{}_{t,i} \mathbf{U}^2{}_{t,j} \mathbf{U}^3{}_{t,k}.$$

The column vectors $\mathbf{U}^1{}_{:,i}$, $\mathbf{U}^2{}_{:,j}$, $\mathbf{U}^3{}_{:,k}$ are called as factor vectors.

Tucker model. Given an order-3 tensor $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$, a rank-$(r_1, r_2, r_3)$ Tucker model factorizes it into one core tensor $\mathcal{G} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ together with three factor matrices: $\mathbf{U}^1 \in \mathbb{R}^{r \times d_1}$, $\mathbf{U}^2 \in \mathbb{R}^{r \times d_2}$, $\mathbf{U}^3 \in \mathbb{R}^{r \times d_3}$, such that the predicted tensor entry is given as

$$\hat{\mathcal{T}}_{i,j,k} = \sum_{t_1=1}^{r_1} \sum_{t_2=1}^{r_2} \sum_{t_3=1}^{r_3} \mathcal{G}_{t_1, t_2, t_3} \mathbf{U}^1{}_{t_1, i} \mathbf{U}^2{}_{t_2, j} \mathbf{U}^3{}_{t_3, k}.$$

3.3 Nonlinear Tensor Factorization Formulation

Definition 3.1 (Generalized Tensor Factorization Model). Given an order-N tensor $\mathcal{T} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$, denote $J_{\mathcal{T}} = \{(i_1, \ldots, i_N) | \forall i_1 \in \{1, ..., d_1\}, \ldots, i_N \in \{1, ..., d_N\}\}$ as the set of the indices of all entries in the tensor. A rank-$(r_1, \ldots, r_N)$ tensor factorization model consists of a mapping function $f : J_{\mathcal{T}} \rightarrow \mathbb{R}$ with parameter θ and N factor matrices $\mathbf{U}^1 \in \mathbb{R}^{r_1 \times d_1}, \ldots, \mathbf{U}^N \in \mathbb{R}^{r_N \times d_N}$. The reconstructed tensor $\hat{\mathcal{T}}$ can be represented as

$$\hat{\mathcal{T}}_{i_1,\ldots,i_N} = f(i_1, \ldots, i_N; \mathbf{U}^1, \ldots, \mathbf{U}^N, \theta). \tag{3}$$

Definition 3.2 (Multi-linear Tensor Factorization Model). Assume a tensor factorization model has a mapping function f with parameter θ and factor matrices $\mathbf{U}^1, \ldots \mathbf{U}^N$. It is a multi-linear model if the following equations hold for any $(i_1, \ldots, i_N) \in J_{\mathcal{T}}$

$$\left\| \frac{\partial^2 f(i_1, \ldots, i_N; \mathbf{U}^1, \ldots, \mathbf{U}^N, \theta)}{\partial^2 \mathbf{U}^1{}_{:, i_1}} \right\|_2 = 0,$$
$$\ldots,$$
$$\left\| \frac{\partial^2 f(i_1, \ldots, i_N; \mathbf{U}^1, \ldots, \mathbf{U}^N, \theta)}{\partial^2 \mathbf{U}^N{}_{:, i_N}} \right\|_2 = 0.$$

Definition 3.3 (Nonlinear Tensor Factorization Model). A tensor factorization model is nonlinear if its mapping function f is not multi-linear.

4 METHODOLOGY

4.1 The CoSTCo Model

The core structure of CoSTCo is a convolutional neural network. For a order-N tensor of shape $(d_1, \ldots, d_N)$, the input of CoSTCo neural network is an indexing vector $\mathbf{x} = \mathbf{e}_{d_1 \ldots d_N}(i_1, \ldots, i_N)$ and the output of the neural network is the reconstructed tensor entry $\hat{\mathcal{T}}_{i_1,\ldots,i_N}$. The neural network contains an embedding module, an nonlinear mapping module, and final aggregation module. The model's architecture is shown in Figure 2.

Embedding module. For a tensor entry at index $(i_1, \ldots, i_N)$, the embedding module takes an indexing vector $\mathbf{x}$ as the input, and outputs the corresponded factor vectors of these indices. The embedding module has a block diagonal matrix $\mathbf{M}$ with dimension $Nr \times \sum_{i=1}^{N} d_i$, consisting of the N factor matrices, namely $\mathbf{U}^1, \ldots, \mathbf{U}^N$.

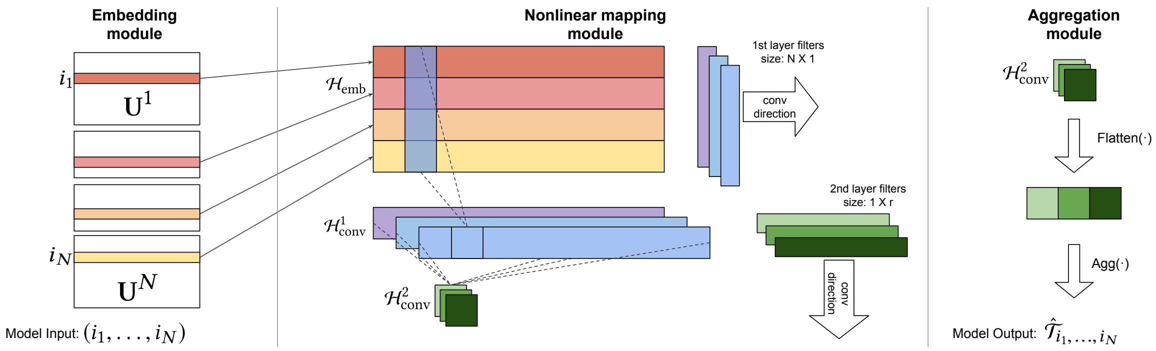


Figure 2: Model architecture of CoSTCo.

The same rank r is used acrossed all factor matrices. More specifically, we have

$$\mathcal{H}'_{\text{emb}} = \mathbf{Mx} = \mathbf{Me}_{d_1,d_2,\ldots,d_N}(i_1, i_2, \ldots, i_N)$$

$$= \begin{bmatrix} \mathbf{U}^1 & & & \\ & \mathbf{U}^2 & & \\ & & \ddots & \\ & & & \mathbf{U}^N \end{bmatrix} \begin{bmatrix} \mathbf{e}_{d_1}(i_1) \\ \mathbf{e}_{d_2}(i_2) \\ \vdots \\ \mathbf{e}_{d_N}(i_N) \end{bmatrix} = \begin{bmatrix} \mathbf{U}^1{}_{:,i_1} \\ \mathbf{U}^2{}_{:,i_2} \\ \vdots \\ \mathbf{U}^N{}_{:,i_N} \end{bmatrix} \quad (4)$$

$$\mathcal{H}'_{\text{emb}} = \text{Reshape}(\mathcal{H}'_{\text{emb}}) \in \mathbb{R}^{1\times r\times N}$$

The module's output $\mathcal{H}_{\text{emb}}$ is constructed by reshaping a vector $\mathcal{H}'_{\text{emb}} \in \mathbb{R}^{rN\times 1}$ into a $1 \times r \times N$ tensor.

Nonlinear mapping module. The mapping module consists of two 2D convolution layers, one with filter size $(r, 1)$ and another with filter size $(1, N)$. Let C be the number of channels in the convolutional layers and θ_1, θ_2 are convolution filters of the two convolution layers. Both layers have nonlinear activation function $\sigma(\cdot)$. For example, if we choose the ReLU function, then $\sigma = \max(\cdot, 0)$. The output of convolution module is [4]

$$\begin{aligned} \mathcal{H}^1_{\text{conv}} &= \sigma(\text{Conv}(\mathcal{H}_{\text{emb}}; \theta_1)) \in \mathbb{R}^{C\times 1\times N} \\ \mathcal{H}^2_{\text{conv}} &= \sigma(\text{Conv}(\mathcal{H}^1_{\text{conv}}; \theta_2)) \in \mathbb{R}^{C\times 1\times 1} \end{aligned} \quad (5)$$

Aggregation module. The aggregation module takes an $C \times 1 \times 1$ tensor as its input, flattens it into a length-C vector, aggregates it by some predefined aggregation function, and outputs one scalar. The aggregation function can be a fully-connected neural network layer, or simply outputs the average of the input vector. Denote θ_a as the potential parameter of the aggregation module. The output of aggregation module can be written as

$$y = \hat{\mathcal{T}}_{i_1,\ldots,i_N} = \text{Agg}(\mathcal{H}^2_{\text{conv}}; \theta_a) \in \mathbb{R}. \quad (6)$$

In summary, a rank-r CoSTCo model is a generalized tensor factorization model represented by a neural network, which stores the factor matrices in the embedding module, and has a mapping function $y = f(x; \mathbf{U}^1, \ldots, \mathbf{U}^N, \{\theta_1, \theta_2, \theta_a\})$ composed of the mapping and aggregation module defined by eqs. (5) and (6). We use $f(x; \cdot)$ for simplicity whenever possible.

[4] We use the channel-first notation for 2d convolution operations.

4.2 Optimization and Generalization of CoSTCo

Suppose the target tensor is $\mathcal{T}$ and the reconstructed one is $\hat{\mathcal{T}}$. Let the training set be $S = \{(x, y) | x \in J_{\mathcal{T}}, y \in \mathbb{R}\}$ where x is an index and y is the observed tensor entry. The loss function of a reconstructed tensor $\hat{\mathcal{T}}$ is the Frobenius norm of the difference between it and the target tensor $\mathcal{T}$, $\left\|\hat{\mathcal{T}} - \mathcal{T}\right\|_F^2$. In the sparse observation setting where only parts of the tensor entries are observed, the loss function is $\left\|\mathcal{O} * (\hat{\mathcal{T}} - \mathcal{T})\right\|_F^2$. The value of $\mathcal{O}_{i_1,\ldots,i_N}$ equals to 1 if the index $(i_1, \ldots, i_N)$ is observed in S and otherwise 0. The loss function is the same as the mean squared error.

$$\text{Loss} = \left\|\mathcal{O} * (\hat{\mathcal{T}} - \mathcal{T})\right\|_F^2 = \sum_{(x,y)\in S} \Big(f(x; \cdot) - y\Big)^2. \quad (7)$$

Denote the regularization loss is $\mathcal{R}(\mathbf{U}^1, \ldots, \mathbf{U}^N, \theta_1, \theta_2, \theta_a)$. The overall objective function of CoSTCo is

$$\text{Loss} = \sum_{(x,y)\in S} \Big(f(x; \cdot) - y\Big)^2 + \mathcal{R}(\mathbf{U}^1, \ldots, \mathbf{U}^N, \theta_1, \theta_2, \theta_a) \quad (8)$$

Inspired by a series of work [7, 13] which showed shallow neural networks have nice theoretical guarantees, we prove that a shallow version of CoSTCo (Definition is in Appendix B) is equivalent to an specific neural architecture which has nice guarantees of both optimization and generalization. We state the main result in Theorem 4.1. The idea is that if CoSTCo only uses one convolutional layer, we can squeeze its embedding module and the nonlinear mapping module into one neural network layer. Hence, with sum/mean function as the aggregation function, the shallow version of CoSTCo falls into a category of one-layer neural networks which has been proven to have the nice properties. Proof is in Appendix B.

Theorem 4.1. *Consider a tensor completion task over a order-N target tensor $\mathcal{T} \in \mathbb{R}^{d_1\times\cdots\times d_N}$. A sparse training set S is given and $|S| = O(\sum_{i=1}^N d_i)$. Each data point $(\mathbf{x}, y)$ in training set S is sampled i.i.d. from some distribution $\mathcal{P}$ where $\mathbf{x}$ is represented by an indexing vector $\mathbf{x} = \mathbf{e}_{d_1,\ldots,d_N}(i_1, \ldots, i_N)$ and y is the observed tensor entry $\mathcal{T}_{i_1,\ldots,i_N}$. Under mild conditions, the shallow version of CoSTCo falls into a nice set of shallow nets whose the objective function, if trained with weight decay, satisfies*

- *(Optimization) All local minima are global minima and all saddle points are strict saddle points.*
- *(Generalization) With high probability, the generalization error is bounded by* $O\left(\sqrt{\log \sum_{i=1}^{N} d_i / |S|}\right)$.

4.3 Parameter Efficiency of CoSTCo

The core idea behind CoSTCo is that it uses a parameter-efficient mapping module to model the nonlinear interaction function among factor matrices. Comparing to fully-connected (FC) layers, the convolutional layers, due to its parameter sharing scheme and CoSTCo's unique filter shapes, intuitively forces the embedding module efficiently learns generalizable and informative factor matrices.

The compact representation of the nonlinear function is important, especially for neural nets which have been shown to be capable of memorizing even random labels with enough parameters [46]. Failing to learn the compact representation could lead to a undesirable situation where the nonlinear mapping module can have enough parameters to memorize the training data even when the embedding module has not found good factors. Section 5.1 shows with a fixed randomly initialized embedding module, a model using FC layers in its mapping module can memorize and overfit the training data while CoSTCo doesn't suffer from this issue.

One reason is that a FC layer can have much more parameters than a convolutional layer. Assume the target tensor is a order-N tensor of shape $(d_1, \ldots, d_N)$. The first convolutional layer of CoSTCo outputs have $C \times r$ units[5] where C is the number of channels. The convolutional layer has $(1 \times C) \times (1 \times N) = CN$ parameters. However, the FC layer with the same output size will have $(N \times r) \times (N \times C) = rCN^2$ parameters, which is rN times larger.

Besides, CoSTCo can't work well without learning a good embedding module, because the number of parameters in other modules is limited. For the architecture used in Section 5.2, the total amount of parameters will be

$$\underbrace{r \times N \times \bar{d}}_{\text{embedding}} + \underbrace{\overbrace{1 \times C \times (r \times 1)}^{\text{1st conv layer}} + \overbrace{C \times C \times (1 \times N)}^{\text{2nd conv layer}}}_{\text{nonlinear mapping}} + \underbrace{C \times C}_{\text{aggregation}}$$

where $\bar{d}$ is the average length per mode $(\sum_{i=1}^{N} d_t)/N$. and if we set $C = r$, the number of parameters in the mapping module and the aggregation module is approximately $r^2 N$. Comparably, the embedding module will have $rN\bar{d}$ parameters. In low-rank settings, we usually have $\bar{d} \gg r$. The number of parameters in the embedding module is much larger than those in the rest modules.

CoSTCo learns an informative factors because of the specific shape of the convolutional filters used. Recall that its first convolutional layer uses a filter of shape $(N, 1)$, so its k-th output will be computed from $\mathbf{U}^1{}_{ki_1}, \mathbf{U}^2{}_{ki_2}, \ldots, \mathbf{U}^N{}_{ki_N}$ with some function g. Intuitively, functions such as g will force the factors to be "aligned" in each dimension. Aligning factors to dimensions is how multilinear methods themselves learn informative factors, which are often extracted in factorization-based latent semantic indexing [6]. We provide evidence that our CoSTCo model indeed finds informative factors in Section 5.4.

[5] We ignore the biases terms for both convolutional layers and fully-connected layers.

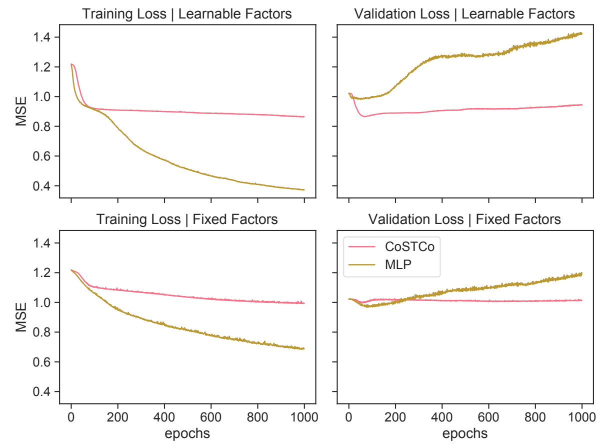


Figure 3: Training/Validation MSE v.s. epochs curves collected in a synthetic tensor. MLP can overfit the data even with fixed random factor matrices. CoSTCo is less likely to overfit and CoSTCo gets worse results when the factor matrices are randomly initialized and not trainable.

5 EXPERIMENTS

We conduct extensive experiments to answer the following questions regarding CoSTCo:

- Can the parameter sharing scheme of the convolutional layers in CoSTCo reduce overfitting on training data?
- Can CoSTCo outperform other state-of-the-art linear/nonlinear completion models for the sparse tensor completion task?
- Does CoSTCo work for downstream tasks of tensor completion, such as recommendation?
- Does CoSTCo learn meaningful factor matrices?

We answer these questions in the following sections. Detailed information about datasets and implementation are provided in Appendix A.1. Our code is released at https://github.com/USC-Melady/KDD19-CoSTCo.

5.1 Overfitting Experiment

We conduct experiments on a synthetic tensor to show the benefits of convolutional layers. We compare CoSTCo to a similar model (MLP) based on fully-connected layers. Both models have the same numbers of hidden neurons. We run experiments on two settings: whether the factor matrices of the model are learn-able or not (fixed random vectors).

Figure 3 shows the results. Similar to what we have discussed, the MLP model can easily overfit the training data even with randomly initialized factor matrices, as it already has enough parameters in the nonlinear mapping module to memorize the data. In comparison, CoSTCo is more robust to overfitting with a much smaller gap between the training and the validation loss than MLP.

5.2 Sparse Tensor Completion

Datasets. We evaluated our proposed models on the sparse tensor completion tasks using six tensors from real-world data whose

Table 2: Statistics of data in sparse tensor completion task.

Dataset	Shape	#Observed entries
MovingMNIST	(20, 100, 64, 64)	819200
ExtremeClimate	(16, 360, 768, 1152)	41180
Traffic	(2713, 31, 24, 12, 3)	218491
Traffic Sparse	(2713, 31, 24, 12, 3)	50848
SG	(2321, 5596, 1600)	105764
Gowalla	(18737, 32510, 1000)	821931

statistics are shown in Table 2. Details of these tensors are in Appendix A.2. In short, we have MovingMNIST [39] (MM), a order-4 tensor of 20 grayscale videos. ExtremeClimate [32] (EC), a order-4 tensor recording 16 climate variables all around the world for 90 days. Traffic [25] (TR), a order-5 tensor recording outputs of highway sensors. Traffic Sparse (TRs): sparser version of the Traffic tensor. SG [23], Gowalla [29]: two order-3 tensors constructed from Point-of-interest (POI) check-in records.

Baseline methods. We compare CoSTCo to other state-of-the-art sparse tensor completion models. Both linear models and nonlinear models are included for complete comparison. All the baseline methods chosen are designed for sparse tensors. To the best of our knowledge, CoSTCo is the first nonlinear approach that utilizes the convolutional neural network to model nonlinear interactions while preserving the low-rank structure of data. The baseline methods are summarized as follows. (1) CP-WOPT [1]: a weighted optimization version of CP factorization for sparse tensor completion. (2) CP-SPALS [4]: an improved fast CP factorization model with implicit leverage scores based ALS solver. (3) P-Tucker [31]: an improved scalable Tucker model with fully paralleled row-wise updating rule. (4) NeuralCP [27]: a Bayesian tensor factorization framework that uses MLP (Multi-Layer Perceptron) to model the nonlinear interactions among tensor factors. (5) DFNT [48]: a scalable Bayesian nonlinear Tucker model based on tensor-variant Gaussian Process with infinite high-order kernels. (6) NTN [37]: a neural network with tensor multiplication layer for modeling 3d tensor.

Evaluation. We evaluate the performance of these methods based on three popular regression metrics. Suppose there are n entries in the test set. Suppose the ground truth of values are $\mathbf{y} = (y_1, \ldots, y_n)$ and the model's predictions are $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_n)$. We use MAE (Mean absolute error), RMSE (Rooted mean squared error), MAPE (Mean absolute percentage error) to measure the models' performances. More details can be found in Appendix A.3.

Hyper-parameter settings. We train our proposed model with Adam [20] method for 50 epochs. We use learning rate of 1e-4 and batch size of 32. We use 10% of the training data as the validation data and do early-stopping over validation MAE with 10 patient epochs. More information is in Appendix A.1.

Figure 4 shows the models' performance of the three tensors sampled from MovingMNIST dataset with different observation ratios (0.03, 0.1, 0.3), i.e., percentage of entries sampled from the original tensor. We show the results of models tested on different sparsity ratios with different factorization ranks. Despite that CP-SPALS tends to have smaller MAE values, we found that it gets the results by predicting too many zeros. CoSTCo almost always

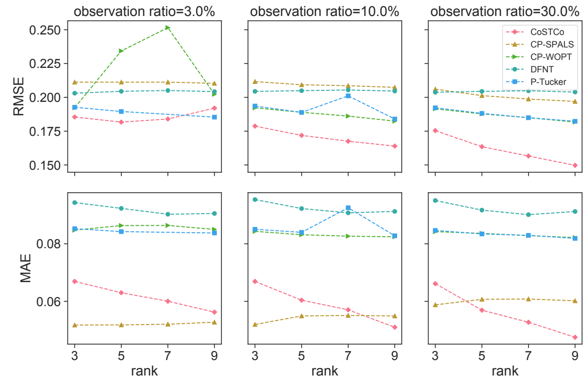


Figure 4: Tensor completion results of MovingMNIST. We collect results on three tensor subsampled from the original MovingMNIST data tensor with different observation ratios (0.03, 0.1, 0.3). We use different factorization ranks to check how models' predictions change. CP-SPALS gets very small MAEs but very high RMSEs because the data has many zero entries and CP-SPALS predicts lots of zeros.

outperforms baseline methods, and the performance increases when the factorization rank increases. For the tensor of observation ratio 0.03, we observe one anomaly that the models' RMSE performance decreases with higher ranks. A potential reason is that we used the same hyper-parameters across all experiments thus the model may not be fully tuned to that dataset. We also observe that with larger observation ratio, the performance of CoSTCo increases faster than other baseline methods, showing the efficiency of CoSTCo.

Table 3 shows the tensor completion results of two POI tensors. We include the results of NTN baseline [6] as these are 3D tensors. We omit some values of CP-SPALS for some experiments, as it has numerical issues (it outputs NaN) when the input tensors are very sparse and the rank is small. One interesting observation is that the performances of all models do not increase with higher ranks. The problem is that the training samples are very limited so that models can easily overfit the training data with higher ranks. CoSTCo shows its robustness: it achieves good performance with a lower rank and remains stable (less overfitting) when the rank increases.

Table 4 shows the tensor completion results on spatiotemporal datasets. We also include the *training* RMSE to highlight the overfitting issue. Both DFNT and P-Tucker get worse test performance when the factorization rank increases: with a higher rank, DFNT has a training problem and P-Tucker suffers from overfitting. The phenomenon is more significant for the Gowalla tensor which is much larger and sparser. Instead, CoSTCo does not suffer from this problem. For the Traffic and ExtremeClimate tensors, the performances of CoSTCo remain stable outperforming all baselines when of the factorization rank increases, implying CoSTCo is able to learn a good representation with small ranks. The results show that CoSTCo is efficient and robust to model a complex tensor with

[6] NTN doesn't treat each dimension equally. We use user and location as the two entities and the POI as the relation and the value of the tensor entry is the strength of the relation between the two entities.

Table 3: Tensor completion results of two POI tensors.

	metric	RMSE				MAE				MAPE (%)			
data	model/rank	20	40	60	80	20	40	60	80	20	40	60	80
SG	NeuralCP	0.1784	0.1748	0.1716	0.1633	0.1004	0.0920	0.0907	0.0840	64.12	54.76	54.26	47.87
	CP-SPALS	-	0.2273	0.2259	0.2257	-	0.1610	0.1595	0.1594	-	100.0	98.93	98.95
	NTN	0.1601	0.1643	0.1832	0.1698	0.0855	0.0918	0.1134	0.1023	50.78	57.25	77.28	67.59
	DFNT	0.2033	0.2270	0.2271	0.2271	0.1242	0.1608	0.1610	0.1610	69.22	99.84	99.99	99.99
	P-Tucker	0.3744	0.3227	0.3337	0.4168	0.2202	0.2022	0.2017	0.2087	162.4	146.2	144.8	148.5
	CoSTCo	**0.1546**	**0.1546**	**0.1546**	**0.1544**	**0.0798**	**0.0786**	**0.0776**	**0.0776**	**44.53**	**42.92**	**41.53**	**41.54**
Gowalla	NeuralCP	0.1402	0.1437	0.1528	0.1466	0.0743	0.0734	0.0811	0.0769	48.54	46.50	54.75	50.35
	CP-SPALS	-	-	-	0.1966	-	-	-	0.1456	-	-	-	99.87
	NTN	0.1381	0.1690	0.1667	0.1580	0.0788	0.1107	0.1125	0.1045	53.60	84.22	87.18	79.05
	DFNT	0.1486	0.1965	0.1968	0.1968	0.0775	0.1455	0.1458	0.1458	44.19	99.74	99.99	100.0
	P-Tucker	0.2972	0.2445	0.2323	0.2333	0.1703	0.1496	0.1391	0.1338	130.4	111.4	102.0	97.17
	CoSTCo	**0.1281**	**0.1278**	**0.1279**	**0.1282**	**0.0623**	**0.0624**	**0.0620**	**0.0623**	**35.95**	**36.02**	**35.59**	**35.86**

Table 4: Tensor completion results of three spatio-temporal tensors.

	metric	RMSE (Train)			RMSE			MAE			MAPE (%)		
data	model/rank	5	10	20	5	10	20	5	10	20	5	10	20
TR	NeuralCP	0.0638	0.0523	0.0415	0.0694	0.0622	0.0614	0.0406	0.0352	0.0332	18.66	15.75	14.84
	CP-SPALS	0.4375	0.4359	0.4352	0.4367	0.4357	0.4357	0.3131	0.3124	0.3124	77.71	77.49	77.49
	DFNT	0.1593	0.1774	0.3617	0.1591	0.1776	0.3611	0.1219	0.1305	0.2558	58.43	51.06	61.52
	P-Tucker	0.0609	0.0480	0.0374	0.0691	0.0641	0.0820	0.0429	0.0375	0.0467	20.77	16.94	19.49
	CoSTCo	0.0573	0.0482	0.0446	**0.0640**	**0.0586**	**0.0568**	**0.0369**	**0.0322**	**0.0303**	**16.94**	**14.05**	**13.16**
TR-s	NeuralCP	0.0617	0.0540	0.0509	**0.0791**	0.0813	0.0815	0.0478	0.0451	0.0469	21.46	**19.57**	20.73
	CP-SPALS	0.4382	0.4378	0.4362	0.4387	0.4387	0.4387	0.3151	0.3151	0.3151	77.95	77.95	77.95
	DFNT	0.1844	0.2061	0.4298	0.1849	0.2061	0.4304	0.1427	0.1549	0.3087	65.08	56.69	75.86
	P-Tucker	0.0495	0.0469	0.0155	0.1230	0.6164	0.9875	0.0662	0.2593	0.4016	27.27	89.42	212.82
	CoSTCo	0.0606	0.0576	0.0594	0.0805	**0.0796**	**0.0776**	**0.0459**	**0.0442**	**0.0435**	**20.54**	19.60	**19.27**
EC	NeuralCP	0.0463	0.0452	0.0357	0.0632	**0.0621**	0.0659	0.0354	0.0349	0.0363	7.65	7.59	8.07
	CP-SPALS	-	-	0.8276	-	-	0.8309	-	-	0.8030	-	-	98.44
	DFNT	0.2246	0.3274	0.7952	0.2295	0.3328	0.7987	0.1854	0.2802	0.7706	30.30	44.53	93.88
	P-Tucker	0.0311	0.0200	0.0104	0.0786	0.0996	0.1247	0.0449	0.0584	0.0832	10.01	14.24	18.60
	CoSTCo	0.0522	0.0529	0.0531	**0.0631**	0.0622	**0.0609**	**0.0347**	**0.0341**	**0.0333**	**7.63**	**7.51**	**7.28**

a small factorization rank. NeuralCP shows competitive results on these three datasets. Although NeuralCP has much more parameters than CoSTCo, CoSTCo obtains better performance, since the mapping function used in CoSTCo is more compact while the one used in NeuralCP has many redundant connections.

5.3 Downstream Tasks

We consider two downstream tasks related with tensor completion: POI recommendation and spatiotemporal forecasting.

POI recommendation. We evaluate CoSTCo on the POI recommendation task. POI recommendation requires a model to predict the user's preferences over POIs based on latent features and geographical features. We compare CoSTCo with the state-of-the-art POI recommendation algorithm, RankGeoFM [23], over the SG and Gowalla datasets. RankGeoFM is an extension of GeoMF [26], a matrix factorization model jointly modeling the latent features and geographical influences, by training the model with pairwise ranking based objective functions to resolve the data sparsity problem.

We use the same pairwise ranking based objective function to train CoSTCo. Training details can be found in Appendix A.3. For RankGeoFM, we use the implementation provided by the authors[7]. For both models, we use the same hyper-parameters as described in [23]: factorization rank of 100, pairwise ranking margin of 0.3. The result is shown in Figure 5. Our model achieves much better results than RankGeoFM, showing that CoSTCo is capable of automatically discovering the geographical influence from the checkin tensor without relying too much on manually designed features.

Spatiotemporal forecasting. Spatiotemporal forecasting is an important task which has many real-world applications. It requires

[7] http://spatialkeyword.sce.ntu.edu.sg/eval-vldb17/

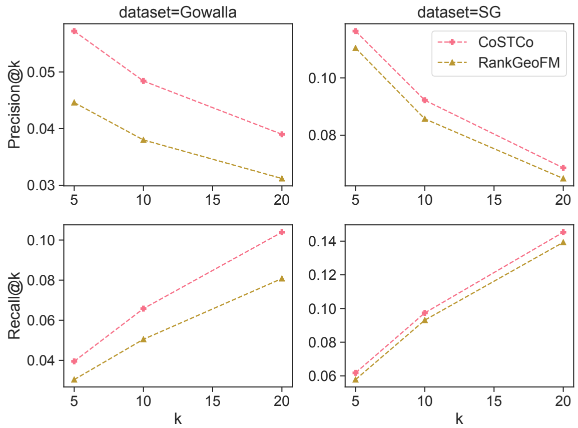


Figure 5: POI recommendation results of two POI checkin datasets. Best viewed in color.

Table 5: Forecasting RMSE on the climate dataset.

Method	Greedy	MLGP	MOGP	MLMTL	CoSTCo
RMSE	0.9069	0.8973	0.9273	0.9528	**0.8693**

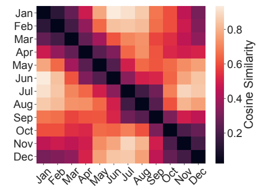
a model to predict future values based on past observations. Spatiotemporal forecasting is often formalized as a tensor learning [3, 45] or multi-task learning [34]. We use the climate data used in [3], which records the values of 17 different climate variables in 125 locations monthly for 13 years with missing data interpolated. The data is a tensor of size $17 \times 125 \times 156$. For each variable, we normalize it to have zero mean and unit variance. We split the dataset 80%/20% by time to the training/testing dataset.

Though CoSTCo is designed for tensor completion, it can be extended to predict entries of future time steps for the climate forecasting task. The first trick is to do a time-index split. A natural observation of climate data is that it contains annual periodicity. To model this periodicity, we reshape the original mode-3 tensor to a mode-4 tensor, by splitting its time index into both a yearly index and a monthly index. By doing this, the information inside each mode is more distinguishable, so factor interactions will be simpler and more easily captured. Another one is applying a regularization over the factor matrices in the time dimension such that the factor vectors of two nearby months or years are close.

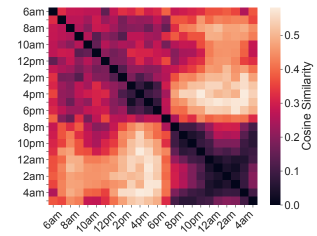
We compare our methods to a set of baselines on this dataset: (1) Greedy tensor learning (Greedy) [3]: a fast tensor learning algorithm for multivariate spatiotemporal forecasting. (2) MLGP [44]: tensor regression with Gaussian process. (3) MOGP [2]: multi-output Gaussian process. (4) MLMTL [34]: multi-task learning with trace norm optimization. Table 5 shows the RMSE of different approaches. Comparing to other multilinear methods, CoSTCo clearly achieves improved performance.

5.4 Visualization of Learned Factors

To further understand the model learned by CoSTCo, we visualize its learned factors. Figure 6a shows pairwise cosine similarities among



(a) Similarity heatmap among "months" factors learned from the climate forecasting tensor.



(b) Similarity heatmap among "hours" factors learned from the traffic tensor.

Figure 6: Pairwise cosine similarities among factors learned in some temporal dimensions.

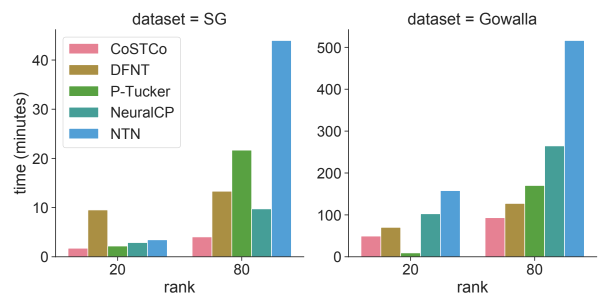


Figure 7: Running time of different completion methods.

temporal (monthly) factors learned from the climate forecasting tensor. Intuitively, the four dark blocks in the heat map suggest that the learned monthly temporal factors capture seasonal changes inside the climate data. Figure 6b also shows such similar behavior among factors of non-rush hours (9pm to 5am) and factors of working hours (8am to 7pm). Intuitively, the traffic factors learned by CoSTCo could be capturing the traffic patterns associated with rush hours and non-rush hours.

5.5 Running Time Comparison

The training time of CoSTCo is linear w.r.t. the number of observations and is irrelevant to the size of the target tensor and doesn't involve complex tensor operations. Figure 7 shows the running time of each method in the tensor completion task over the SG and Gowalla tensors. Gowalla is a larger and sparser tensor than SG. We find that CoSTCo runs fastest in most cases. Besides, its running time does not increase drastically as P-Tucker/NTN does, which both require tensor operations. All the experiments are run in the same machine with 32-core Intel(R) Xeon(R) CPU E5-2640 and 256G memory. The architectures of neural network models are shown in Appendix A.1 and we use early-stopping during training.

6 CONCLUSION

We proposed CoSTCo, a carefully designed convolution based model for nonlinear sparse tensor completion. CoSTCo efficiently captures the nonlinear interactions and has an inductive bias to learn generalizable and informative factors. Extensive experiments on real-world tensor completion datasets showed CoSTCo achieved better performance than state-of-the-art linear/nonlinear completion methods. Besides, CoSTCo outperformed corresponding state-of-the-art models on downstream tasks. For future work, we will investigate the following directions: 1) provide interpretations of the inferred latent factors, 2) investigate theoretical properties for CoSTCo with deeper neural networks and general activation functions, 3) generalize CoSTCo for non-linear Tucker decomposition while preserving the low-rank property, 4) extend CoSTCo with more downstream tasks, such as image inpainting or streaming tensor completion.

7 ACKNOWLEDGMENTS

We are very thankful to Umang Gupta, Dehua Cheng and anonymous reviewers for their thoughtful advice. This research has been funded in part by NSF grants IIS-1254206, IIS-1539608, USC Annenberg Graduate Fellowship to Yaguang Li and Michael Tsang. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any of the sponsors such as NSF.

REFERENCES

[1] Evrim Acar, Daniel M Dunlavy, Tamara G Kolda, and Morten Mørup. 2011. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems* 106, 1 (2011), 41–56.

[2] Mauricio A Álvarez and Neil D Lawrence. 2011. Computationally efficient convolved multiple output gaussian processes. *JMLR* 12, May (2011), 1459–1500.

[3] Mohammad Taha Bahadori, Qi Rose Yu, and Yan Liu. 2014. Fast multivariate spatio-temporal analysis via low rank tensor learning. In *NIPS*. 3491–3499.

[4] Dehua Cheng, Richard Peng, Yan Liu, and Ioakeim Perros. 2016. SPALS: Fast alternating least squares via implicit leverage scores sampling. In *NIPS*. 721–729.

[5] Wei Chu and Zoubin Ghahramani. 2009. Probabilistic models for incomplete multi-dimensional arrays. In *AISTATS*. 89–96.

[6] Chris Ding, Tao Li, and Wei Peng. 2006. Nonnegative matrix factorization and probabilistic latent semantic indexing: Equivalence chi-square statistic, and a hybrid method. In *AAAI*, Vol. 42. 137–143.

[7] Simon S Du and Jason D Lee. 2018. On the Power of Over-parametrization in Neural Networks with Quadratic Activation. In *International Conference on Machine Learning*. 1328–1337.

[8] Gintare Karolina Dziugaite and Daniel M Roy. 2015. Neural network matrix factorization. *arXiv preprint arXiv:1511.06443* (2015).

[9] Martin Ester et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *KDD*, Vol. 96. 226–231.

[10] Martín Abadi et al. 2016. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. In *OSDI*. Software available from tensorflow.org.

[11] Xiaomin Fang, Rong Pan, Guoxiang Cao, Xiuqiang He, and Wenyuan Dai. 2015. Personalized Tag Recommendation through Nonlinear Tensor Factorization Using Gaussian Kernel.. In *AAAI*. 439–445.

[12] Silvia Gandy, Benjamin Recht, and Isao Yamada. 2011. Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems* 27, 2 (2011), 025010.

[13] Benjamin Haeffele, Eric Young, and Rene Vidal. 2014. Structured Low-Rank Matrix Factorization: Optimality, Algorithm, and Applications to Image Processing. In *ICML*. 2007–2015.

[14] Richard A Harshman. 1970. Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multimodal factor analysis. (1970).

[15] Lifang He, Xiangnan Kong, Philip S Yu, Xiaowei Yang, Ann B Ragin, and Zhifeng Hao. 2014. Dusk: A dual structure-preserving kernel for supervised tensor learning with applications to neuroimages. In *SDM*. SIAM, 127–135.

[16] Lifang He, Chun-Ta Lu, Guixiang Ma, Shen Wang, Linlin Shen, S Yu Philip, and Ann B Ragin. 2017. Kernelized support tensor machines. In *ICML*. 1442–1451.

[17] Joyce C Ho, Joydeep Ghosh, and Jimeng Sun. 2014. Marble: high-throughput phenotyping from electronic health records via sparse nonnegative tensor factorization. In *KDD*. ACM, 115–124.

[18] U Kang, Evangelos Papalexakis, Abhay Harpale, and Christos Faloutsos. 2012. Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries. In *KDD*. ACM, 316–324.

[19] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. 2016. Convolutional matrix factorization for document context-aware recommendation. In *RecSys*. ACM, 233–240.

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

[21] Tamara G Kolda and Brett W Bader. 2009. Tensor decompositions and applications. *SIAM review* 51, 3 (2009), 455–500.

[22] Neil D Lawrence and Raquel Urtasun. 2009. Non-linear matrix factorization with Gaussian processes. In *ICML*. ACM, 601–608.

[23] Xutao Li, Gao Cong, Xiao-Li Li, Tuan-Anh Nguyen Pham, and Shonali Krishnaswamy. 2015. Rank-geofm: A ranking based geographical factorization method for point of interest recommendation. In *SIGIR*. ACM, 433–442.

[24] Xutao Li, Yunming Ye, and Xiaofei Xu. 2017. Low-rank tensor completion with total variation for visual data inpainting. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[25] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *ICLR*.

[26] Defu Lian, Cong Zhao, Xing Xie, Guangzhong Sun, Enhong Chen, and Yong Rui. 2014. GeoMF: joint geographical modeling and matrix factorization for point-of-interest recommendation. In *KDD*. ACM, 831–840.

[27] Bin Liu, Lirong He, Yingming Li, Shandian Zhe, and Zenglin Xu. 2018. NeuralCP: Bayesian Multiway Data Analysis with Neural Tensor Decomposition. *Cognitive Computation* 10, 6 (2018), 1051–1061.

[28] Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye. 2013. Tensor completion for estimating missing values in visual data. *PAMI* 35, 1 (2013), 208–220.

[29] Yiding Liu, Tuan-Anh Nguyen Pham, Gao Cong, and Quan Yuan. 2017. An experimental evaluation of point-of-interest recommendation in location-based social networks. *VLDB* 10, 10 (2017), 1010–1021.

[30] Atsuhiro Narita, Kohei Hayashi, Ryota Tomioka, and Hisashi Kashima. 2012. Tensor factorization using auxiliary information. *Data Mining and Knowledge Discovery* 25, 2 (2012), 298–324.

[31] Sejoon Oh, Namyong Park, Sael Lee, and U Kang. 2018. Scalable tucker factorization for sparse tensors-algorithms and discoveries. In *ICDE*. IEEE, 1120–1131.

[32] Evan Racah, Christopher Beckham, Tegan Maharaj, Samira Ebrahimi Kahou, Mr Prabhat, and Chris Pal. 2017. ExtremeWeather: A large-scale climate dataset for semi-supervised detection, localization, and understanding of extreme weather events. In *NIPS*. 3402–3413.

[33] Steffen Rendle and Lars Schmidt-Thieme. 2010. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM*. ACM, 81–90.

[34] Bernardino Romera-Paredes, Hane Aung, Nadia Bianchi-Berthouze, and Massimiliano Pontil. 2013. Multilinear multitask learning. In *ICML*. 1444–1452.

[35] Bernardino Romera-Paredes and Massimiliano Pontil. 2013. A new convex relaxation for tensor completion. In *NIPS*. 2967–2975.

[36] Amnon Shashua and Tamir Hazan. 2005. Non-negative tensor factorization with applications to statistics and computer vision. In *ICML*. ACM, 792–799.

[37] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*. 926–934.

[38] Qingquan Song, Xiao Huang, Hancheng Ge, James Caverlee, and Xia Hu. 2017. Multi-aspect streaming tensor completion. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 435–443.

[39] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. 2015. Unsupervised learning of video representations using lstms. In *ICML*. 843–852.

[40] Ledyard R Tucker. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika* 31, 3 (1966), 279–311.

[41] Xian Wu, Baoxu Shi, Yuxiao Dong, Chao Huang, and Nitesh Chawla. 2018. Neural Tensor Factorization. *arXiv preprint arXiv:1802.04416* (2018).

[42] Zenglin Xu, Feng Yan, et al. 2011. Infinite Tucker decomposition: Nonparametric Bayesian models for multiway data analysis. *arXiv:1108.6296* (2011).

[43] Rose Yu, Dehua Cheng, and Yan Liu. 2015. Accelerated online low rank tensor learning for multivariate spatiotemporal streams. In *ICML*. 238–247.

[44] Rose Yu, Guangyu Li, and Yan Liu. 2018. Tensor Regression Meets Gaussian Processes. In *AISTATS*. 482–490.

[45] Rose Yu and Yan Liu. 2016. Learning from multiway data: Simple and efficient tensor regression. In *ICML*. 373–381.

[46] Chiyuan Zhang et al. 2016. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530* (2016).

[47] Shandian Zhe, Zenglin Xu, Xinqi Chu, Yuan Qi, and Youngja Park. 2015. Scalable nonparametric multiway data analysis. In *AISTATS*. 1125–1134.

[48] Shandian Zhe, Kai Zhang, Pengyuan Wang, Kuang-chih Lee, Zenglin Xu, Yuan Qi, and Zoubin Ghahramani. 2016. Distributed flexible nonlinear tensor factorization. In *NIPS*. 928–936.

APPENDIX

A EXPERIMENTAL DETAIL

A.1 Implementation and Hyper-parameters

We implement the experiments with TensorFlow [10]. For all the experiments, we use the Adam [20] optimizer with the default hyper-parameters to train neural networks. The hyper-parameters are shown in the Table A1. We use pairwise ranking loss A.1 to train the model for the POI recommendation task. For the climate forecasting task, we use weight decay of 0.0001.

Sparse tensor completion experiments. The aggregation modules used are MLPs, which have *nc* units in the hidden layer and 1 unit in the output layer. *nc* is the number of convolutional channels. For SG/Gowalla, *nc* equals to the model's rank. For TR/TR-s/EC, *nc* equals to the double of the model's rank. For NeuralCP, we replace the convolutional layers with FC layers such that the output sizes are the same. We run with different random seeds and report the average scores. We initially tune the hyper-parameters of NeuralCP, NTN and `CoSTCo`, and keep the hyper-parameters consistent among methods for all experiments. We found that when we tune these methods independently, `CoSTCo` still consistently outperforms the baselines. For the rest of the baselines, we use default hyper-parameters if any.

Ranking loss in POI recommendation task. We train `CoSTCo` using the same pairwise ranking loss as RankGeoMF [23] does. More specifically, each train data contains two tensor entries $(\mathrm{u}, \mathrm{POI}_1, \mathrm{LOC}_1)$, $(\mathrm{u}, \mathrm{POI}_2, \mathrm{LOC}_2)$, and the values y_1, y_2. Assume the model's prediction is y_1', y_2'. The loss is $\mathrm{loss} = I(y_1' \leq y_2' + \epsilon) I(y_1 \geq y_2)$. I is the indicator function such that $I(\mathrm{true} = 1), I(\mathrm{false} = 0)$. The parameter ϵ is served as a margin and potentially can help model's generalization. We use $\epsilon = 0.3$ as indicated in [23].

Table A1: Hyper-parameter settings.

Dataset	lr	epochs	batch size
MovingMNIST	0.001	200	64
Gowalla/SG/TR/TRs/Ec	0.0001	50	32
POI reco	0.1	500	256
Climate Forecasting	0.03	1000	1024

A.2 Datasets

MovingMNIST [39] (MM): a dataset of 20 videos about moving numbers. Each video is a 3d tensor and each frame is a grey-scale 2d image. We sample 100 frames from all 10000 frames for each video. Each tensor index is (video_id, time_id, row_id, column_id).

ExtremeClimate [32] (EC): a climate dataset of 4d tensors, recording 16 different climate variables all over the world. We use the data from 1989's first three months and sample 0.05% observations to construct the tensor. The values of variables' measurements differ a lot, so we normalize the data per variable by first subtracting the minimal value of that variable, adding one to all values, then taking the log function, and finally dividing it by the maximum value of that variable. Each tensor index is (time_id, latitude_id, longitude_id, variable_id).

Traffic [25] (TR): a dataset of sensor data collected from intersections of LA highways. The data is collected from all sensors every 5 minutes. Each sensor reports three varibales: average traffic speed, traffic volume, and occupancy. We use the data from March 2016 and sample 0.3% observations to construct the tensor. Each tensor index is a tuple of (sensor_id, day, hour, minute, variable_id). We sample the Traffic Sparse (TRs) by using only 0.07% observations.

SG [23] checkins is collected by Foursquare in Singapore and Gowalla [29] checkins is collected by Gowalla all over the world. For SG, we split the geographical region into 1600 grids and convert the raw locations (latitude, longitude) to grid indices. For Gowalla, since the location range is much larger, we use DBSCAN [9] to cluster the locations and use the cluster id of each location as its location id. Due to the large value range, we normalize the tensor values by first clipping tensor entries with a maximum of 10 and then dividing them by 10. Each tensor index is a tuple of (user_id, location_id, poi_id).

A.3 Metrics

We use the standard MAE (Mean absolute error) and RMSE (Rooted mean squared error) metrics. For MAPE (Mean absolute percentage error), we add a threshold δ to avoid too small values on the denominator $\mathrm{MAE}(\mathbf{y}, \hat{\mathbf{y}}) = 100\% \times (1/n) \times \sum_{i=1}^{n} |\mathbf{y}_i - \hat{\mathbf{y}}_i| / \max(|\mathbf{y}_i|, \delta)$. We use $\delta = 0.1$ through the whole paper.

For the recommendation task, given a user u, we recommend k POIs (remove those only encountered in the training set) named reco_u. Denote the set real_u to be the set of all POIs visited by the user u, then $\mathrm{Precision@}k = |\mathrm{real}_u \cap \mathrm{reco}_u| / K$, $\mathrm{Recall@}k = |\mathrm{real}_u \cap \mathrm{reco}_u| / |\mathrm{reco}_u|$. Precision@$k$ and Recall@k are averaged among all users. A better model will have higher scores.

B PROOF OF THEOREM 4.1

Our proof of Theorem 4.1 is based on recent advances of theories of shallow neural networks [7, 13]. If we only use one convolutional layer, we can get informative theoretical analysis about `CoSTCo`. We will first show that the shallow version of `CoSTCo` falls into a nice category of one-layer neural network then provide a useful theorem characterizing the properties of that neural network category.

Definition of the shallow `CoSTCo`. Assume `CoSTCo` has one convolutional layer which has C output channels and filters of size (w, h). Assume the activation function $\sigma(x) = x^2$ and the aggregation module outputs the sum of its inputs.

One observation is that a mapping module with one convolutional layer can be represented as one single fully-connected layer.

Theorem B.1. *Consider a modified* `CoSTCo` *on order-N tensor* $\mathcal{T}$ *of shape* $(d_1, \ldots, d_N)$. *Assume there are* C *convolutional kernels, the size of the kernel is* (w, h), *the stride is 1 and no padding is added. Denote* θ *as the parameter of the convolutional layer, a tensor of size* $1, C, w, h$. *The number of convolutional layer's outputs is* $n_o = C \times (N - w + 1) \times (r - h + 1)$. *Let us further assume we have a fully-connected layer who has* Nr *input nodes,* n_o *hidden nodes,* 1 *output nodes and the same activation function as* `CoSTCo`. *Then there exists a mapping function* $h : \mathbb{R}^{C \times w \times h} \rightarrow \mathbb{R}^{n_o \times N \times r}$, *such that the mapping module is equivalent to a fully-connected layer whose input-to-hidden weight matrix is* $h(\mathcal{W})$ *and hidden-to-output weight matrix is all one matrix of shape* $1 \times n_o$.

Proof. Denote the input tensor of the convolutional layer as $\mathcal{X} \in \mathbb{R}^{1 \times Nr}$. We use the $\mathbf{x} \in \mathbb{R}^{Nr}$ to represent the flattened vector of $\mathcal{X}$. A 1-stride, zero-padding, size w, h convolution kernel will have $C \times (N-w+1) \times (r-h+1)$ outputs. Suppose $\mathcal{W} \in \mathbb{R}^{C \times w \times h}$ is the convolution filter's parameter. The (k, i, j)-th output of the convolutional layer is given by $\sum_{a=1}^{w} \sum_{b=1}^{h} \mathcal{W}_{k,a,b} \mathcal{X}_{1,i+a,j+b}$. For (k, i, j)-th output, We can find a vector $\mathbf{w}^{kij} \in 1 \times N \times r$, such that $\forall m \in \{1, \ldots, N-w\}$, $n \in \{1, \ldots, r-h\}$, $a \in \{1, \ldots, w\}$, $b \in \{1, \ldots, h\}$,

$$\mathbf{w}^{kij}_{1,(m+a)*r+(n+b)} = \begin{cases} \mathcal{W}_{k,a,b}, & if\, m = i \text{ and } n = j \\ 0, & \text{otherwise.} \end{cases}$$

Then the (k, i, j)-th output of the convolutional layer equals to $\mathbf{w}^{kij}\mathbf{x}$. We can then construct a matrix $\mathbf{W}'$

$$\mathbf{W}' = \begin{bmatrix} \mathbf{w}^{k=1, i=1, j=1} \\ \vdots \\ \mathbf{w}^{k=C, i=(N-w+1), j=(r-h+1)} \end{bmatrix}$$

and use it as the fully-connected layer's weight matrix. Define h in the way that $h(\mathcal{W}) = \mathbf{W}'$. By doing so, we get the mapping function such that the mapping module is equivalent to a fully-connected layer. □

Since there is no nonlinear activation function between the embedding module and the nonlinear mapping module, we can merge them and squeeze `CoSTCo` into a shallow neural network.

We state this formally in Theorem B.2.

Lemma B.2. *Assume the convolutional filters of* `CoSTCo` *is fixed and it has C filters, each of size $w \times h$. Denote n_o as the number of output of the convolutional layer, $n_o = C \times (N-w+1) \times (r-h+1)$. Then there exists a mapping function $h : \mathbb{R}^{C \times w \times h} \to \mathbb{R}^{n_o \times Nr}$, such that* `CoSTCo`*'s neural network function f can be written as*

$$f(\mathbf{x}; \{\Theta_W, \Theta_M\}) = g\left(\sum_{t=1}^{n_o} \left(\mathbf{W}_{t,:}\mathbf{x}\right)^2\right),$$

where $\mathbf{W} = h(\Theta_W)\Theta_M$, $\mathbf{W} \in \mathbb{R}^{n_o \times (d_1+d_2+d_3)}$. The right-hand side is equivalent to a one-hidden-layer neural network with quadratic activation in hidden layer and g in output layer.

Theorem B.3. *Consider the modified* `CoSTCo` *and the mapping function h as defined in B.2. Let the aggregation module output the sum of all its inputs. Denote $f(x; \mathbf{U}^1, \ldots, \mathbf{U}^N, \mathcal{W})$ be the function of the modified* `CoSTCo` *where $\mathcal{W}$ is the parameter of the first convolutional layer. The modified* `CoSTCo` *is equivalent to a one-hidden-layer neural network with hidden parameter $\hat{\mathbf{W}} \in \mathbb{R}^{n_o \times Nr}$ such that*

$$f(\mathbf{x}; \mathbf{U}^1, \ldots, \mathbf{U}^N, \mathcal{W}) = \sum_{i=1}^{n_o} \left(\sum_{j=1}^{Nr} \hat{\mathbf{W}}_{ij}\mathbf{x}_j\right)^2$$

Proof. Given an input vector $\mathbf{x}$, the output of embedding layer is $\mathbf{Mx}$ as defined in Equation 4. Theorem B.1 states that the convolutional layer equals to a fully-connected layer of parameter $h(\mathcal{W})$. As its activation function is quadratic function and its aggregation module outputs the sum of all its inputs, the output of the modified `CoSTCo` given an input $\mathbf{x}$, will

$$f(\mathbf{x}; \mathbf{U}^1, \ldots, \mathbf{U}^N, \mathcal{W}) = \mathrm{Agg}\Big(\sigma\Big(h(\mathcal{W})\mathbf{Mx}\Big)\Big) = \sum_{i=1}^{n_o} \left(\sum_{j=1}^{d} \hat{\mathbf{W}}_{ij}\mathbf{x}_j\right)^2$$

where $\hat{\mathbf{W}} = h(\mathcal{W})\mathbf{M}$. Thus, the modified `CoSTCo` is equivalent to a one-hidden-layer neural network. □

We cite a useful theorem in [7] that characterizes the nice properties of a specific kind of one-layer neural networks with quadratic activations.

Lemma B.4 (Theorem 3.2 and Theorem 4.3 in [7]). *Suppose the training set S contains the input data $\mathbf{x} \in \mathbb{R}^d$ and target $y \in \mathbb{R}$. Consider a one-hidden-layer neural network represented by $f(\mathbf{x}; \mathbf{W}) = \sum_{i=1}^{m} \left(\sum_{j=1}^{d} \mathbf{W}_{ij}\mathbf{x}_j\right)^2$ where $\mathbf{W} \in \mathbb{R}^{m \times d}$ is the weight from input layer to the hidden layer. Assume that $m * (m-1) \geq 2|S|$. Let $\ell(\hat{y}, y) = (\hat{y} - y)^2$ and the objective function is*

$$L(\mathbf{W}) = \frac{1}{2|S|} \sum_{(\mathbf{x},y) \in S} \ell\left(f\left(\mathbf{x}; \mathbf{W}\right), y\right) + \frac{\lambda}{2}\|\mathbf{W}\|_F^2 + \langle \mathbf{C}, \mathbf{W}^T\mathbf{W}\rangle, \quad (9)$$

where $\langle \cdot, \cdot \rangle$ denotes dot-product and $\mathbf{C}$ is some random positive semi-definite matrix with $\|\mathbf{C}\|_F^2 \leq \delta$. The objective function 9 has the following three properties:

1. *All local minima of $L(\mathbf{W})$ are global minima.*
2. *All saddles point of $L(\mathbf{W})$ are strict saddle points.*
3. *For any local optimum $\mathbf{W}'$ of $L(\mathbf{W})$, for all $\mathbf{W} \in \mathbb{R}^{m \times d}$, we have*

$$\frac{1}{2|S|} \sum_{(\mathbf{x},y) \in S} \ell\left(f\left(\mathbf{x}; \mathbf{W}'\right), y\right) + \frac{\lambda}{2}\|\mathbf{W}'\|_F^2 \leq$$
$$\frac{1}{2|S|} \sum_{(\mathbf{x},y) \in S} \ell\left(f\left(\mathbf{x}; \mathbf{W}\right), y\right) + \frac{\lambda}{2}\|\mathbf{W}\|_F^2 + \delta\|\mathbf{W}\|_F^2.$$

In addition, for some constants $c_1, c_2, c_3 \in \mathbb{R}$, assume that $(\mathbf{x}, y) \in S$ is drawn i.i.d. from some probabilistic distribution P such that $\mathbf{x}$ is inside a bounded ball: $\|\mathbf{x}\|_2^2 \leq c_1$, $\left\|\mathbf{x}\mathbf{x}^T\right\|_2^2 \leq \infty$. For all $\mathbf{W} \in \mathbb{R}^{m \times d}$ that satisfies $\|\mathbf{W}\|_F^2 \leq c_2$, the gap between training error (empirical test error) $\frac{1}{|S|}\sum_{(\mathbf{x},y) \in S} \ell(f(\mathbf{x}; \mathbf{W}), y)$ and generalization error (expected test error) $\mathbb{E}_{(\mathbf{x},y) \sim P}[\ell(f(\mathbf{x}; \mathbf{W}), y)]$ is bounded by

$$\mathbb{E}_{(\mathbf{x},y) \sim P}[\ell(f(\mathbf{x}; \mathbf{W}), y)] - \frac{1}{|S|} \sum_{(\mathbf{x},y) \in S} \ell(f(\mathbf{x}; \mathbf{W}), y) \leq c_1^2 c_2^2 c_3 \sqrt{\frac{\log d}{|S|}},$$

Finally, we give the proof of Theorem 4.1.

Proof. Theorem B.3 shows that the modified `CoSTCo` can be written as $f(\mathbf{x}) = \sum_{i=1}^{n_o} \left(\sum_{j=1}^{d} \hat{\mathbf{W}}_{ij}\mathbf{x}_j\right)^2$, which is exactly the same function format as in Lemma B.4. Moreover, $\mathbf{x}$ is an indexing vector and by definition 1 we have $\|\mathbf{x}\|_2^2 = N$, $\left\|\mathbf{x}\mathbf{x}^T\right\|_2^2 = N, \forall \mathbf{x} \in S$. Assume $w = N/2+1$, $h = 1$, then $n_o = CNr/2$. As we assumed that $|S| = O(\sum_{i=1}^{N} d_i)$, if $C = O(\sqrt{\max_i d_i/(Nr^2)})$, then $n_o \times (n_o - 1) \geq O(|S|)$. To sum it up, if we train the modified `CoSTCo` with objective function 9 and $\mathbf{C}$ is some random positive semi-definite matrix with $\|\mathbf{C}\|_F^2 \leq \delta$. Then $L(\hat{\mathbf{W}})$ satisfies all three properties in Lemma B.4. Besides, if for some constants $c_2, c_3 \in \mathbb{R}$ and $\left\|\hat{\mathbf{W}}\right\|_F^2 \leq c_2$, the generalization error is bounded by

$$\mathbb{E}_{(\mathbf{x},y) \sim P}[\ell(f(\mathbf{x}; \mathbf{W}), y)] - \frac{1}{|S|} \sum_{(\mathbf{x},y) \in S} \ell(f(\mathbf{x}; \mathbf{W}), y) \leq c_2^2 c_3 \sqrt{\frac{\log \sum_{i=1}^{N} d_i}{|S|}}.$$

□