Meta-learning from Heterogeneous Tensors for Few-shot Tensor Completion

Tomoharu Iwata **Atsutoshi Kumagai**
NTT Corporation

Abstract

We propose neural network-based models for tensor completion in few observation settings. The proposed model can meta-learn inductive bias from multiple heterogeneous tensors without shared modes. Although many tensor completion methods have been proposed, the existing methods cannot leverage knowledge across heterogeneous tensors, and their performance is low when only a small number of elements are observed. The proposed model encodes each element of a given tensor by considering information about other elements while reflecting the tensor structure via a self-attention mechanism. The missing values are predicted by tensor-specific linear projection from the encoded vectors. The proposed model is shared across different tensors, and it is meta-learned such that the expected tensor completion performance is improved using multiple tensors. By experiments using synthetic and real-world tensors, we demonstrate that the proposed method achieves better performance than the existing meta-learning and tensor completion methods.

1 INTRODUCTION

Data are often represented by tensors, or multidimensional arrays, in a wide variety of applications, which include signal processing (Muti and Bourennane, 2005), sensor array processing (Sidiropoulos et al., 2000), spatio-temporal analysis (Takeuchi et al., 2013), computer vision (Vasilescu and Terzopoulos, 2002), and neuroscience (Mocks, 1988). In these applications, tensor completion is an important machine learning task, which aims to impute missing values of a partially observed tensor (Romera-Paredes and Pontil, 2013). For tensor completion, many methods have been proposed, such as tensor decomposition (Hitchcock, 1927; Tucker, 1966; Oseledets, 2011; Wu et al., 2022; Razin et al., 2021; Tomioka et al., 2011) and neural network-based methods (Liu et al., 2019; Zhao et al., 2020; Socher et al., 2013; Ibrahim et al., 2023). However, these existing methods are trained for each tensor, and they do not leverage knowledge across multiple tensors. Therefore, they often fail when the number of observed elements is small, e.g., the tensor size is small, and the missing value ratio is high.

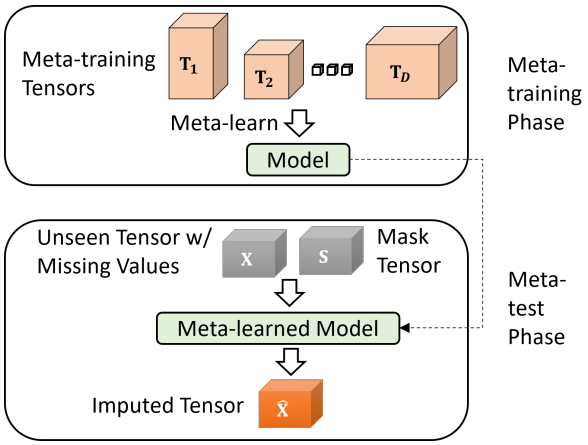


Figure 1: Proposed framework. In the meta-training phase, our model is meta-learned using multiple tensors with different sizes without shared modes. In the meta-test phase, given an unseen tensor with missing values and a mask tensor, we impute the missing values using the meta-learned model.

In this paper, we propose neural network-based models that are meta-learned using multiple tensors for few observation settings. Figure 1 illustrates our proposed framework. In the meta-training phase, the proposed model is meta-learned using a set of heterogeneous tensors, where their modes are not shared across tensors,

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

their sizes are different, and they are obtained from different domains. In the meta-test phase, the meta-learned model is used to impute missing values of unseen tensors that are different from but related to the meta-training tensors.

The existing tensor decomposition methods use inductive bias discovered by domain experts, such as low-rankness (Hitchcock, 1927), sparsity (Xue et al., 2021), train (Oseledets, 2011), ring (Zhao et al., 2016), hierarchical (Grasedyck, 2010), and network (Li and Sun, 2020) structures. These inductive biases are common in tensors from various domains. The proposed method learns inductive biases from related tensors based on a data-driven approach, and stores them in our model. Due to the high expressive power of neural networks, we can obtain more complex inductive biases than the existing tensor completion methods, and use them adaptively for each tensor by inputting it into our model.

For meta-learning from heterogeneous tensors, we develop neural tensor attention networks (NTANs) that take a tensor with missing values as input, and output the imputed missing values. Our model consists of an embedding layer, multi-head m-mode tensor attention layers (mMTAs), fiber-wise feed-forward networks (FFNs), and a prediction layer as shown in Figuer 2. The embedding layer linearly projects observed values with missing value information, and obtains an embedding vector for each element. The mMTA transforms the embeddings by performing self-attention between subtensors sliced along each mode alternately, which enables us to aggregate information of other elements while considering tensor structure. Intuitively, elements that compose similar subtensors are likely to have similar embeddings. The mMTA is an extension of vanilla attention (Vaswani et al., 2017) for higher-order tensor inputs, allowing attention to be performed along arbitrary modes. FFNs nonlinearly transform the embeddings, by which the expressive power is improved. The prediction layer outputs an imputed tensor by linearly projecting the transformed embeddings, where the projection is adapted to the observed values. By the adaptation, we can flexibly output an imputed tensor that matches the input tensor. The adapted projection can be obtained in a closed form by minimizing the squared error, which enables effective meta-learning.

The model parameters are shared across different tensors except for the prediction layer, by which we can learn useful common inductive biases for imputing various tensors. The parameters are optimized by minimizing the expected test squared errors of missing value imputation when a few observed values are given by randomly generating various tensors from the meta-training tensors with an episodic training framework (Ravi and Larochelle, 2017). The meta-learned model can perform tensor completion efficiently since it can output a predicted tensor by a forwarding pass of the neural network without iterative optimization. Since the proposed model does not assume that modes are shared across tensors, we can use the learned inductive bias for newly given tensors obtained in domains different from meta-training tensors.

The main contributions of this paper are as follows: 1) We present a meta-learning framework for tensor completion using various tensors without shared modes. 2) We propose an attention-based neural network model for tensor completion that can handle heterogeneous tensors. 3) Using synthetic and real-world tensor datasets, we demonstrate the effectiveness of the proposed method in tensor completion with a small number of observations.

2 RELATED WORK

Many tensor completion methods have been proposed (Song et al., 2019; Liu et al., 2012; Liu and Moitra, 2020; Nimishakavi et al., 2018; Bugg et al., 2022; Zhang et al., 2020; Lee and Wang, 2021; Romera-Paredes and Pontil, 2013; Lacroix et al., 2018; Yang et al., 2022; Li et al., 2023). However, these methods are trained using only a single tensor, and cannot learn knowledge common among different tensors. Although transfer learning methods for tensor completion (Chen et al., 2021; Mohammadi et al., 2019) can transfer knowledge between tensors, they assume only two tensors (source and target) and require the target tensor for training. In contrast, the proposed method can handle more than two tensors and does not require target tensors for training neural networks. Core tensor networks (Zhang et al., 2021) can transfer knowledge across multiple tensors. However, they are inapplicable to tensors that do not share modes, and tensors of different sizes. Also, existing attention-based methods for tensor data (Wang et al., 2023; Babiloni et al., 2020; Bai et al., 2018) cannot handle tensors of different sizes. The exchangeable tensor layers (Hartford et al., 2018) can handle tensors of different sizes, and they are used for transfer learning in tensor completion. However, they do not use attention mechanisms, and are not meta-learned using various tensors.

Meta-learning has been successfully used for improving performance with a limited number of observations by learning how to learn from various tasks (Garnelo et al., 2018; Finn et al., 2017; Vinyals et al., 2016; Li et al., 2017; Ravi and Larochelle, 2017; Snell et al., 2017). However, the existing meta-learning methods are inapplicable to tensor completion. In meta-

learning for regression tasks, it is effective in obtaining task-specific parameters adapted to the given data in a closed form (Bertinetto et al., 2018; Iwata and Tanaka, 2021), where linear models or Gaussian processes are used. In the proposed method, by obtaining an embedding vector for each element in the input tensor with missing values, a tensor completion task is transformed into a regression task. Therefore, we can perform the closed-form adaptation for tensor completion although the existing tensor completion methods require iterative optimization.

3 PRELIMINARIES

In this section, we describe the notation of tensors and their operations used in this paper. Let $\mathbf{X} \in \mathbb{R}^{N_1 \times \cdots \times N_M}$ be an Mth-order tensor, or a tensor with M modes. An element of tensor $\mathbf{X}$ is denoted by $\mathbf{X}(n_1, \ldots, n_M) \in \mathbb{R}$. A colon is used to indicate all elements of a mode. For example, $\mathbf{X}(n_1, \ldots, n_{M-1}, :) = [\mathbf{X}(n_1, \ldots, n_{M-1}, 1), \ldots, \mathbf{X}(n_1, \ldots, n_{M-1}, N_M)] \in \mathbb{R}^{N_M}$ is a vector called mode-M fiber (Kolda and Bader, 2009). The m-mode product of tensor $\mathbf{X}$ with matrix $\mathbf{W} \in \mathbb{R}^{N_m \times H}$ is denoted by $\mathbf{X} \times_m \mathbf{W} \in \mathbb{R}^{N_1 \times \cdots \times N_{m-1} \times H \times N_{m+1} \times \cdots \times N_M}$, where $(\mathbf{X} \times_m \mathbf{W})(n_1, \ldots, n_{m-1}, h, n_{m+1}, \ldots, n_M) = \sum_{n=1}^{N_m} \mathbf{X}(n_1, \ldots, n_{m-1}, n, n_{m+1}, \ldots, n_M)\mathbf{W}(n, h)$ elementwisely.

4 PROPOSED METHOD

4.1 Problem formulation

In the meta-training phase, we are given meta-training tensors, $\mathcal{D} = \{\mathbf{T}_d\}_{d=1}^D$, where $\mathbf{T}_d \in \mathbb{R}^{N_{d1} \times N_{d2} \times \cdots \times N_{dM}}$ is the dth tensor with M modes. The modes are not shared across tensors. The size for each mode can be different across tensors, i.e., $N_{dm} \neq N_{d'm}$. The tensors can contain missing values. We assume that all tensors have an identical number of modes M. An extension of our model to different numbers of modes is described in Section 4.4. In the meta-test phase, we are given a meta-test tensor with missing values $\mathbf{X} \in \mathbb{R}^{N_1 \times \cdots \times N_M}$, which is different from but related to the meta-training tensors. The missing values are set to zero in $\mathbf{X}$. Mask tensor $\mathbf{S} \in \{0, 1\}^{N_1 \times \cdots \times N_M}$ is also given for indicating observed elements, where $\mathbf{S}(n_1, \ldots, n_M) = 1$ if observed, and zero otherwize. Our aim is to improve the performance to predict missing values of meta-test tensors.

4.2 Model

Given an Mth-order observed tensor with missing value $\mathbf{X} \in \mathbb{R}^{N_1 \times \cdots \times N_M}$ and its mask tensor $\mathbf{S} \in \{0, 1\}^{N_1 \times \cdots \times N_M}$, the neural tensor attention network (NTAN) outputs its imputed values $\hat{\mathbf{X}} \in \mathbb{R}^{N_1 \times \cdots \times N_M}$. Figure 2 illustrates the architecture of the NTAN, and Algorithm 1 shows the forwarding pass. The NTAN has a multi-head m-mode tensor attention $m\text{MTA}_{\ell m}$ and a fiber-wise feed-forward network $\text{FFN}_{\ell m}$ for each layer ℓ and for each mode m.

Algorithm 1 Neural tensor attention network.

Input: Observed tensor with missing values $\mathbf{X}$, mask tensor $\mathbf{S}$.
Output: Imputed tensor $\hat{\mathbf{X}}$.
Obtain embedding tensor $\mathbf{Z}_{1,0}$ by Eq. (1).
for $\ell = 1$ **to** L **do**
  for $m = 1$ **to** M **do**
    Transform tensor by $\mathbf{O}_{\ell m} = m\text{MTA}_{\ell m}(\mathbf{Z}_{\ell,m-1})$ in Eq. (7).
    Transform tensor by $\mathbf{Z}_{\ell m} = \text{FFN}_{\ell m}(\mathbf{O}_{\ell m})$ in Eq. (8).
  end for
  Set $\mathbf{Z}_{\ell+1,0} = \mathbf{Z}_{\ell M}$.
end for
Set $\mathbf{Z} = \mathbf{Z}_{LM}$.
Obtain adapted prediction layer $\hat{\mathbf{W}}^{\text{L}}$ using $\mathbf{X}$, $\mathbf{S}$, $\mathbf{Z}$ in Eq. (10).
Obtain imputed tensor by $\hat{\mathbf{X}} = \mathbf{Z}' \times_{M+1} \hat{\mathbf{W}}^{\text{L}}$ in Eq. (9).

4.2.1 Embedding layer

First, by concatenating observed and mask tensors, $\mathbf{X}$ and $\mathbf{S}$, we obtain $(M+1)$th-order tensor, $\text{concat}_{M+1}(\mathbf{X}, \mathbf{S}) \in \mathbb{R}^{N_1 \times \cdots \times N_M \times 2}$, where concat_m represents the concatination of tensors along the mth mode. Then, the concatenated tensor is transformed by a linear layer,

$$\mathbf{Z}_{1,0} = \text{concat}_{M+1}(\mathbf{X}, \mathbf{S}) \times_{M+1} \mathbf{W}^{\text{E}}, \tag{1}$$

where $\mathbf{Z}_{1,0} \in \mathbb{R}^{N_1 \times \cdots \times N_M \times H}$ is the $(M+1)$th-order embedding tensor, and $\mathbf{W}^{\text{E}} \in \mathbb{R}^{2 \times H}$ is a linear projection matrix. By the embedding layer, we can encode information on observed values and missingness.

4.2.2 Multi-head m-mode tensor attention layer

The m-mode tensor attention layer (mTA) transforms $(M+1)$th-order tensor to another $(M+1)$th-order tensor by performing attention across subtensors sliced along the mth mode. An mTA can handle tensors with different sizes of modes except for the last mode, i.e., N_m can vary for $m = 1, \cdots, M$.

Let $\mathbf{Z}' \in \mathbb{R}^{N_1 \times \cdots \times N_M \times H}$ be an input tensor. First, the input tensor is linearly transformed to query $\mathbf{Q}$, key

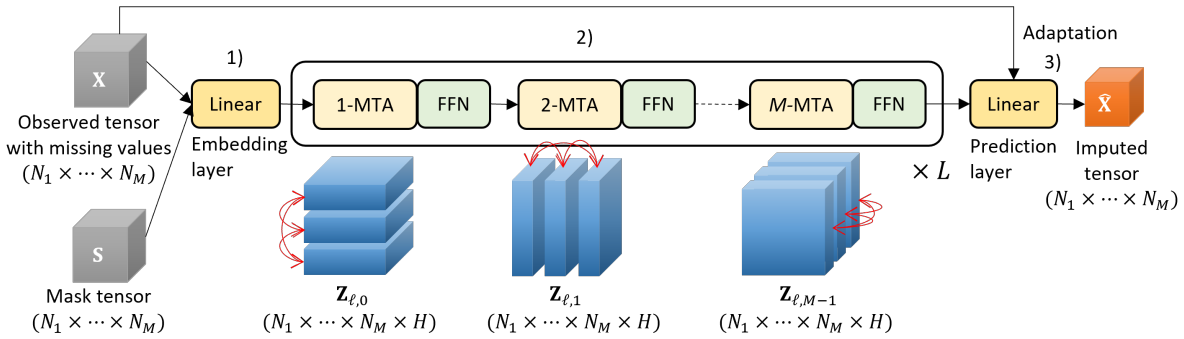


Figure 2: Neural tensor attention networks. 1) Observed and mask tensors, $\mathbf{X}$ and $\mathbf{S}$, are embedded into an $(M+1)$th-order tensor by a linear embedding layer described in Section 4.2.1. 2) The tensor is transformed by multi-head m-mode tensor attentions (mMTA) and fiber-wise feed-forward networks (FFNs) for each mode m alternately (Sections 4.2.2 and 4.2.3)). It is iterated L times. 3) Imputed tensor $\hat{\mathbf{X}}$ is obtained by a linear prediction layer that is adapted by given observed tensor $\mathbf{X}$ (Section 4.2.4).

$\mathbf{K}$, and value $\mathbf{V}$ tensors,

$$\mathbf{Q} = \mathbf{Z}' \times_{M+1} \mathbf{W}^{\mathrm{Q}} \in \mathbb{R}^{N_1 \times \cdots \times N_M \times H_{\mathrm{Q}}}, \quad (2)$$

$$\mathbf{K} = \mathbf{Z}' \times_{M+1} \mathbf{W}^{\mathrm{K}} \in \mathbb{R}^{N_1 \times \cdots \times N_M \times H_{\mathrm{Q}}}, \quad (3)$$

$$\mathbf{V} = \mathbf{Z}' \times_{M+1} \mathbf{W}^{\mathrm{V}} \in \mathbb{R}^{N_1 \times \cdots \times N_M \times H_{\mathrm{V}}}, \quad (4)$$

where $\mathbf{W}^{\mathrm{Q}} \in \mathbb{R}^{H \times H_{\mathrm{Q}}}$, $\mathbf{W}^{\mathrm{K}} \in \mathbb{R}^{H \times H_{\mathrm{Q}}}$, and $\mathbf{W}^{\mathrm{V}} \in \mathbb{R}^{H \times H_{\mathrm{V}}}$ are linear projection matrices.

Next, attention matrix $\mathbf{A}$ is obtained using query $\mathbf{Q}$ and key $\mathbf{K}$ tensors,

$$\mathbf{A} = \mathrm{softmax}\left(\frac{\mathbf{Q}_{(m)} \mathbf{K}_{(m)}^{\top}}{\sqrt{H_{\mathrm{Q}} \prod_{m' \neq m} N_{m'}}} \right) \in \mathbb{R}^{N_m \times N_m}, \quad (5)$$

where softmax is the softmax function, and $\mathbf{Q}_{(m)} \in \mathbb{R}^{N_1 \ldots N_{m-1} N_{m+1} \ldots N_M H_{\mathrm{Q}} \times N_m}$ and $\mathbf{K}_{(m)} \in \mathbb{R}^{N_1 \ldots N_{m-1} N_{m+1} \ldots N_M H_{\mathrm{Q}} \times N_m}$ are the m-mode matricization of tensors $\mathbf{Q}$ and $\mathbf{K}$, respectively. A value in the attention matrix $\mathbf{A}(n, n')$ represents relationship between the nth and n'th slices along the mth mode of input tensor $\mathbf{Z}'$.

The output of the mTA is obtained by m-mode product of value tensor $\mathbf{V}$ and attention matrix $\mathbf{A}$,

$$m\mathrm{TA}(\mathbf{Z}') = \mathbf{V} \times_m \mathbf{A} \in \mathbb{R}^{N_1 \times \cdots \times N_M \times H_{\mathrm{V}}}, \quad (6)$$

where the nth slice along the mth mode of the output tensor is calculated using other slices n' of the value tensor by weighting with their attention values $\mathbf{A}(n, n')$. With the mTA, the embedding of an element is updated by referring to the embeddings of other elements that compose similar subtensors sliced along the mth mode.

The multi-head version of the mTA is given by concatenating multiple mTAs,

$$\begin{aligned} \mathbf{O} &= \mathrm{concat}_{M+1}(m\mathrm{TA}_1(\mathbf{Z}'), \ldots, m\mathrm{TA}_R(\mathbf{Z}')) \times_{M+1} \mathbf{W}^{\mathrm{O}} \\ &\equiv m\mathrm{MTA}(\mathbf{Z}') \in \mathbb{R}^{N_1 \times \cdots \times N_M \times H_{\mathrm{O}}}, \end{aligned} \quad (7)$$

where $m\mathrm{TA}_r$ is the rth mTA, each head has different linear projection matrices, R is the number of heads, and $\mathbf{W}^{\mathrm{O}} \in \mathbb{R}^{H_{\mathrm{V}} R \times H_{\mathrm{O}}}$ is a linear projection matrix. The use of multiple heads allows us to capture various types of dependencies at different parts in the input tensor simultaneously. Although we omit mode, layer, and head indices in the parameters in Section 4.2.2 for simplicity, e.g., $\mathbf{W}^{\mathrm{Q}}$, the NTAN has parameters for each layer ℓ, for each mode m, and for each head r, e.g., $\mathbf{W}^{\mathrm{Q}}_{\ell m r}$.

When the input tensor is a second-order tensor, or matrix, the one-mode tensor attention corresponds to the vanilla attention (Vaswani et al., 2017). When the input tensor is a third-order tensor, the one-mode tensor attention corresponds to the variable-feature attention (Iwata and Kumagai, 2023).

4.2.3 Fiber-wise feed-forward networks

After a multi-head m-mode tensor attention (mMTA) layer, tensor $\mathbf{O}$ is nonlinearly transformed to another tensor $\mathbf{Z} \in \mathbb{R}^{N_1 \times \cdots \times N_M \times H}$ by a fiber-wise feed-forward network (FFN). As position-wise feed-forward networks in Transformer (Vaswani et al., 2017), FFNs can improve the expressive power of the NTAN. In particular, each mode-$(M+1)$ fiber $\mathbf{O}(n_1, \ldots, n_M, :) \in \mathbb{R}^{H_{\mathrm{O}}}$ is transformed by a residual feed-forward neural network with layer normalization (Ba et al., 2016),

$$\begin{aligned} &\mathbf{Z}(n_1, \ldots, n_M, :) = \mathbf{O}(n_1, \ldots, n_M, :)^{\top} \mathbf{W}^{\mathrm{R}} \\ &+ \mathrm{FF}(\mathrm{LN}(\mathbf{O}(n_1, \ldots, n_M, :))) \\ &\equiv \mathrm{FFN}(\mathbf{O}(n_1, \ldots, n_M, :)) \in \mathbb{R}^{H} \end{aligned} \quad (8)$$

where $\mathbf{W}^{\mathrm{R}} \in \mathbb{R}^{H_{\mathrm{O}} \times H}$ is a residual linear projection matrix, $\mathrm{FF} : \mathbb{R}^{H_{\mathrm{O}}} \to \mathbb{R}^{H}$ is a feed-forward neural network, and LN is a layer normalization. The parameters in $\mathbf{W}^{\mathrm{R}}$, FF, and LN are shared across all fibers.

As shown in Lines 5–8 in Algorithm 1, mMTAs and FFNs are alternately perform for each mode $m = 1, \ldots, M$, by which we can capture information on the input tensor in different aspects (modes). By iterating mMTAs and FFNs for M modes and L layers, we obtain tensor representation $\mathbf{Z} \in \mathbb{R}^{N_1 \times \cdots \times N_M \times H}$.

4.2.4 Adaptive Prediction layer

Imputed tensor $\hat{\mathbf{X}}$ of the same size with the input tensor $\mathbf{X}$ is obtained by linear projection of tensor representation $\mathbf{Z}$,

$$\hat{\mathbf{X}} = \mathbf{Z}^{\mathbf{1}} \times_{M+1} \hat{\mathbf{W}}^{\mathrm{L}} \equiv \mathrm{NTAN}(\mathbf{X}, \mathbf{S}) \in \mathbb{R}^{N_1 \times \cdots \times N_M}, \tag{9}$$

where $\mathbf{Z}^{1} = \mathrm{concat}_{M+1}(\mathbf{Z}, \mathbf{1}) \in \mathbb{R}^{N_1 \times \cdots \times N_M \times (H+1)}$, $\mathbf{1} \in \mathbb{R}^{N_1 \times \cdots \times N_M \times 1}$ is the tensor with value one for all elements, and $\hat{\mathbf{W}}^{\mathrm{L}} \in \mathbb{R}^{(H+1) \times 1}$ is a linear projection vector adapted to given observed tensor $\mathbf{X}$. Ones tensor $\mathbf{1}$ is concatenated to introduce a bias term. NTAN is our neural tensor attention network for tensor completion that takes observed $\mathbf{X}$ and mask $\mathbf{S}$ tensors as input. The adapted linear projection matrix $\hat{\mathbf{W}}^{\mathrm{L}}$ is obtained in a closed form by minimizing the squared Frobenius norm between the observed and imputed tensors with ℓ_2 regularization as follows,

$$\begin{aligned}
&\hat{\mathbf{W}}^{\mathrm{L}} \\
&= \operatorname*{argmin}_{\mathbf{W}^{\mathrm{L}}} \| \mathbf{S} \odot (\mathbf{X} - \mathbf{Z}^{\mathbf{1}} \times_{M+1} \mathbf{W}^{\mathrm{L}}) \|_{\mathrm{F}}^2 + \beta \| \mathbf{W}^{\mathrm{L}} \|_{\mathrm{F}}^2 \\
&= (\mathbf{Z}_{\mathbf{S}=1}^{\mathbf{1}\top} \mathbf{Z}_{\mathbf{S}=1}^{\mathbf{1}} + \beta \mathbf{I})^{-1} \mathbf{Z}_{\mathbf{S}=1}^{\mathbf{1}\top} \mathbf{X}_{\mathbf{S}=1},
\end{aligned} \tag{10}$$

where $\mathbf{Z}_{\mathbf{S}=1}^{\mathbf{1}} \in \mathbb{R}^{I \times (H+1)}$ is the matrix of mode-$(M+1)$ fibers of $\mathbf{Z}^{\mathbf{1}}$ at observed elements, $\mathbf{X}_{\mathbf{S}=1} \in \mathbb{R}^{I}$ is the vector of observed values in $\mathbf{X}$, $\| \cdot \|_{\mathrm{F}}$ is the Frobenius norm, $\odot$ is the element-wise product, I is the number of observed elements, $\beta \in \mathbb{R}_{>0}$ is a positive scalar parameter, and $\mathbf{I}$ is the identity matrix of size $(H+1) \times (H+1)$. Since NTAN can obtain a vector representation for each input tensor element $\mathbf{Z}(n_1, \ldots, n_m, :) \in \mathbb{R}^{H}$ even for missing elements, the prediction can be formulated as a linear regression problem, which enables us to adapt the prediction layer analytically as shown in Eq. (10).

Model paramters $\boldsymbol{\theta}$ in NTAN are a linear projection matrix in the embedding layer, $\mathbf{W}^{\mathrm{E}}$, linear projection matrices in mMTAs, $\{\{\{\mathbf{W}^{\mathrm{Q}}_{\ell m r}, \mathbf{W}^{\mathrm{K}}_{\ell m r}, \mathbf{W}^{\mathrm{V}}_{\ell m r}\}_{r=1}^{R}, \mathbf{W}^{\mathrm{O}}_{\ell m}\}_{m=1}^{M}\}_{\ell=1}^{L}$, and parameters in FFNs, which include parameters in feed-forward neural networks, $\{\{\mathrm{FF}_{\ell m}\}_{m=1}^{M}\}_{\ell=1}^{L}$, linear projection matrices, $\{\{\mathbf{W}^{\mathrm{R}}_{\ell m}\}_{m=1}^{M}\}_{\ell=1}^{L}$, parameters in layer normalizations $\{\{\mathrm{LN}_{\ell m}\}_{m=1}^{M}\}_{\ell=1}^{L}$, and ℓ_2 regularization weight β in the adaptive prediction layer. Parameters θ are shared across different tensors, by which we can learn common knowledge on tensor completion among heterogeneous tensors. All the parameters do not depend on input tensor size $(N_1, \ldots, N_M)$. Therefore, NTAN can predict missing values for tensors of different sizes.

Algorithm 2 Meta-learning procedures.

1: **Input:** Meta-training tensors $\mathcal{D} = \{\mathbf{T}_1, \ldots, \mathbf{T}_D\}$, taget tensor size $(N_1, \ldots, N_M)$, observed ratio μ.
2: **Output:** Trained model parameters $\boldsymbol{\theta}$.
3: **while** End condition is satisfied **do**
4: Randomly sample meta-training tensor index d from $\{1, \ldots, D\}$.
5: Randomly sample subtensor $\mathbf{X}$ of size $N_1 \times \cdots \times N_M$ from meta-training tensor $\mathbf{T}_d$.
6: Randomly generate mask tensor $\mathbf{S}$ of size $N_1 \times \cdots \times N_M$ according to observed ratio μ.
7: Predict missing values from observed tensor $\hat{\mathbf{X}} = \mathrm{NTAN}(\mathbf{X} \odot \mathbf{S}, \mathbf{S})$, where the prediction layer is optimized using observed values $\mathbf{X} \odot \mathbf{S}$.
8: Evaluate prediction error on missing elements $\| (1 - \mathbf{S}) \odot (\mathbf{X} - \hat{\mathbf{X}}) \|_{\mathrm{F}}^2$.
9: Update model parameters by minimizing the error by a stochastic gradient method.
10: **end while**

4.3 Meta-learning

Model parameters $\boldsymbol{\theta}$ are trained by minimizing the following expected test error of predicting missing values,

$$\mathbb{E}_d \mathbb{E}_{(\mathbf{X}, \mathbf{S}) \sim \mathbf{T}_d} \left[\| (1 - \mathbf{S}) \odot (\mathbf{X} - \mathrm{NTAN}(\mathbf{X} \odot \mathbf{S}, \mathbf{S})) \|_{\mathrm{F}}^2 \right], \tag{11}$$

where $\mathbb{E}_d$ is the expectation over meta-training tensors $d = 1, \ldots, D$, and $\mathbb{E}_{(\mathbf{X}, \mathbf{S}) \sim \mathbf{T}_d}$ is the expectation over observed and masked tensors randomly generated from the dth meta-training tensor $\mathbf{T}_d$. Here, $\mathbf{X} \odot \mathbf{S}$ is an observed tensor where its missing values are set to zero, which is used for the input of the NTAN. The error is calculated only on the elements that are not used for training by multiplying $(1 - \mathbf{S})$, which enables us to evaluate generalization performance. When meta-training tensors and meta-test tensors are generated from the same distribution, Eq. (11) converges to the true test error as the number of the meta-training tensors increases.

Algorithm 2 shows the meta-learning procedures of NTAN. The expectation in Eq. (11) is approximated by the Monte Carlo method. In Lines 4–5, a tensor is randomly generated from meta-training tensors. Here, the modes can be shuffled to increase the variety of

training tensors, $\mathbf{X}$ and $\mathbf{S}$. In Line 6, a mask tensor is randomly generated assuming that meta-test tensors are missing completely at random (Little and Rubin, 2019). When they are missing at random or missing not at random, we can model the missing mechanism using the meta-training tensors, and generate the mask tensor using the missing mechanism model. In Line 7, imputed tensor $\hat{\mathbf{X}}$ is obtained from observed tensor $\mathbf{X} \odot \mathbf{S}$ and its mask tensor $\mathbf{S}$. In Line 8, the prediction error at missing elements is evaluated.

When meta-training tensors contain missing values, elements of originally missing values are excluded from the prediction error evaluation. By minimizing the test error on various tensors randomly generated from meta-training tensors with an episodic training framework (Ravi and Larochelle, 2017) as described above, the NTAN is expected to perform well on unseen tensors. The proposed method is a meta-learning method that solves a bilevel optimization problem, where the inner optimization corresponds to the adaptation of the prediction layer by minimizing the error on observed values $\mathbf{X} \odot \mathbf{S}$ in Line 7, and the outer optimization corresponds to the update of model parameters shared across tensors by minimizing the error on held-out values $(1 - \mathbf{S}) \odot \mathbf{X}$ in Lines 8–9. The proposed method can be considered as self-supervised learning for tensors, where a model is trained by minimizing the prediction error at randomly set missing elements.

The computational complexity of m-mode tensor attention is $O(H_{\mathrm{Q}} N_m^2 \prod_{m' \neq m} N_{m'} + H_{\mathrm{V}} \prod_{m'} N_{m'})$. It linearly increases with the size of each mode except for mth mode, and quadratically increases with the size of the mth mode. The complexity of adapting the prediction layer is cubic to the number of hidden units $O(H^3)$, which is required for the inverse of matrix $\mathbf{Z}^{\mathbf{1}\top}_{\mathbf{S}=1}\mathbf{Z}^{\mathbf{1}}_{\mathbf{S}=1} + \beta\mathbf{I}$ of size $(H+1)\times(H+1)$ in Eq. (10).

Although we fix target tensor size $(N_1, \ldots, N_M)$ for simplicity in mete-learning, we can consider a range of target tensor sizes by defining minimum and maximum tensor sizes, $(N_1^{\min}, \ldots, N_M^{\min})$ and $(N_1^{\max}, \ldots, N_M^{\max})$, uniform randomly selecting a tensor size from them for each iteration. Similarly, we can consider a range of observed ratios for μ in meta-learning.

4.4 Permutation equivariance

An important property for modeling tensor data is permutation equivariance (Zaheer et al., 2017; Hartford et al., 2018). mMTAs and FFNs are permutation equivariant on elements for each mode except for the last mode. mMTAs are permutation equivariant on modes except for the mth and $(M+1)$th modes. Since the NTAN has mode-specific mMTAs, the NTAN is not permutation equivariant on modes. In Algorithm 1, mMTAs are sequentially performed for each mode in Line 6. Instead, by performing mMTAs (and FFN) parallelly and aggregating the outputs with average pooling, the transformation becomes permutation equivariant on all modes,

$$\mathbf{Z}_{\ell+1} = \frac{1}{M} \sum_{m=1}^{M} \mathrm{FFN}_\ell(m\mathrm{MTA}_\ell(\mathbf{Z}_\ell)), \qquad (12)$$

where $\mathbf{Z}_\ell$ is tensor representation at the ℓth layer, and the parameters in the mMTA and FFN are shared across all modes. When Eq. (12) is used, our model can transform tensors with different numbers of modes. Although traditional sequence-based attention mechanisms typically include positional encoding (Vaswani et al., 2017), since it eliminates the permutation equivariance property, NTAN does not use positional encoding.

5 EXPERIMENTS

5.1 Synthetic Data

Data First, we evaluated the proposed method using three types of synthetic tensors: low-rank, nonlinear, and random. Low-rank tensors were generated by a sum of two rank-one tensors, where rank-one tensors were obtained by the outer product of vectors, whose elements were normally distributed with mean zero and variance one. Nonlinear tensors were generated by nonlinearly transforming low-rank tensors via a three-layered feed-forward neural network. The number of hidden units was eight, the parameters in the neural network were uniform randomly determined in $[-1, 1]$, and the rectified linear unit was used for activation. Random tensors were generated uniform randomly in $[-1, 1]$ for each element. The size of all tensors was $10 \times 10 \times 10$.

Four cases of meta-training tensors were considered: LowR, NonL, Rand, and Mixed. Here, LowR (NonL, Rand) contains only low-rank (nonlinear, random) tensors, and Mixed contains all of the low-rank, nonlinear, and random tensors. Three cases of meta-test tensors were considered: LowR, NonL, and Rand. For each meta-test tensor, held-out elements were randomly selected with observed ratio $\mu \in \{0.1, 0.2, 0.3\}$ assuming missing completely at random. There was no overlap between meta-training and meta-test tensors. The values were normalized with mean zero and variance one for each tensor. The number of meta-training tensors was 700, and the number of meta-test tensors was 200 for each experiment. The experiments were conducted ten times.

Settings In NTAN, we used $L = 2$ layers of mMTAs with four heads and FFNs, where the linear projec-

Table 1: Test mean squared errors on synthetic tensors. Each column shows the method, where NTAN Mixed, LowR, NonL, Rand represent our method meta-trained with mixed, low-rank, nonlinear, and random tensors, respectively. CPD represents canonical polyadic decomposition. Each row shows the meta-test tensors, and μ represents the observed ratios of the meta-test tensors. Values in bold typeface are not statistically significantly different at the 5% level from the best performing method in each setting according to a paired t-test.

Test \ Train	μ	NTAN Mixed	NTAN LowR	NTAN NonL	NTAN Rand	CPD
LowR	0.1	0.531	**0.412**	0.698	1.000	1.273
	0.2	0.182	**0.099**	0.207	1.000	0.632
	0.3	0.068	**0.032**	0.115	1.000	0.310
NonL	0.1	0.569	1.037	**0.533**	0.999	1.156
	0.2	0.302	0.766	**0.271**	0.999	0.732
	0.3	0.188	0.815	**0.165**	1.001	0.500
Rand	0.1	1.004	1.400	1.200	**1.000**	1.760
	0.2	1.049	1.306	1.423	**1.000**	1.498
	0.3	1.002	1.401	1.263	**1.000**	1.371

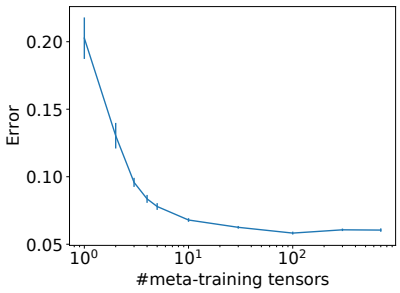


Figure 3: Test mean squared errors by our method on synthetic tensors with different numbers of meta-training tensors. Meta-training tensors are LowR, and meta-test tensors are LowR with observed ratio 0.2. Bar shows the standard error.

tion size was set to $H = H_{\mathrm{Q}} = H_{\mathrm{V}} = H_{\mathrm{O}} = 32$ for all layers, and FFNs were three-layered feed-forward neural networks with 32 hidden units. For the activation function, we used the rectified linear unit. We optimized models using Adam (Kingma and Ba, 2015) with learning rate 10^{-4}, and batch tensor size of eight. The meta-validation tensors were used for early stopping, for which the maximum number of meta-training epochs was 3,000. In meta-learning, we generated sub-tensors by shuffling modes with the same size and observed ratio with meta-test tensors.

Results Table 1 shows the test mean squared errors on meta-test tensors. Here, in addition to the proposed NTAN with different meta-training tensors (Mixed, LowR, NonL, and Rand), canonical polyadic decomposition (Hitchcock, 1927) (CPD) with rank two was compared. First, NTAN achieved the lowest error for different types of meta-test tensors when the types of meta-training and meta-test tensors were identical. This result indicates that NTAN can flexibly meta-learn different kinds of inductive bias based on the attention-based architecture from the meta-training tensors. Second, NTAN meta-learned with mixed tensors achieved the second best performance in all cases. This result implies that when different types of tensors are included in meta-training, NTAN can adaptively select inductive bias to use depending on the input tensor. Third, when meta-training and meta-test tensors were not similar (e.g., meta-trained with LowR and meta-tested with NonL), NTAN did not perform well. To improve performance with NTAN, it is beneficial to prepare meta-training tensors that are related to meta-test tensors. Fourth, NTAN meta-trained with random tensors did not improve the performance. When there is no pattern in meta-training tensors, we cannot learn inductive bias. Fifth, although CPD used inductive bias of low-rankness, NTAN (meta-trained with Mixed, LowR, and NonL) achieved the better performance than CPD for LowR meta-test tensors. It is because NTAN meta-learned other inductive bias than low-rankness, such as the distribution of rank-one tensors. Figure 3 shows that the test mean squared errors decreased as the number of meta-training tensors increased. By using more tensors, NTAN can meta-learn inductive bias more. Examples of imputed tensors by NTAN were shown in Figure 5 in Appendix B.2.

5.2 Real-world Data

Data Next, we evaluated the proposed method using five real-world third-order tensors: Flow, Enose, Amino, Sugar [1], and Traffic[2]. Flow, Enose, Amino, and Sugar tensors were data in chemometric. Flow tensor consists of data of flow injection analysis on chemical substances of size $12 \times 100 \times 89$ (substance $\times$ wavelength $\times$ reaction time). Enose tensor consists of electronic nose data of size $18 \times 241 \times 12$ (sample $\times$ reaction time $\times$ sensor). Amino tensor consists of amino acids fluorescence of size $5 \times 201 \times 61$ (sample $\times$ emission $\times$ excitation) (Bro, 1997). Sugar tensor consists of sugar flourescence of size $268 \times 571 \times 7$ (sample $\times$ emission $\times$ excitation) (Bro, 1999). Traffic tensor consists of vehicle counting in January 2019 by Grenoble Traffic Lab of size $31 \times 24 \times 21$ (day $\times$ hour $\times$ segment) (De Wit et al., 2015).

We constructed five datasets of meta-training and

[1]Data in chemometric were downloaded from `https://ucphchemometrics.com/datasets/`.

[2]Traffic tensor was downloaded from `https://gtl.inrialpes.fr/data_download`.

Table 2: Test mean squared errors with different observed ratios μ. Values in bold typeface are not statistically significantly different at the 5% level from the best performing method in each setting according to a paired t-test. The standard error is shown in Table 9 in Appendix B.3.

	μ	NTAN	MSEM	MFEA	SEM	FEA	CPD	TD	TTD	TRD	TSVD	USVD	FCTN
Flow	0.01	**0.951**	**0.950**	**0.973**	0.963	1.008	1.107	1.114	1.625	0.995	0.995	0.995	0.995
	0.05	**0.648**	0.810	0.816	0.834	0.971	1.099	1.112	1.206	0.999	0.972	0.929	0.997
	0.10	**0.447**	0.749	0.792	0.723	0.955	1.096	1.105	1.210	0.964	0.872	0.684	0.864
Enose	0.01	**0.068**	0.255	0.220	0.520	0.841	0.909	0.941	1.051	0.976	0.999	0.998	0.961
	0.05	**0.010**	0.048	0.063	0.254	0.704	0.045	1.020	0.116	0.405	0.972	0.835	0.421
	0.10	**0.008**	0.033	0.071	0.176	0.654	**0.009**	1.036	0.054	0.293	0.589	0.185	0.293
Amino	0.01	**0.887**	0.918	0.934	0.937	1.026	1.174	1.321	1.479	0.986	0.992	0.992	0.987
	0.05	**0.557**	0.658	0.737	0.785	0.957	0.974	1.085	1.113	0.953	0.992	0.966	0.915
	0.10	**0.330**	0.501	0.659	0.719	0.943	0.565	1.091	0.829	0.853	0.709	0.607	0.610
Sugar	0.01	**0.536**	0.586	0.657	0.666	0.856	1.097	1.106	1.144	0.997	0.997	0.997	0.996
	0.05	0.225	**0.157**	0.299	0.383	0.737	0.378	0.974	0.311	0.674	0.646	0.345	0.672
	0.10	0.210	**0.127**	0.176	0.312	0.782	0.179	1.042	0.184	0.474	0.341	0.174	0.472
Traffic	0.01	0.473	**0.447**	0.523	0.711	0.902	1.097	1.033	1.103	0.990	1.000	1.000	0.988
	0.05	**0.186**	0.279	0.322	0.475	0.812	0.385	1.063	0.447	0.558	0.482	0.379	0.437
	0.10	**0.130**	0.216	0.265	0.380	0.781	0.191	1.064	0.436	0.271	0.357	0.219	0.263

Table 3: Ablation study. Test mean squared errors averaged over all observed ratios and datasets, and their standard errors are shown. NoAdapt is NTAN without an adaptive prediction layer, where a linear layer is shared across all tensors. NoAttn is NTAN without attention mechanism, where exchangeable tensor layers were used instead of mMTAs and FFNs. NoShuf is NTAN without mode shuffling when generating subtensors in meta-learning. Share is NTAN that shares parameters of mMTAs and FFNs across different modes for each layer by Eq. (12).

NTAN	NoAdapt	NoAttn	NoShuf	Share
0.378	0.424	0.412	0.519	0.388

meta-test tensors, where one of the five tensors was used for meta-test, and the other four tensors were used for meta-training. We called a dataset using the name of the tensor used for meta-test; e.g., the Flow dataset consists of meta-test tensors generated from Flow tensor, and meta-training tensors of Enose, Amino, Sugar, and Traffic tensors. There was no overlap between meta-training and meta-test tensors. We randomly generated 50 meta-test tensors for each dataset by sampling subtensors such that the number of elements in each mode was not larger than 30, where all elements were used when the element size was not larger than 30. The values were normalized with mean zero and variance one for each tensor. For each meta-test tensor, held-out elements were randomly selected with observed ratio $\mu \in \{0.01, 0.05, 0.1\}$ assuming missing completely at random. The experiments were conducted ten times by permutating meta-training tensors, and resampling meta-test tensors. When the size of meta-training tensors was smaller than that of meta-test tensors, we fit the smaller one.

Comparing methods We compared the proposed NTAN method with the following methods: the self-supervised exchangeable model (Hartford et al., 2018) (SEM), the factorized exchangeable autoencoder (Hartford et al., 2018) (FAE), meta-learning of SEM and FAE (MSEM and MFAE), canonical polyadic decomposition (Hitchcock, 1927) (CPD), Tucker decomposition (Tucker, 1966) (TD), tensor train decomosition (Oseledets, 2011) (TTD), tensor ring decomposition (TRD) (Zhao et al., 2016), tensor SVD by Fourier transform (G.-J. Song and Zhang, 2020) (TSVD), tensor SVD by unitary transform (G.-J. Song and Zhang, 2020) (USVD), and fully-connected tensor network (Zheng et al., 2021) (FCTN). NTAN, MSEM, and MFAE are meta-learning methods, which are trained using meta-training tensors. Since there have been no existing meta-learning methods for tensor completion, we newly developed MSEM and MFAE as baselines. SEM, FAE, CPD, TD, TTD, TRD, TSVD, USVD, and FCTN are trained for each meta-test tensor. NTAN, MSEM, MFAE, SEM, and FAE are neural network-based methods, and the other methods are tensor decomposition-based methods. The detailed experimental settings of the compared methods are described in Appendix B.1.

Results Table 2 shows the test mean squared errors of missing value prediction in meta-test tensors averaged over ten experiments. The proposed NTAN achieved the lowest error in many cases. As the observed ratio increased, the error generally decreased in all methods. The errors by meta-learning methods (NTAN, MSEM, and MFEA) were lower than the other methods that were trained for each meta-test tensor. This result indicates that the meta-learning approach is effective for tensor completion when the

Table 4: Test mean squared errors when meta-learned with third-order tensors, and meta-tested with fifth-order tensors.

NTAN (Share)	SEM	FEA	CPD	TD	TTD	TRD	FCTN
0.715	0.906	1.004	0.846	1.058	0.842	0.846	0.934

Table 5: Test mean squared errors when meta-learned with target tensor size $30 \times 24 \times 21$ and meta-tested with tensors of size $30 \times 24 \times 1$ (or matrices of size 30×24) on Traffic dataset with different observed ratio μ. We omit TSVD and USVD since the implementation did not work for tensors with modes of size one.

μ	NTAN	MSEM	MFEA	SEM	FEA	CPD	TD	TTD	TRD	FCTN
0.01	0.985	1.177	1.256	**0.959**	1.005	1.131	1.366	1.304	1.022	1.022
0.05	**0.590**	0.788	0.779	0.706	0.896	0.926	1.132	1.133	1.023	1.023
0.10	**0.350**	0.512	0.472	0.522	0.762	0.551	0.899	0.775	0.923	0.915

number of observed elements is small. The better performance of the proposed method compared with MSEM and MFEA demonstrates the effectiveness of our neural network-based model.

Table 3 shows the results of the ablation study. When the prediction layer was not adapted for each tensor (NoAdapt), and when our tensor attention layers were not used (NoAttn), the performance decreased. This result indicates the effectiveness of the adaptation and attention in our model. Since we can train with more diverse tensors by shuffling modes in meta-learning, removing shuffling (NoShuf) increased the error. The sharing (Share) slightly degraded the performance.

The proposed method took 1.9 hours for meta-training, 1.3 seconds for meta-test on computers with Nvidia A100 GPU with 40GB memory and AMD EPYC 7262 CPU with 512GB memory for the Traffic dataset with observed ratio 0.01. Although the meta-training time was long, the proposed method efficiently performed tensor completion by using the meta-learned model.

When the proposed model shares the parameters across modes in Eq. (12), it can handle tensors with different numbers of modes. Table 4 shows the result when the number of modes of meta-test tensors is larger than that of meta-training tensors with observed ratio 0.1. NTAN with shared parameters was meta-learned using the Traffic dataset, which consists of tensors with three modes. The number of modes of meta-test tensors was five, where we used a tensor of enzymatic activity of size $3 \times 3 \times 3 \times 3 \times 5$ (Mortensen and Bro, 2006)[3]. The tensor contains the activity of polyphenol oxidase at different levels of O2, CO2, temperature, pH, and substrate according to a factorial design. Even when the number of modes in meta-test was larger than that in meta-training, the proposed method achieved low errors. MSEM and MFEA are inapplicable to tensors with different numbers of modes.

When the number of modes of meta-test tensors is smaller than that of meta-training tensors, The proposed method is directly applicable by adding modes of size one to the meta-test tensors to align the number of modes. Table 5 shows that the proposed method performed well in such a case.

Tables 7 and 8 in Appendix B.3 show that even when the number of elements in each mode is different between meta-test and meta-training tensors, the proposed method can use the meta-learned knowledge, and achieved high performance.

Additional experimental results, such as the results with different hyperparameters, and those with different numbers of meta-training tensors, are shown in Appendix B.3.

6 CONCLUSION

We proposed neural attention tensor networks for tensor completion, which is meta-learned using various tensors without shared modes. The experimental results show that our method achieves the high tensor completion performance with a small number of observations compared with the existing methods.

Although we believe that our work is an important step for learning from a wide variety of datasets, we must extend our approach in several directions. First, we plan to develop our method for tensors with missing not at random. Second, we want to extend our method to be able to handle heterogeneous types of elements, e.g., categorical values and natural language. Third, for handling tensors with a large number of elements, we need to use scalable attention models (Ainslie et al., 2020; Zaheer et al., 2020; Grefenstette et al., 2019).

[3]The enzymatic activity data were downloaded from `https://ucphchemometrics.com/datasets/`.

References

J. Ainslie, S. Ontanon, C. Alberti, V. Cvicek, Z. Fisher, P. Pham, A. Ravula, S. Sanghai, Q. Wang, and L. Yang. ETC: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 268–284, 2020.

J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

F. Babiloni, I. Marras, G. Slabaugh, and S. Zafeiriou. TESA: Tensor element self-attention via matricization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13945–13954, 2020.

Y. Bai, J. Fu, T. Zhao, and T. Mei. Deep attention neural tensor network for visual question answering. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 20–35, 2018.

L. Bertinetto, J. F. Henriques, P. Torr, and A. Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2018.

R. Bro. PARAFAC. tutorial and applications. *Chemometrics and Intelligent Laboratory Systems*, 38(2): 149–171, 1997.

R. Bro. Exploratory study of sugar production using fluorescence spectroscopy and multi-way analysis. *Chemometrics and Intelligent Laboratory Systems*, 46(2):133–147, 1999.

C. Bugg, C. Chen, and A. Aswani. Nonnegative tensor completion via integer optimization. *Advances in Neural Information Processing Systems*, 35:10008–10020, 2022.

Z. Chen, Z. Xu, and D. Wang. Deep transfer tensor decomposition with orthogonal constraint for recommender systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4010–4018, 2021.

C. C. De Wit, F. Morbidi, L. L. Ojeda, A. Y. Kibangou, I. Bellicot, and P. Bellemain. Grenoble traffic lab: An experimental platform for advanced traffic monitoring and forecasting. *IEEE Control Systems Magazine*, 35(3):23–39, 2015.

C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1126–1135, 2017.

M. K. N. G.-J. Song and X.-J. Zhang. Robust tensor completion using transformed tensor singular value decomposition. *Numerical Linear Algebra with Applications*, 27:e2299, 2020.

M. Garnelo, D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. Rezende, and S. A. Eslami. Conditional neural processes. In *International Conference on Machine Learning*, pages 1690–1699, 2018.

L. Grasedyck. Hierarchical singular value decomposition of tensors. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2029–2054, 2010.

E. Grefenstette, B. Amos, D. Yarats, P. M. Htut, A. Molchanov, F. Meier, D. Kiela, K. Cho, and S. Chintala. Generalized inner loop meta-learning. *arXiv preprint arXiv:1910.01727*, 2019.

J. Hartford, D. Graham, K. Leyton-Brown, and S. Ravanbakhsh. Deep models of interactions across sets. In *International Conference on Machine Learning*, pages 1909–1918, 2018.

F. L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

S. Ibrahim, X. Fu, R. Hutchinson, and E. Seo. Undercounted tensor completion with neural incorporation of attributes. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 14283–14315, 2023.

T. Iwata and A. Kumagai. Meta-learning of semi-supervised learning from tasks with heterogeneous attribute spaces. *arXiv preprint arXiv:2311.05088*, 2023.

T. Iwata and Y. Tanaka. Few-shot learning for spatial regression via neural embedding-based Gaussian processes. *Machine Learning*, pages 1–19, 2021.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

T. Lacroix, N. Usunier, and G. Obozinski. Canonical tensor decomposition for knowledge base completion. In *International Conference on Machine Learning*, pages 2863–2872. PMLR, 2018.

C. Lee and M. Wang. Beyond the signs: Nonparametric tensor completion via sign series. *Advances in Neural Information Processing Systems*, 34:21782–21794, 2021.

C. Li and Z. Sun. Evolutionary topology search for tensor network decomposition. In *International Conference on Machine Learning*, pages 5947–5957, 2020.

Y. Li, W. Liang, K. Xie, D. Zhang, S. Xie, and K. Li. LightNestle: quick and accurate neural sequential

tensor completion via meta learning. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2023.

Z. Li, F. Zhou, F. Chen, and H. Li. Meta-SGD: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

R. J. Little and D. B. Rubin. *Statistical analysis with missing data*, volume 793. John Wiley & Sons, 2019.

A. Liu and A. Moitra. Tensor completion made practical. *Advances in Neural Information Processing Systems*, 33:18905–18916, 2020.

H. Liu, Y. Li, M. Tsang, and Y. Liu. Costco: A neural tensor completion model for sparse tensors. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 324–334, 2019.

J. Liu, P. Musialski, P. Wonka, and J. Ye. Tensor completion for estimating missing values in visual data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):208–220, 2012.

J. Mocks. Topographic components model for event-related potentials and some biophysical considerations. *IEEE Transactions on Biomedical Engineering*, 35(6):482–484, 1988.

S. M. Mohammadi, S. Kouchaki, S. Sanei, D.-J. Dijk, A. Hilton, and K. Wells. Tensor factorisation and transfer learning for sleep pose detection. In *European Signal Processing Conference*, pages 1–5, 2019.

P. P. Mortensen and R. Bro. Real-time monitoring and chemical profiling of a cultivation process. *Chemometrics and Intelligent Laboratory Systems*, 84(1-2): 106–113, 2006.

D. Muti and S. Bourennane. Multidimensional filtering based on a tensor approach. *Signal Processing*, 85 (12):2338–2353, 2005.

M. Nimishakavi, P. K. Jawanpuria, and B. Mishra. A dual framework for low-rank tensor completion. *Advances in Neural Information Processing Systems*, 31, 2018.

I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017.

N. Razin, A. Maman, and N. Cohen. Implicit regularization in tensor factorization. In *International Conference on Machine Learning*, pages 8913–8924. PMLR, 2021.

B. Romera-Paredes and M. Pontil. A new convex relaxation for tensor completion. *Advances in Neural Information Processing Systems*, 26, 2013.

N. D. Sidiropoulos, R. Bro, and G. B. Giannakis. Parallel factor analysis in sensor array processing. *IEEE Transactions on Signal Processing*, 48(8):2377–2388, 2000.

J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

R. Socher, D. Chen, C. D. Manning, and A. Ng. Reasoning with neural tensor networks for knowledge base completion. *Advances in Neural Information Processing Systems*, 26, 2013.

Q. Song, H. Ge, J. Caverlee, and X. Hu. Tensor completion algorithms in big data analytics. *ACM Transactions on Knowledge Discovery from Data*, 13 (1):1–48, 2019.

K. Takeuchi, R. Tomioka, K. Ishiguro, A. Kimura, and H. Sawada. Non-negative multiple tensor factorization. In *2013 IEEE 13th International Conference on Data Mining*, pages 1199–1204. IEEE, 2013.

R. Tomioka, T. Suzuki, K. Hayashi, and H. Kashima. Statistical performance of convex tensor decomposition. *Advances in Neural Information Processing Systems*, 24, 2011.

L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

M. Usvyatsov, R. Ballester-Ripoll, and K. Schindler. tntorch: Tensor network learning with PyTorch. *The Journal of Machine Learning Research*, 23(1): 9394–9399, 2022.

M. A. O. Vasilescu and D. Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. In *European Conference on Computer Vision*, pages 447–460. Springer, 2002.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

J. Wang, A. Qu, Q. Wang, Q. Zhao, J. Liu, and Q. Wu. TT-Net: Tensorized transformer network

for 3D medical image segmentation. *Computerized Medical Imaging and Graphics*, 107:102234, 2023.

Z.-C. Wu, T.-Z. Huang, L.-J. Deng, H.-X. Dou, and D. Meng. Tensor wheel decomposition and its tensor completion application. *Advances in Neural Information Processing Systems*, 35:27008–27020, 2022.

J. Xue, Y. Zhao, S. Huang, W. Liao, J. C.-W. Chan, and S. G. Kong. Multilayer sparsity-based tensor decomposition for low-rank tensor completion. *IEEE Transactions on Neural Networks and Learning Systems*, 33(11):6916–6930, 2021.

C. Yang, C. Qian, N. Singh, C. D. Xiao, M. Westover, E. Solomonik, and J. Sun. ATD: Augmenting CP tensor decomposition by self supervision. *Advances in Neural Information Processing Systems*, 35:32039–32052, 2022.

M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. In *Advances in Neural Information Processing Systems*, pages 3391–3401, 2017.

M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, et al. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297, 2020.

J. Zhang, Z. Tao, L. Zhang, and Q. Zhao. Tensor decomposition via core tensor networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2130–2134, 2021.

Z. Zhang, J. Cai, and J. Wang. Duality-induced regularizer for tensor factorization based knowledge graph completion. *Advances in Neural Information Processing Systems*, 33:21604–21615, 2020.

Q. Zhao, G. Zhou, S. Xie, L. Zhang, and A. Cichocki. Tensor ring decomposition. *arXiv preprint arXiv:1606.05535*, 2016.

X.-L. Zhao, W.-H. Xu, T.-X. Jiang, Y. Wang, and M. K. Ng. Deep plug-and-play prior for low-rank tensor completion. *Neurocomputing*, 400:137–149, 2020.

Y.-B. Zheng, T.-Z. Huang, X.-L. Zhao, Q. Zhao, and T.-X. Jiang. Fully-connected tensor network decomposition and its application to higher-order tensor completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11071–11078, 2021.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes] We described them in Section 4.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes] The analysis of the properties is described in Section 5, and the complexity is described in Section 4.3.
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [No]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
   (b) Complete proofs of all theoretical results. [Not Applicable]
   (c) Clear explanations of any assumptions. [Not Applicable]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [No] The code is proprietary. The details of our experiments were described in Section 5 and Appendix B.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes] The training details were described in Section 5 and Appendix B.1.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes] The definition, statistical test results, and error bars are described in Section 5 and Appendix B.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes] It is described in the last paragraph of Section 5.2.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]

(d) Information about consent from data providers/curators. [Not Applicable]

(e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

(a) The full text of instructions given to participants and screenshots. [Not Applicable]

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

A Proposed method

Figure 4 illustrates the two-mode tensor attention layer of a fourth-order tensor.

B EXPERIMENTS

B.1 Settings

SEM and FAE used two layers of exchangeable tensor layers (Hartford et al., 2018) with 32 hidden units, where exchangeable tensor layers can perform permutation equivariant nonlinear transformation based on neural networks but no attention mechanisms. The prediction layer is linear in SEM, and it is factorized matrices with rank three in FAE. The factorized matrices were obtained using three-layered feed-forward neural networks with 32 hidden units with average pooling. SEM and FAE were trained with 100 training epochs for each meta-test tensor. In MSEM and MFAE, the same meta-learning procedures were used as NTAN. NTAN, MSEM, MFAE, SEM, and FAE were implemented with PyTorch (Paszke et al., 2019). In CPD, TD, TTD, TRD, and FCTN, the result with the best-performed rank selected from $\{2, 4, 8, 16, 32\}$ was shown. TTD, CPD, and TD were

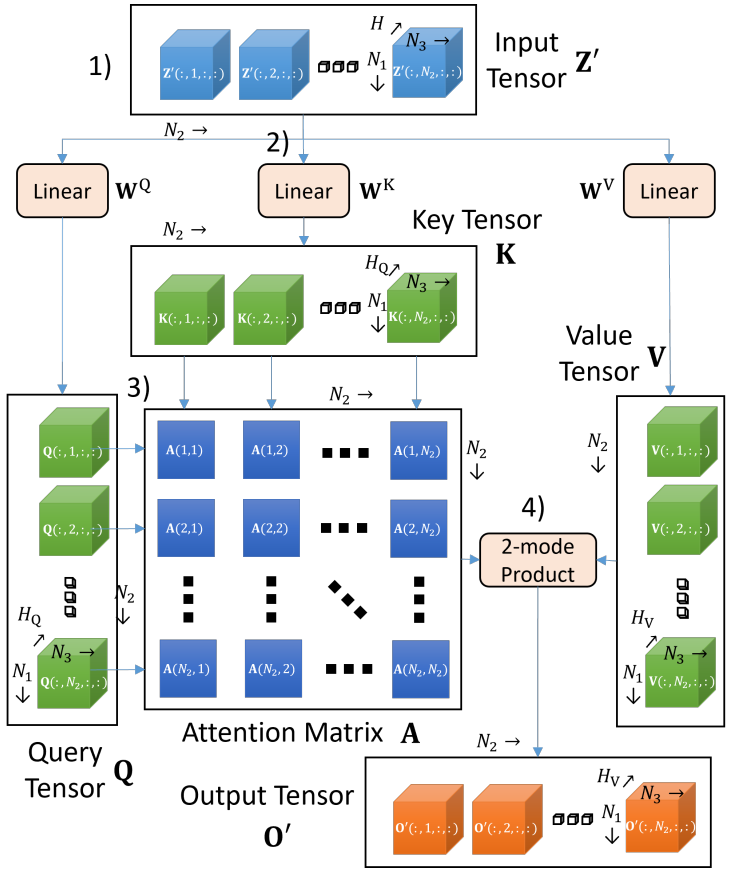
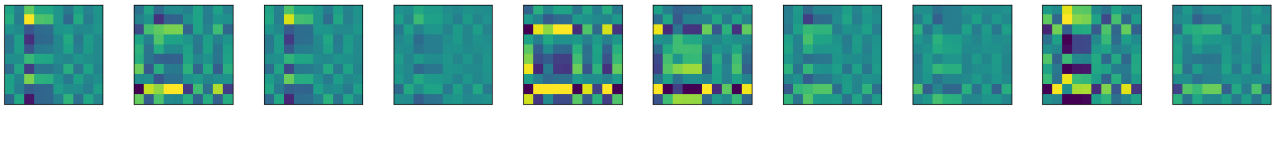
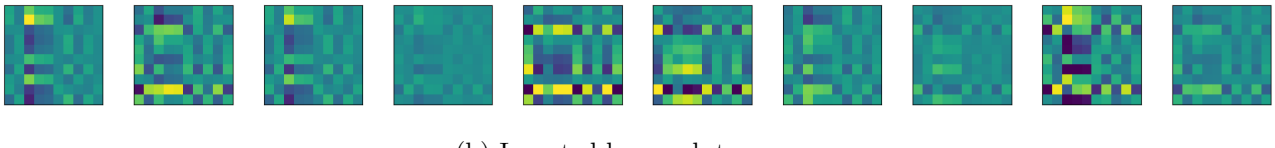


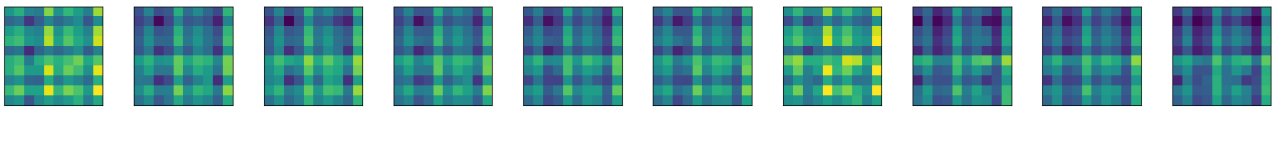
Figure 4: Two-mode tensor attention of a fourth-order tensor input. 1) Input tensor $\mathcal{Z}'$ of size $N_1 \times N_2 \times N_3 \times H$ is represented by a set of N_2 subtensors of size $N_1 \times N_3 \times H$, $\{\mathcal{Z}'(:,1,:,:), \mathcal{Z}'(:,2,:,:), \ldots, \mathcal{Z}'(:,N_2,:,:)\}$, sliced over elements of the second mode. 2) Input tensor $\mathcal{Z}'$ is linearly transformed to query $\mathcal{Q} \in \mathbb{R}^{N_1 \times N_2 \times N_3 \times H_{\mathrm{Q}}}$, key $\mathcal{K} \in \mathbb{R}^{N_1 \times N_2 \times N_3 \times H_{\mathrm{Q}}}$, and value $\mathcal{V} \in \mathbb{R}^{N_1 \times N_2 \times N_3 \times H_{\mathrm{V}}}$ tensors. 3) Attention matrix $\mathbf{A} \in \mathbb{R}^{N_2 \times N_2}$ is calculated from the query and key tensors. Each element of attention matrix $\mathbf{A}(n, n')$ represents relationship between the nth query subtensor $\mathcal{Q}(:,n,:,:)$ and the n'th key subtensor $\mathcal{K}(:,n',:,:)$. 4) By two-mode product of attention matrix $\mathbf{A}$ and value tensor $\mathcal{V}$, output tensor $\mathcal{O}' \in \mathbb{R}^{N_1 \times N_2 \times N_3 \times H_{\mathrm{V}}}$ is obtained, where each subtensor is calculated by $\mathcal{O}'(:,n,:,:) = \sum_{n'=1}^{N_2} A(n,n')\mathcal{V}(:,n',:,:)$.



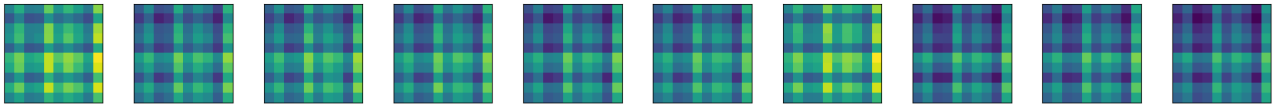
(a) True low-rank tensor



(b) Imputed low-rank tensor



(c) True nonlinear tensor



(c) Imputed nonlinear tensor

Figure 5: Examples of true tensors and imputed meta-test tensors by NTAN meta-trained with mixed tensors with observed ratio 0.2.

implemented with tntorch (Usvyatsov et al., 2022). Tensor Network Toolbox [4] was used for TRD and FCTN. The authors' implementation was used for TSVD and USVD [5]. We used their default hyperparameters.

The evaluation measurement was the test mean squared errors on the meta-test tensors,

$$\frac{1}{D^{\mathrm{T}}}\sum_{d=1}^{D^{\mathrm{T}}}\frac{\| \left(1-\mathbf{S}_d^{\mathrm{T}}\right)\odot\left(\mathbf{X}_d^{\mathrm{T}}-\mathrm{NTAN}(\mathbf{X}_d^{\mathrm{T}}\odot\mathbf{S}_d^{\mathrm{T}},\mathbf{S}_d^{\mathrm{T}})\right)\|_{\mathrm{F}}^2}{|1-\mathbf{S}_d^{D^{\mathrm{T}}}|}, \tag{13}$$

where D^{T} is the number of meta-test tensors, $\mathbf{X}_d^{D^{\mathrm{T}}}$ is the dth meta-test tensor, $\mathbf{S}_d^{D^{\mathrm{T}}}$ is its mask tensor, and $|1-\mathbf{S}_d^{D^{\mathrm{T}}}|$ is the number of missing elements.

B.2 Results on synthetic data

Figure 5 shows examples of true tensors and imputed meta-test tensors by NTAN meta-trained with mixed tensors. Here, the low-rank and nonlinear tensors were imputed using a meta-trained NTAN with the same parameters. This result suggests that NTAN can impute different patterns of tensors with a single model adaptively depending on the input tensor with missing values. Figure 6 shows the test mean squared errors by our method with the nonlinear tensors.

B.3 Results on real-world data

Table 6 shows the performance when the maximum number of elements in each mode was set to 20 for meta-test tensors. The model was meta-learned using target tensors of the same size as meta-test tensors. NTAN outperformed the other methods with smaller target tensor sizes. Since the number of observed elements increases

[4] https://github.com/YuBangZheng/TenNet_ToolBox
[5] https://github.com/xjzhang008/Transformed-Tensor-SVD

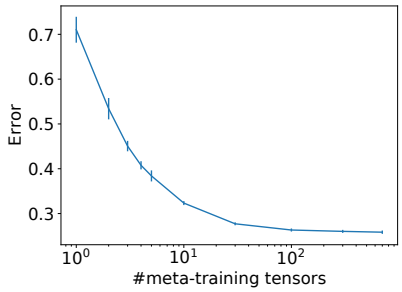


Figure 6: Test mean squared errors by our method on synthetic tensors with different numbers of meta-training tensors. Meta-training tensors are NonL, and meta-test tensors are NonL with observed ratio 0.2. Bar shows the standard error.

Table 6: Test mean squared errors with different observed ratios μ with meta-test tensor size $20 \times 20 \times 20$ at maximum. Values in bold typeface are not statistically significantly different at the 5% level from the best performing method in each setting according to a paired t-test.

	μ	NTAN	MSEM	MFEA	SEM	FEA	CPD	TD	TTD	TRD	TSVD	USVD	FCTN
Flow	0.01	0.987	1.006	1.007	**0.972**	1.029	1.199	1.352	1.660	0.992	0.992	0.992	0.992
	0.05	**0.669**	0.797	0.824	0.853	0.974	1.094	1.097	1.170	1.000	0.987	0.979	0.999
	0.10	**0.470**	0.744	0.788	0.749	0.962	1.010	1.099	1.191	0.985	0.886	0.767	0.992
Enose	0.01	**0.113**	0.388	0.367	0.569	0.859	1.076	1.034	1.163	0.988	1.000	1.000	0.985
	0.05	**0.013**	0.050	0.044	0.286	0.723	0.116	0.975	0.129	0.449	0.956	0.846	0.459
	0.10	**0.010**	0.040	0.080	0.191	0.662	**0.012**	1.041	0.023	0.334	0.600	0.248	0.333
Amino	0.01	**0.956**	0.974	0.981	0.984	1.067	1.190	1.581	1.417	1.000	1.004	1.004	0.999
	0.05	**0.658**	0.740	0.785	0.829	0.975	1.093	1.084	1.246	0.975	1.000	0.996	0.977
	0.10	**0.417**	0.578	0.691	0.751	0.955	0.798	1.081	1.039	0.936	0.794	0.708	0.835
Sugar	0.01	**0.663**	0.713	0.780	0.727	0.884	1.074	1.189	1.164	0.982	0.983	0.983	0.982
	0.05	0.275	**0.192**	0.355	0.430	0.745	0.607	0.854	0.508	0.847	0.768	0.483	0.849
	0.10	0.196	**0.135**	0.192	0.334	0.702	0.269	0.950	0.226	0.553	0.414	0.249	0.552
Traffic	0.01	0.541	**0.509**	0.548	0.734	0.911	1.110	1.088	1.291	0.986	0.995	0.995	0.989
	0.05	**0.215**	0.306	0.331	0.515	0.836	0.520	1.062	0.573	0.710	0.509	0.426	0.567
	0.10	**0.142**	0.228	0.268	0.401	0.789	0.279	1.062	0.380	0.449	0.369	0.261	0.321

Table 7: Test mean squared errors when meta-learned with target tensor size $30 \times 24 \times 21$ and meta-tested with tensors of size $20 \times 20 \times 20$ on Traffic dataset with different observed ratio μ.

μ	NTAN	MSEM	MFEA	SEM	FEA	CPD	TD	TTD	TRD	TSVD	USVD	FCTN
0.01	**0.479**	0.501	0.586	0.734	0.911	1.110	1.088	1.291	0.986	0.995	0.995	0.989
0.05	**0.199**	0.302	0.323	0.515	0.836	0.520	1.062	0.573	0.710	0.509	0.426	0.567
0.10	**0.135**	0.232	0.261	0.401	0.789	0.279	1.062	0.380	0.449	0.369	0.261	0.321

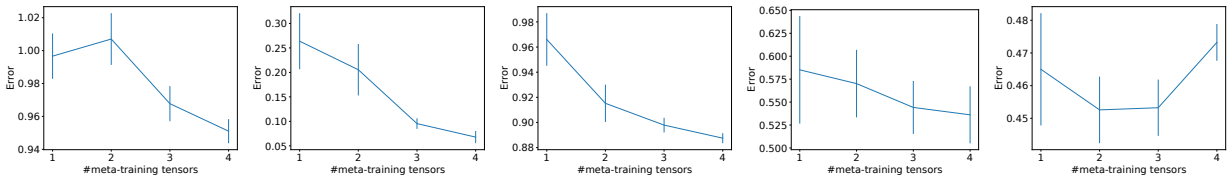
as the tensor size increases, the errors in Table 6 were generally lower than those in Table 2 with larger meta-test tensor sizes.

Tables 7 and 8 show the results when the number of elements in each mode is different betwenn meta-test and meta-training tensors.

Figure 7 shows that the test mean squared errors decreased as the number of meta-training tensors increased except for the Traffic dataset. The proposed method meta-learned a wide variety of patterns in the meta-training tensors with different kinds of modes, and adequately used them for improving the tensor completion performance on meta-test tensors.

Figure 8 shows the results with different numbers of layers of of mMTAs and FFNs, hidden units in mMTAs and FFNs, and heads in mMTAs.

Table 9 shows the standard errors of Table 2 in Section 5.2.

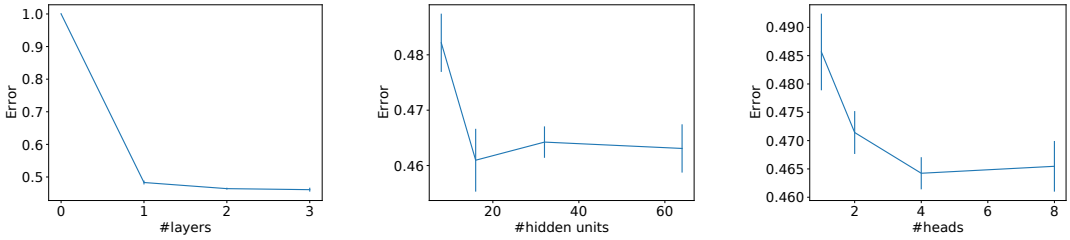


(a) Flow (b) Enose (c) Amino (d) Sugar (e) Traffic

Figure 7: Test mean squared errors by our method with different numbers of meta-training tensors on real-world tensors with observed ratio 0.01. Bar shows the standard error.

Table 8: Test mean squared errors when meta-learned with target tensor size $20 \times 20 \times 20$ and meta-tested with tensors of size $30 \times 24 \times 21$ on Traffic dataset with different observed ratio μ.

μ	NTAN	MSEM	MFEA	SEM	FEA	CPD	TD	TTD	TRD	TSVD	USVD	FCTN
0.01	0.592	**0.447**	0.486	0.711	0.902	1.097	1.033	1.103	0.990	1.000	1.000	0.988
0.05	**0.207**	0.286	0.329	0.475	0.812	0.385	1.063	0.447	0.558	0.482	0.379	0.437
0.10	**0.140**	0.216	0.272	0.380	0.781	0.191	1.064	0.436	0.271	0.357	0.219	0.263



(a) #Layers (b) #Hidden units (c) #Heads

Figure 8: Test mean squared errors by our method with different numbers of (a) layers, (b) hidden units, and (c) heads on Traffic dataset with observed ratio 0.01. Bar shows the standard error.

C LIMITATIONS

Our proposed method requires multiple meta-training tensors to improve performance. When meta-training tensors and meta-test tensors are significantly different, the performance might be degraded. We focus on improving the performance when there are few observed elements in a tensor. When a sufficient number of observed values in a tensor are available, it might be better to train models for each tensor. The proposed method requires a long computational time for meta-learning with large tensors. For handling tensors with a huge number of elements, we need to use scalable attention models (Ainslie et al., 2020; Zaheer et al., 2020; Grefenstette et al., 2019).

Table 9: Test mean squared errors and their standard errors with different observed ratios μ. Values in bold typeface are not statistically significantly different at the 5% level from the best performing method in each setting according to a paired t-test.

	μ	NTAN	MSEM	MFEA	SEM	FEA	CPD
Flow	0.01	**0.951 ± 0.007**	**0.950 ± 0.006**	**0.973 ± 0.014**	0.963 ± 0.006	1.008 ± 0.005	1.107 ± 0.006
	0.05	**0.648 ± 0.007**	0.810 ± 0.004	0.816 ± 0.004	0.834 ± 0.003	0.971 ± 0.003	1.099 ± 0.003
	0.10	**0.447 ± 0.010**	0.749 ± 0.006	0.792 ± 0.008	0.723 ± 0.003	0.955 ± 0.003	1.096 ± 0.002
Enose	0.01	**0.068 ± 0.012**	0.255 ± 0.012	0.220 ± 0.020	0.520 ± 0.002	0.841 ± 0.003	0.909 ± 0.010
	0.05	**0.010 ± 0.001**	0.048 ± 0.002	0.063 ± 0.007	0.254 ± 0.001	0.704 ± 0.002	0.045 ± 0.001
	0.10	**0.008 ± 0.000**	0.033 ± 0.001	0.071 ± 0.008	0.176 ± 0.001	0.654 ± 0.002	**0.009 ± 0.001**
Amino	0.01	**0.887 ± 0.004**	0.918 ± 0.006	0.934 ± 0.005	0.937 ± 0.005	1.026 ± 0.005	1.174 ± 0.008
	0.05	**0.557 ± 0.004**	0.658 ± 0.003	0.737 ± 0.003	0.785 ± 0.003	0.957 ± 0.003	0.974 ± 0.004
	0.10	**0.330 ± 0.006**	0.501 ± 0.003	0.659 ± 0.005	0.719 ± 0.005	0.943 ± 0.006	0.565 ± 0.005
Sugar	0.01	**0.536 ± 0.031**	0.586 ± 0.033	0.657 ± 0.034	0.666 ± 0.031	0.856 ± 0.030	1.097 ± 0.032
	0.05	0.225 ± 0.021	**0.157 ± 0.006**	0.299 ± 0.015	0.383 ± 0.017	0.737 ± 0.027	0.378 ± 0.014
	0.10	0.210 ± 0.015	**0.127 ± 0.003**	0.176 ± 0.010	0.312 ± 0.010	0.782 ± 0.028	0.179 ± 0.003
Traffic	0.01	0.473 ± 0.006	**0.447 ± 0.002**	0.523 ± 0.013	0.711 ± 0.002	0.902 ± 0.002	1.097 ± 0.001
	0.05	**0.186 ± 0.003**	0.279 ± 0.001	0.322 ± 0.004	0.475 ± 0.000	0.812 ± 0.002	0.385 ± 0.003
	0.10	**0.130 ± 0.002**	0.216 ± 0.002	0.265 ± 0.002	0.380 ± 0.000	0.781 ± 0.001	0.191 ± 0.002

	μ	TD	TTD	TRD	TSVD	USVD	FCTN
Flow	0.01	1.114 ± 0.005	1.625 ± 0.011	0.995 ± 0.005	0.995 ± 0.005	0.995 ± 0.005	0.995 ± 0.005
	0.05	1.112 ± 0.003	1.206 ± 0.004	0.999 ± 0.003	0.972 ± 0.004	0.929 ± 0.003	0.997 ± 0.003
	0.10	1.105 ± 0.003	1.210 ± 0.004	0.964 ± 0.003	0.872 ± 0.003	0.684 ± 0.003	0.864 ± 0.004
Enose	0.01	0.941 ± 0.014	1.051 ± 0.023	0.976 ± 0.001	0.999 ± 0.001	0.998 ± 0.001	0.961 ± 0.002
	0.05	1.020 ± 0.009	0.116 ± 0.008	0.405 ± 0.004	0.972 ± 0.004	0.835 ± 0.006	0.421 ± 0.005
	0.10	1.036 ± 0.005	0.054 ± 0.007	0.293 ± 0.004	0.589 ± 0.005	0.185 ± 0.005	0.293 ± 0.003
Amino	0.01	1.321 ± 0.024	1.479 ± 0.008	0.986 ± 0.004	0.992 ± 0.004	0.992 ± 0.004	0.987 ± 0.004
	0.05	1.085 ± 0.004	1.113 ± 0.005	0.953 ± 0.003	0.992 ± 0.003	0.966 ± 0.004	0.915 ± 0.007
	0.10	1.091 ± 0.005	0.829 ± 0.006	0.853 ± 0.005	0.709 ± 0.005	0.607 ± 0.004	0.610 ± 0.007
Sugar	0.01	1.106 ± 0.037	1.144 ± 0.033	0.997 ± 0.036	0.997 ± 0.036	0.997 ± 0.036	0.996 ± 0.036
	0.05	0.974 ± 0.031	0.311 ± 0.011	0.674 ± 0.029	0.646 ± 0.032	0.345 ± 0.020	0.672 ± 0.031
	0.10	1.042 ± 0.034	0.184 ± 0.004	0.474 ± 0.018	0.341 ± 0.014	0.174 ± 0.004	0.472 ± 0.017
Traffic	0.01	1.033 ± 0.003	1.103 ± 0.007	0.990 ± 0.001	1.000 ± 0.001	1.000 ± 0.001	0.988 ± 0.002
	0.05	1.063 ± 0.002	0.447 ± 0.003	0.558 ± 0.009	0.482 ± 0.001	0.379 ± 0.001	0.437 ± 0.002
	0.10	1.064 ± 0.001	0.436 ± 0.006	0.271 ± 0.007	0.357 ± 0.001	0.219 ± 0.000	0.263 ± 0.003