Temporal Tensor Factorization: A Framework for Low-Rank Multilinear Time Series Forecasting

Jackson Cates*, Randy C. Hoover*, Kyle Caudle†, David Marchette‡, Karissa Schipke*

*Department of Electrical Engineering and Computer Science, South Dakota Mines, Rapid City, South Dakota, USA
†Department of Mathematics, South Dakota Mines, Rapid City, South Dakota, USA
‡Naval Surface Warfare Center, Dahlgren Division, Dahlgren, Virginia, USA

***Abstract*—In the era of big data, there are increasing demands to forecast incomplete, sparse, and nonstationary data. The current research aims to solve these problems for multilinear time series through a combination of tensor autoregression and low-rank tensor factorization. We propose an expansion of TMF to second-order data: temporal tensor factorization (TTF). The current research aims to interpolate missing values via low-rank tensor factorization, which produces a latent time series. We perform forecasting in the latent space with a transform-based tensor autoregression ($\mathcal{L}-$TAR) process. We present experimental results of the proposed method with other state-of-the-art methods on the benchmark datasets which include video imaging, dynamic networks, and energy usage.**

I. Introduction

Time series forecasting is a well-studied task in the area of machine learning. Many time series problems involve extrapolation: forecasting future observations based on present and past observations. Historically, numerous methods have been created to meet the challenges of forecasting, including exponential smoothing and neural networks [1]–[6]. A famous classical method is the Box-Jenkins Vector Autoregressive (VAR) model, where the forecast is made with a linear combination of recent historical values, also known as *lags* [4]. An important selection for this model is the "order", which dictates the number of lags to be used. For example, given a time series matrix $Y \in \mathbb{R}^{\ell \times T}$ with ℓ variables and T observations, the p^{th} order VAR model can be written as

$$\mathbf{y}_t = A_1 \mathbf{y}_{t-1} + \cdots + A_p \mathbf{y}_{t-p} + \boldsymbol{\epsilon}_t, \tag{1}$$

where $\mathbf{y}_t \in \mathbb{R}^{\ell}$ is the multivariate observation at time t, and also the t^{th} column of Y. The goal is to estimate the model parameters $\boldsymbol{\theta} = \{A_1, A_2, \ldots, A_p\}$, where $A_i \in \mathbb{R}^{\ell \times \ell}$ is a collection of learnable parameter matrices for $i = 1, 2, \ldots, p$. Additionally, it is generally assumed that $p \ll T$, the errors are independent and identically distributed, i.e., $\boldsymbol{\epsilon}_t \sim N(0, \Sigma)$, and the observations $\mathbf{y}_t$ are stationary.

In many applications, observations can be expressed in more complex natural structures, such as dynamic networks, video sequencing, correlated image sets, and geospatial data [7]–[11]. In these cases, observations are no longer vectors but viewed as a lateral slice of a tensor $\mathcal{Y}_t \in \mathbb{R}^{\ell \times 1 \times m}$. While possible to employ VAR methods in these applications by flattening the observations $\mathcal{Y}_t$ into a vector, performing this operation destroys the spatial representation of the observations, which can lead to poor forecasts. Therefore, many researchers focus on developing methods that perform forecasting while keeping the natural representation of the observations [12]–[14]. The authors in [14] focus on extending the VAR model into a tensor autoregressive problem. They achieve this by dividing the tensor autoregressive problem into multiple independent VAR problems via a tensor-transform. Because they utilize a tensor-transform, they refer to their method as the transform-based tensor autoregressive model, or $\mathcal{L}-$TAR. Given a multilinear time series tensor $\mathcal{Y} \in \mathbb{R}^{\ell \times T \times m}$, the p^{th} order $\mathcal{L}-$TAR model would be written as,

$$\mathcal{Y}_t = \mathcal{A}_i \bullet_{\mathcal{L}} \mathcal{Y}_{t-1} + \cdots + \mathcal{A}_p \bullet_{\mathcal{L}} \mathcal{Y}_{t-p} + \mathcal{E}_t \tag{2}$$

where $\mathcal{Y}_t$ is the multilinear observation at time t, and also the t^{th} lateral slice of $\mathcal{Y}$. $\bullet_{\mathcal{L}}$ indicates the tensor-product[1] as outlined in Section II. Similar to the VAR model, the goal is to estimate the model parameters $\Theta = \{\mathcal{A}_1, \ldots, \mathcal{A}_p\}$ where $\mathcal{A}_i$ is a collection of learnable parameter tensors for $i = 1, \ldots, p$. Additionally, it is assumed that $p \ll T$, the errors are independent and identically distributed, and the observations are stationary. Stationary can be addressed with lagged differencing, as outlined by the authors in [14], [15].

In many real-world problems, data is often messy and incomplete. Additionally, observations are often high-dimensional which may lead to the curse of dimensionality. To address these issues researchers focused on integrating the VAR model with low-rank matrix factorization techniques, called Temporal Matrix Factorization (TMF) [16]–[18]. The goal is to factorize the time series matrix $Y \in \mathbb{R}^{\ell \times T}$ into two latent factor matrices $W \in \mathbb{R}^{\ell \times r}, X \in \mathbb{R}^{r \times T}$ with rank r. The main idea is that X will contain the low-rank dynamics of the time series. Forecasts are made in the latent time series X, which is typically approximated by a VAR process. TMF has shown promise in forecasting time series with

The current research was supported in part by the Department of the Navy, Naval Engineering Education Consortium under Grant No. (N00174-19-1-0014) and the National Science Foundation under Grant No. (2007367). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Naval Engineering Education Consortium or the National Science Foundation.

[1]For brevity, we will drop $\mathcal{L}$ for the tensor-product $\bullet$ for the remainder of the paper.

missing observations, and there are many extensions of this framework [16]–[18]. In [17], the authors designed a temporal regularizer to apply temporal structure into the standard matrix factorization formulation, aptly named Temporal Regularized Matrix Factorization (TRMF). Additionally, the authors in [18] focus on forecasting nonstationary time series by differencing the latent time series, aptly named the Nonstationary Temporal Matrix Factorization (NoTMF). The general optimization problem of TMF can be summarized as

$$\min_{W,X,A} \frac{1}{2}||\mathcal{P}_\Omega(WX - Y)||_F^2 + \frac{\lambda}{2}\sum_{t=p+1}^{T}||\mathbf{x}_t - \sum_{i=1}^{p} A_i\mathbf{x}_{t-i}||_2^2 \quad (3)$$

where W, X are the rank r factor matrices and $A_i \in \mathbb{R}^{r\times r}$ are the coefficient matrices for the latent time series X. $\mathbf{x}_t$ is the latent time series observation at time t, and also the t^{th} column of X. λ is a weight parameter for regularization, and $\mathcal{P}_\Omega$ is a linear mapping on the observed index set Ω, where entries that are not in Ω are set to 0. This linear mapping $\mathcal{P}_\Omega$ ensures that missing entries are not included during gradient descent by setting them to 0.

Multilinear time series may have similar data quality issues to their multivariate counterparts. However as mentioned, applying multivariate methods to multilinear observations destroys their natural representation. In previous research tensor completion methods have been used to approximate the missing observations within a tensor. Typically these methods use the tensor nuclear-norm to promote low-rank structures [19], [20]. While these authors show promise in tensor completion, they do not focus on forecasting or incorporating temporal structure. The authors in [21] extend matrix factorization techniques into a tensor format, which is called low-rank tensor factorization. Given a data tensor $\mathcal{Y} \in \mathbb{R}^{\ell\times T\times m}$, the goal of low-rank tensor factorization is to factorize the tensor into two latent factor tensors $\mathcal{W} \in \mathbb{R}^{\ell\times r\times m}, \mathcal{X} \in \mathbb{R}^{r\times T\times m}$ with rank r such that $\mathcal{Y} \approx \mathcal{W} \bullet \mathcal{X}$.

In the following paper, we propose Temporal Tensor Factorization (TTF) to model multilinear, incomplete, and nonstationary time series. Building on the work in [14] and [21], we integrate the $\mathcal{L}-$TAR model with low-rank tensor factorization to perform tensor completion. TTF is a natural extension of TMF for multilinear observations. Similar to multivariate counterparts, we factorize the time series $\mathcal{Y}$ to capture the low-rank temporal dynamics by a latent time series $\mathcal{X}$. We perform forecasting by applying the $\mathcal{L}-$TAR process to the latent time series. The main contributions of this paper are:

1) Development of the TTF model which is obtained by integrating $\mathcal{L}-$TAR with low rank tensor factorization.
2) Presentation of a novel minimization framework to learn the parameter tensors $\mathcal{W}, \mathcal{X},$ and $\mathcal{A}_1, \dots, \mathcal{A}_p$.
3) Outline a differencing process that allows the TTF model to be applied to nonstationary observations.

The remainder of the paper is organized as follows: In Section II we provide some mathematical background for the tensor linear algebra we use throughout the paper. In Section III we present our TTF framework, and derive an effective and efficient training procedure to find the latent factor tensors $\mathcal{W}, \mathcal{X}$ and coefficient tensors $\mathcal{A}_1, \dots, \mathcal{A}_p$. We also introduce a rolling forecast mechanism and demonstrate how our framework can be extended for nonstationary observations. In Section IV we present some experimental results with the proposed method and compare/contrast with the state-of-the-art. Section V contains our conclusions and future directions.

II. Mathematical Preliminaries

To keep this paper self-contained, we will outline some of the mathematical foundations of the tensor notation and operations presented in [22]–[30].

A. Notation

In the context of this paper, the term *tensor* refers to a multi-dimensional array of numbers, sometimes called an *n-way* or *n-mode* array. If, for example, $\mathcal{A} \in \mathbb{R}^{\ell\times m\times n}$ then we say $\mathcal{A}$ is a third-order tensor where *order* is the number of ways or modes of the tensor. Thus, matrices and vectors are second-order and first-order tensors, respectively. It will be convenient to divide a tensor $\mathcal{A} \in \mathbb{R}^{\ell\times m\times n}$ into various slices and tubal elements. The i^{th} lateral slice will be denoted $\mathcal{A}_i$ whereas the j^{th} frontal slice will be denoted $\mathcal{A}^{(j)}$. The i^{th}, j^{th} frontal tube will be denoted $\mathbf{a}_{ij}$. The i^{th}, j^{th}, k^{th} element will be denoted a_{ijk}. In terms of *Python* slicing, this means $\mathcal{A}_i \equiv \mathcal{A}[:, i, :]$, $\mathcal{A}^{(j)} \equiv \mathcal{A}[:, :, j]$, $\mathbf{a}_{ij} \equiv \mathcal{A}[i, j, :]$, and $a_{ijk} \equiv \mathcal{A}[i, j, k]$. Additionally, we will need to define a "zero" tensor (tensor containing only zeros), which will be denoted as $\mathbf{0}_{\ell\times m\times n}$ where ℓ, m, n are the dimensions of the tensor.

B. Tensor Definitions and Operations

An important operation is the transform of a tensor via any invertible discrete transform $\mathcal{L} : \mathbb{C}^n \to \mathbb{C}^n$ [31]. As such, we have the following definition:

Definition 1. The $\mathcal{L}$-transform of the tensor $\mathcal{A}$, given by

$$\tilde{\mathcal{A}} = \mathcal{L}(\mathcal{A}) \in \mathbb{C}^{\ell\times m\times n},$$

is computed by applying the discrete transform[2] $\mathcal{L}$ of your choice along the frontal tubes $\mathbf{a}_{ij}$ of $\mathcal{A}$.

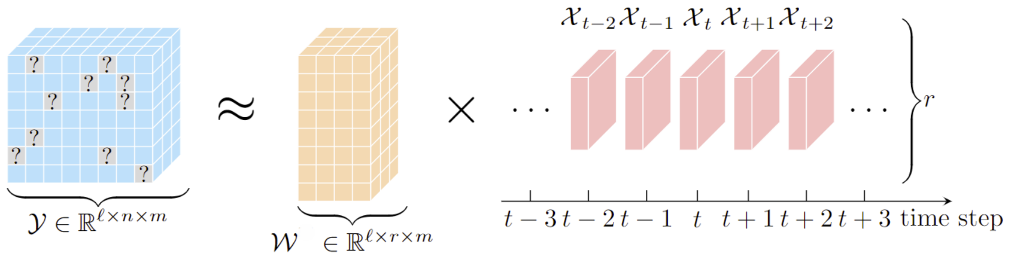
Using this formulation, we define the product of two third-order tensors which results in a third-order tensor. This operation, referred to as the tensor-product, is defined as follows:

Definition 2. Given two third-order tensors $\mathcal{A} \in \mathbb{C}^{\ell\times m\times n}$ and $\mathcal{B} \in \mathbb{C}^{m\times p\times n}$, the *face-wise* product $\mathcal{A}\Delta\mathcal{B}$ performs matrix multiplication for each of the frontal slices of $\mathcal{A}$ and $\mathcal{B}$. This results in a third-order tensor $\mathcal{C} \in \mathbb{C}^{\ell\times p\times n}$ and is defined as

$$\mathcal{C}^{(i)} = \mathcal{A}^{(i)} \cdot \mathcal{B}^{(i)},$$

for $i = 1, \dots, n$.

[2]Note: the current work focuses on the DWT and DCT. However, the DFT framework also applies here.



$\mathcal{X}_{t-2}\,\mathcal{X}_{t-1}\,\mathcal{X}_t\,\mathcal{X}_{t+1}\,\mathcal{X}_{t+2}$
?
$\approx$
$\times$
r
$\mathcal{Y} \in \mathbb{R}^{\ell\times n\times m}$
$\mathcal{W} \in \mathbb{R}^{\ell\times r\times m}$
$t-3\; t-2\; t-1\; t\; t+1\; t+2\; t+3$ time step

Fig. 1: Illustration of the TTF framework. The original multilinear time series $\mathcal{Y}$ contains some missing entries, and Ω would be the set of all known entries (solid blue). $\mathcal{Y}$ is factored into a latent spatial tensor $\mathcal{W}$ and a latent temporal tensor $\mathcal{X}$. This figure was adapted from its multivariate counterpart in [18].

Definition 3. Given two third-order tensors $\mathcal{A} \in \mathbb{C}^{\ell\times m\times n}$ and $\mathcal{B} \in \mathbb{C}^{m\times p\times n}$, the *tensor-product* $\bullet$ between $\mathcal{A}$ and $\mathcal{B}$ is defined as:

$$\tilde{\mathcal{A}} = \mathcal{L}(\mathcal{A})$$
$$\tilde{\mathcal{B}} = \mathcal{L}(\mathcal{B})$$
$$\mathcal{A} \bullet \mathcal{B} = \mathcal{L}^{-1}(\tilde{\mathcal{A}} \Delta \tilde{\mathcal{B}})$$

Definition 4. The tensor norm $\|\mathcal{A}\|_F \in \mathbb{R}$ for $\mathcal{A} \in \mathbb{R}^{\ell\times m\times n}$ is given by

$$\|\mathcal{A}\|_F = \sqrt{\sum_{i=1}^{\ell}\sum_{j=1}^{m}\sum_{k=1}^{n} a_{ijk}^2}.$$

For a given tensor $\mathcal{A} \in \mathbb{R}^{\ell\times m\times n}$ there are two definitions to describe its rank: *multi-rank* and *tubular rank* [24]. Tubular rank is defined by the *tensor singular value decomposition* (t-SVD).

Definition 5. The identity tensor $\mathcal{I} \in \mathbb{R}^{\ell\times \ell\times n}$ is the tensor whose first frontal slice is the $\ell \times \ell$ identity matrix and the other frontal slices are zeros in the transform domain.

Definition 6. If $\mathcal{A} \in \mathbb{R}^{\ell\times m\times n}$, then the tensor transpose $\mathcal{A}^T \in \mathbb{R}^{m\times \ell\times n}$ is computed by transposing the frontal slices of $\mathcal{A}$ in the transform domain.

Definition 7. A tensor $\mathcal{A} \in \mathbb{R}^{\ell\times m\times n}$ is *orthogonal* if $\mathcal{A} \bullet \mathcal{A}^T = \mathcal{A}^T \bullet \mathcal{A} = \mathcal{I}$.

Definition 8. (t-SVD [23]) For $\mathcal{A} \in \mathbb{R}^{\ell\times m\times n}$, then there exists tensors $\mathcal{U}, \mathcal{S}, \mathcal{V}$ such that

$$\mathcal{A} = \mathcal{U} \bullet \mathcal{S} \bullet \mathcal{V}^T$$

where $\mathcal{U} \in \mathbb{R}^{\ell\times \ell\times n}$ is an orthogonal tensor of left-singular matrices (analogous to left-singular vectors in second-order tensors), $\mathcal{V} \in \mathbb{R}^{m\times m\times n}$ is an orthogonal tensor of right-singular matrices (analogous to right-singular vectors in second-order tensors), and $\mathcal{S} \in \mathbb{R}^{\ell\times m\times n}$ is a f-diagonal tensor (analogous to singular values in second-order tensors). A f-diagonal tensor is where the frontal tubes for the i^{th} row and j^{th} column are nonzero for $i = j$ and zero otherwise.

Definition 9. For $\mathcal{A} \in \mathbb{R}^{\ell\times m\times n}$, its *tubular rank* is the number of nonzero singular tubes of $\mathcal{S}$. Its *multi-rank* is a vector defined as $r = [\text{rank}(\tilde{\mathcal{A}}^{(1)}), \ldots, \text{rank}(\tilde{\mathcal{A}}^{(n)})]$ [23], [24].

III. Proposed Approach to Low-Rank Multilinear Time-Series Forecasting

In this section, we discuss the details of our proposed method TTF. Namely, we give an overview of our framework, derive the alternating minimization framework, and introduce our forecasting mechanism.

A. Formulation of Temporal Tensor Factorization

Let $\mathcal{Y} \in \mathbb{R}^{\ell\times T\times m}$ be a partially observed multilinear time series tensor with T observations for $t = 1, \ldots, T$. Also, let $\mathcal{Y}_t$ be the t^{th} lateral slice of $\mathcal{Y}$, and the given observation at time t. For this problem, $\mathcal{Y}$ contains all the observations $\mathcal{Y}_t$ but may have some missing data. Therefore, let Ω be the set of observed entries $\{a_{ijk} \mid (i,j,k) \in \Omega\}$. This is illustrated in Fig. 1, where $\mathcal{Y}$ contains missing data. The solid blue cells indicates the observed entries Ω. Our goal is to factor the observation $\mathcal{Y}$ into a spatial latent tensor $\mathcal{W} \in \mathbb{R}^{\ell\times r\times m}$ and a temporal latent tensor $\mathcal{X} \in \mathbb{R}^{r\times T\times m}$ such that $r < \min(\ell, T)$ and

$$\mathcal{Y} \approx \mathcal{W} \bullet \mathcal{X}. \tag{4}$$

This step is necessary in order to capture the spatial and temporal correlations of the original time series $\mathcal{Y}$. In addition, this operation allows us to perform tensor completion for the missing entries in $\mathcal{Y}$.

After factoring the time series $\mathcal{Y}$, the temporal latent tensor $\mathcal{X}$ contains the low-rank temporal dynamics of $\mathcal{Y}$. In fact, we define the lateral slices $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_t$ as the latent time series of $\mathcal{Y}$. Additionally, $\mathcal{X}$ contains no missing entries, so a $\mathcal{L}-$TAR process can be applied here. We apply a p^{th} order $\mathcal{L}-$TAR process as

$$\mathcal{X}_t \approx \sum_{i=1}^{p} \mathcal{A}_i \bullet \mathcal{X}_{t-i} \tag{5}$$

where $\mathcal{A}_i \in \mathbb{R}^{r\times r\times m}$ is the coefficient tensor for lag i for $i = 1, \ldots, p$. Similar to the definition in Eqn. (2), we assume that the observations $\mathcal{X}_t$ are stationary, i.i.d., and $p \ll T$. After a forecast is obtained, i.e. $\hat{\mathcal{X}} \in \mathbb{R}^{r\times w\times m}$ for a forecast of w

steps, we map the latent forecast $\hat{\mathcal{X}}$ back into the original feature space by the latent spatial tensor $\mathcal{W}$ as

$$\hat{\mathcal{Y}} \approx \mathcal{W} \bullet \hat{\mathcal{X}} \in \mathbb{R}^{\ell \times w \times m}. \tag{6}$$

$\hat{\mathcal{Y}}$ will now contain the forecast of our original time series in its original feature space. Additionally, this forecast is made for all entries in $\hat{\mathcal{Y}}$ due to our tensor factorization framework. A visual illustration of this process is shown in Fig. 1.

B. Optimization

In order to effectively learn the latent factors $\mathcal{W}, \mathcal{X}$ and the coefficient tensors $\mathcal{A}_i$ for $i = 1, \dots, p$, we define the optimization problem for TTF as

$$\min_{\mathcal{W},\mathcal{X},\mathcal{A}} \frac{1}{2}\|\mathcal{P}_\Omega(\mathcal{W}\bullet\mathcal{X}-\mathcal{Y})\|_F^2+\frac{\lambda}{2}\sum_{t=p+1}^{T}\|\mathcal{X}_t-\sum_{i=1}^{p}\mathcal{A}_i\bullet\mathcal{X}_{t-i}\|_F^2. \tag{7}$$

Similar to the multivariate counterparts, this objective function balances the low-rank tensor factorization loss and the temporal loss with a regularization parameter λ. This regularization is crucial to maintain important temporal and spatial relationships [12]–[14]. $\mathcal{P}_\Omega$ is a linear mapping on the observed index set Ω such that

$$\mathcal{P}_\Omega(\mathcal{A}) = \begin{cases} a_{ijk} & \text{if } (i,j,k) \in \Omega \\ 0 & \text{if } (i,j,k) \notin \Omega. \end{cases} \tag{8}$$

Because $\mathcal{Y}$ may have missing entries, a $\mathcal{L}$-transform cannot be directly computed. To solve this problem, in a similar fashion to the standard tensor factorization framework in [21], we introduce an auxiliary variable $\mathcal{C}$ to solve the problem (7) more conveniently:

$$\min_{\mathcal{W},\mathcal{X},\mathcal{A},\mathcal{C}} \frac{1}{2}\|\mathcal{W}\bullet\mathcal{X}-\mathcal{C}\|_F^2+\frac{\lambda}{2}\sum_{t=p+1}^{T}\|\mathcal{X}_t-\sum_{i=1}^{p}\mathcal{A}_i\bullet\mathcal{X}_{t-i}\|_F^2, \quad \text{s.t. } \mathcal{P}_\Omega(\mathcal{C}-\mathcal{Y}) = \mathbf{0}_{\ell\times T\times m}. \tag{9}$$

Using the $\mathcal{L}$-transform, we can transform TTF to a set of m independent TMF problems [12], [14], [21]. Therefore, the problem (9) is equivalent to

$$\begin{aligned} \min_{\tilde{\mathcal{W}}^{(j)},\tilde{\mathcal{X}}^{(j)},\tilde{\mathcal{A}}^{(j)},\tilde{\mathcal{C}}^{(j)}} & \frac{1}{m}\sum_{j=1}^{m}\frac{1}{2}\|\tilde{\mathcal{W}}^{(j)}\tilde{\mathcal{X}}^{(j)}-\tilde{\mathcal{C}}^{(j)}\|_F^2 \\ & +\frac{\lambda}{2}\sum_{t=p+1}^{T}\|\tilde{\mathcal{X}}_t^{(j)}-\sum_{i=1}^{p}\tilde{\mathcal{A}}_i^{(j)}\tilde{\mathcal{X}}_{t-i}^{(j)}\|_F^2, \\ & \text{s.t. } \mathcal{P}_\Omega(\mathcal{C}-\mathcal{Y}) = \mathbf{0}_{\ell\times T\times m}, \end{aligned} \tag{10}$$

for $j = 1, \dots, m$. To simply the problem further, we define a sequence of temporal operators that allow us to represent our latent time series as the full tensor $\mathcal{X}$ as

$$\Psi_k = \begin{bmatrix} \mathbf{0}_{(T-p)\times(p-k)} & I_{T-p} & \mathbf{0}_{(T-p)\times k} \end{bmatrix} \in \mathbb{R}^{(T-p)\times T} \tag{11}$$

for $k = 0, 1, \dots, p$ [18]. This allows us to further rewrite the problem (10) to

$$\begin{aligned} \min_{\tilde{\mathcal{W}}^{(j)},\tilde{\mathcal{X}}^{(j)},\tilde{\mathcal{A}}^{(j)},\tilde{\mathcal{C}}^{(j)}} & \frac{1}{m}\sum_{j=1}^{m}\frac{1}{2}\|\tilde{\mathcal{W}}^{(j)}\tilde{\mathcal{X}}^{(j)}-\tilde{\mathcal{C}}^{(j)}\|_F^2 \\ & +\frac{\lambda}{2}\|\tilde{\mathcal{X}}^{(j)}\Psi_0^T-\tilde{\mathcal{A}}^{(j)}(I_d\otimes\tilde{\mathcal{X}}^{(j)})\Psi^T)\|_F^2, \\ & \text{s.t. } \mathcal{P}_\Omega(\mathcal{C}-\mathcal{Y}) = \mathbf{0}_{\ell\times T\times m}, \end{aligned} \tag{12}$$

where $\otimes$ denotes the Kronecker product, $\tilde{\mathcal{A}}^{(j)} = [\tilde{\mathcal{A}}_1^{(j)} \cdots \tilde{\mathcal{A}}_p^{(j)}] \in \mathbb{R}^{r\times rp}$, and $\Psi = [\Psi_1 \cdots \Psi_p] \in \mathbb{R}^{(T-p)\times pT}$. Let f be the optimization problem outlined in Eqn. (12). f can be solved with an alternating minimization framework for the parameters $\mathcal{W}, \mathcal{X}, \mathcal{A},$ and, $\mathcal{C}$. Additionally, since each TMF problem is independent, each problem can be solved efficiently in parallel.

1) Updating the Auxiliary Variable: To update $\mathcal{C}$, we can perform the update as follows:

$$\mathcal{C} = \mathcal{W}\bullet\mathcal{X} + \mathcal{P}_\Omega(\mathcal{Y}-\mathcal{W}\bullet\mathcal{X}). \tag{13}$$

2) Updating the Latent Spatial Tensor: To update $\mathcal{W}$, we can derive the update as follows:

$$\begin{aligned} \frac{\partial f}{\partial\tilde{\mathcal{W}}^{(j)}} &= (\tilde{\mathcal{W}}^{(j)}\tilde{\mathcal{X}}^{(j)}-\tilde{\mathcal{C}}^{(j)})(\tilde{\mathcal{X}}^{(j)})^* = \mathbf{0}_{\ell\times r}, \\ \tilde{\mathcal{W}}^{(j)}\tilde{\mathcal{X}}^{(j)}(\tilde{\mathcal{X}}^{(j)})^* &- \tilde{\mathcal{C}}^{(j)}(\tilde{\mathcal{X}}^{(j)})^* = \mathbf{0}_{\ell\times r}, \\ \tilde{\mathcal{W}}^{(j)} &= \tilde{\mathcal{C}}^{(j)}(\tilde{\mathcal{X}}^{(j)})^*(\tilde{\mathcal{X}}^{(j)}(\tilde{\mathcal{X}}^{(j)})^*)^\dagger. \end{aligned} \tag{14}$$

3) Updating the Latent Time Series Tensor: To update $\mathcal{X}$, we can derive the update as follows:

$$\begin{aligned} \frac{\partial f}{\partial\tilde{\mathcal{X}}^{(j)}} &= (\tilde{\mathcal{W}}^{(j)})^*(\tilde{\mathcal{W}}^{(j)}\tilde{\mathcal{X}}^{(j)}-\tilde{\mathcal{C}}^{(j)}) \\ &+\lambda\sum_{i=1}^{p}(\tilde{\mathcal{A}}_i^{(j)})^*(\sum_{k=1}^{p}\tilde{\mathcal{A}}_k^{(j)}\tilde{\mathcal{X}}^{(j)}\Psi_k^T)\Psi_i = \mathbf{0}_{r\times T}. \end{aligned} \tag{15}$$

Eqn. (15) is a generalized Sylvester equation [18], [32]. This equation can be converted to a system of linear equations to solve for $\tilde{\mathcal{X}}^{(j)}$, however, that would be computationally expensive as solving Eqn. (15) would take $O(T^3)$. The authors in [18] overcomes this challenge by approximating the solution $\tilde{\mathcal{X}}^{(j)}$ with a conjugate gradient method. They demonstrated that the conjugate gradient is an effective method to approximate their latent time series matrix while requiring a small number of iterations. More details on the specific implementation can be found in [18].

4) Updating the Coefficient Tensors: To update $\mathcal{A}$, we can derive the update as follows:

$$\begin{aligned} \frac{\partial f}{\partial\tilde{\mathcal{A}}^{(j)}} &= -\lambda(\tilde{\mathcal{X}}^{(j)}\Psi_0^T-\tilde{\mathcal{A}}^{(j)}(I_p\otimes\tilde{\mathcal{X}^{(j)}})\Psi^T)\Psi(I_p\otimes\tilde{\mathcal{X}}^{(j)})^T \\ &= \mathbf{0}_{r\times rp}, \\ \tilde{\mathcal{A}}^{(j)} &= \tilde{\mathcal{X}}^{(j)}\Psi_0^T((I_p\otimes\tilde{\mathcal{X}}^{(j)})\Psi^T)^\dagger. \end{aligned} \tag{16}$$

Algorithm 1 TTF

```
Input: Multilinear time series $\mathcal{Y} \in \mathbb{R}^{\ell \times T \times m}$, $p$ lags and rank $r$.
Initialize $\mathcal{X}, \mathcal{Y}$, and $\mathcal{A}$ randomly
while not converged do
    Update $\mathcal{C}$ by (13) and compute $\tilde{\mathcal{C}} = \mathcal{L}(\mathcal{C})$
    for j = 1,...,m do
        Update $\tilde{\mathcal{W}}^{(j)}$ by (14)
        Update $\tilde{\mathcal{X}}^{(j)}$ by (15) using conjugate gradient descent
        Update $\tilde{\mathcal{A}}^{(j)}$ by (16)
    end for
    epoch = epoch + 1
end while
return $\mathcal{W}$, $\mathcal{X}$, and $\mathcal{A}$
```

Algorithm 2 TTF Rolling Forecast for Online Learning

```
Input: Multilinear time series $\mathcal{Y} \in \mathbb{R}^{\ell \times T \times m}$, $\mathcal{W} \in \mathbb{R}^{\ell \times r \times m}$, $\mathcal{X} \in \mathbb{R}^{r \times T \times m}$, and $\mathcal{A} \in \mathbb{R}^{r \times r \times m}$ for $i = 1, \ldots, p$.
t = T+1
while True do
    Retrieve a new observation $\mathcal{Y}_t$ and append to $\mathcal{Y}$
    Update $\mathcal{C}$ by (13) and compute $\tilde{\mathcal{C}} = \mathcal{L}(\mathcal{C})$
    for j = 1,...,m do
        Update $\tilde{\mathcal{X}}^{(j)}$ by (15) using conjugate gradient descent
        Update $\tilde{\mathcal{A}}^{(j)}$ by (16)
        Generate a forecast $\hat{\mathcal{Y}}_t$ with Eqns. (4) and (5).
    end for
    $t = t + 1$
end while
return $\mathcal{W}$, $\mathcal{X}$, and $\mathcal{A}$
```

C. Training Procedure

We formulate our training procedure based on the update derivations in the previous section. Eqns. (14), (15), and (16) are all in the transform domain, and as such the parameters $\mathcal{W}, \mathcal{X}$, and $\mathcal{A}$ can be kept in the transform domain during training. Eqn. (13) requires $\mathcal{W} \bullet \mathcal{X}$ to be in the spatial domain to compute $\mathcal{P}_\Omega(\mathcal{Y} - \mathcal{W} \bullet \mathcal{X})$. Therefore, the most efficient method is to compute $\tilde{\mathcal{G}}^{(j)} = \tilde{\mathcal{W}}^{(j)} \tilde{\mathcal{X}}^{(j)}$ for $j = 1, \ldots, m$ and then compute $\mathcal{G} = \mathcal{L}^{-1}(\tilde{\mathcal{G}}) = \mathcal{W} \bullet \mathcal{X}$. After $\mathcal{C}$ is computed, we directly compute $\tilde{\mathcal{C}} = \mathcal{L}(\mathcal{C})$ and utilize $\tilde{\mathcal{C}}$ to update Eqns. (14), (15), and (16), which can be done in parallel. Nevertheless, the overall training procedure is quite efficient and has a runtime of $O(T)$. The overall training procedure is shown in Algorithm 1.

D. Handling Nonstationary Observations

The original formation of $\mathcal{L}-$TAR requires that the observations $\mathcal{X}_t$ are stationary. However, in many applications and domains, $\mathcal{Y}$ may not be necessarily stationary, as the observations can contain seasonality or trends. It is generally assumed that the latent time series $\mathcal{X}$ will carry the same temporal characteristics as $\mathcal{Y}$, and the stationarity condition for $\mathcal{L}-$TAR may be violated [16], [18]. As such, we extend our formulation of TTF to handle nonstationary observations. We use the differencing operations that were developed in [14].

The temporal operator Ψ defined in Eqn. 11 already handles the p-lag temporal shifts in the $\mathcal{L}-$TAR process. To handle nonstationary observations, we utilize the temporal operators defined in [18], which will apply a d^{th} lagged difference to the latent time series $\mathcal{X}$. The temporal operator Ψ with a d^{th} lagged difference is formulated as

$$\begin{aligned}\Psi_k = &\left[\mathbf{0}_{(T-p-d)\times(p-k)} \quad -I_{T-p-d} \quad \mathbf{0}_{(T-p-d)\times(k+d)}\right] \\ &+ \left[\mathbf{0}_{(T-p-d)\times(p+d-k)} \quad I_{T-d-p} \quad \mathbf{0}_{(T-p-d)\times k}\right] \\ &\in \mathbb{R}^{(T-p-d\times T)}.\end{aligned} \tag{17}$$

To apply d^{th} lagged difference to our original formation of TTF, we simply apply the new temporal operator Ψ outlined in Eqn. (17) to the update Eqns. (15) and (16).

E. Rolling Forecast Mechanism

Unlike the work in [14], a rolling forecast mechanism is nontrivial due to the latent time series $\mathcal{X}$. That is, if $\mathcal{X}_t$ is learned for timesteps $t = 1, \ldots, T$, $\mathcal{X}_t$ is unknown for $t > T$. As such, multivariate methods outlined in [33] and [18] employ a dictionary learning scheme to learn the latent time series $\mathcal{X}_t$ for $t > T$. It is assumed that the spatial information of the time series will remain static, while the temporal dynamics may change [33]. Therefore, this dictionary learning scheme will keep the latent spatial tensor $\mathcal{W}$ static while updating the temporal parameters $\mathcal{X}$ and $\mathcal{A}$. This results in a similar procedure to Algorithm 1, without the update for $\mathcal{W}$ in Eqn. 14.

IV. Experimental Results

In this section, we compare our proposed TTF model with baseline models in the literature. We illustrate how our framework can learn the underlying spatial and temporal relationships with partially observed data.

A. Baseline Models and Experimental Methology

We compare our approach against the current state-of-the-art TMF models. Namely, the original Temporal Matrix Factorization TMF, the temporal regularizer version TRMF, and the nonstationary version NoTMF [16]–[18]. In addition as a baseline, we will also compare against $\mathcal{L}-$TAR [14]. An overview of all baseline models:

- **TMF**: The standard Temporal Matrix Factorization. The observations $\mathcal{Y}_t$ are flattened into vectors and treated as a multivariate problem.
- **TRMF**: Temporal Matrix Factorization with temporal regularization. The observations $\mathcal{Y}_t$ are flattened into vectors and treated as a multivariate problem.

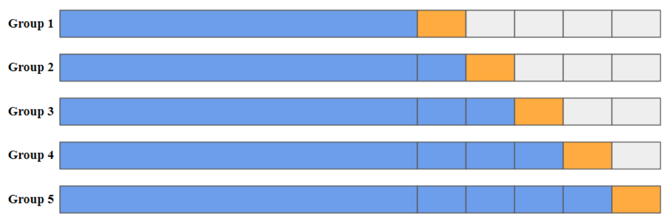


Fig. 2: Illustration of cross-validation for time series. The time series is split into a train/test set, with the test set split further into 5 groups. For each group, a model is trained using the sections colored in blue. The model is then validated with the observations colored in orange. The gray-colored observations are omitted from testing. At the end, the error from each group (orange) is averaged to give a cross-validated metric.

- **NoTMF**: Temporal Matrix Factorization designed to handle nonstationary observations by applying lagged differencing in the latent time series. The observations $\mathcal{Y}_t$ are flattened into vectors and treated as a multivariate problem.
- $\mathcal{L}$**−TAR**: A transform-based tensor autoregressive model. Originally not designed to handle missing observations, so entries that are not in the index set Ω are set to 0 before fitting.

We evaluate all baseline models by cross-validation. We split each dataset into a train and test set. The test set is divided into 5 equal groups. For each group, the training set is all observations that accrued prior to the first observation that forms the test set [34]. All of the group's error is averaged to give a cross-validated metric. This process is illustrated graphically in Fig. 2. Similar to other multilinear models in the literature, we utilize relative error as a performance metric

$$\frac{\|\mathcal{Y}_t - \hat{\mathcal{Y}}_t\|_F}{\|\mathcal{Y}_t\|_F}, \tag{18}$$

where $\mathcal{Y}_t$ is the observed observation at time t, and $\hat{\mathcal{Y}}_t$ is the forecast [12]–[14]. In addition to cross-validation, we compute two types of forecast: 1) A single-step forecast, which uses the rolling forecast mechanism as outlined in Section III-E, and 2) a multi-step forecast, where all observations in a given group are forecast at once. This is to evaluate each model's ability to perform both short-term and long-term forecasts. The single-step forecast will be much more accurate due to the much smaller forecasting horizon.

B. Datasets

We utilize a combination of baseline datasets from [14] and [18]. These datasets are designed for multilinear forecasting, where the observations have a natural representation as a tensor. The datasets include geospatial data, video imaging, and dynamic graphs. Namely: (a) the JERICHO-E-usage dataset of energy usage across different regions and energy sectors in Germany [7]; (b) a grid of sea-surface temperatures [8]; (c) an adapted version of the MovingMNIST dataset that contains a grid of bouncing MNIST digits [9], [10]; (d) a dynamic traffic network of the taxicab trips in the New York Manhattan area [11]. To evaluate how well the methods can handle missing observations, 20% of entries were randomly removed from each dataset.

TABLE I: Hyperparameters for Each Method and Dataset

Method	Dataset	p	r	d	λ	ρ
TTF	JERICHO-E	1	5	168	1	-
TTF	SST	1	3	24	1	-
TTF	MovingMNIST	2	5	0	0.5	-
TTF	NYC Trip Record	2	20	7	2	-
TMF	JERICHO-E	6	10	-	1	1
TMF	SST	6	5	-	0.5	1
TMF	MovingMNIST	3	15	-	0.1	0.5
TMF	NYC Trip Record	6	20	-	1	1.5
TRMF	JERICHO-E	6	15	-	1	1
TRMF	SST	6	10	-	0.5	1
TRMF	MovingMNIST	3	15	-	0.1	0.5
TRMF	NYC Trip Record	6	20	-	1	1
NoTMF	JERICHO-E	1	10	168	2	1
NoTMF	SST	2	5	24	1	1
NoTMF	MovingMNIST	1	10	1	1	2
NoTMF	NYC Trip Record	3	20	7	1	0.5
L-TAR	JERICHO-E	2	-	24	-	-
L-TAR	SST	1	-	24	-	-
L-TAR	MovingMNIST	2	-	0	-	-
L-TAR	NYC Trip Record	6	-	0	-	-

*1) **JERICHO-E-usage**:* The JERICHO-E-usage dataset contains energy consumption usage across 38 regions of Germany [7]. The time series breaks down into 4 categories: residential, industrial, commerce, and mobility. This dataset records the hourly consumption for the whole year of 2019 (8,760 observations), giving a multilinear time series $\mathcal{Y} \in \mathbb{R}^{38\times8760\times4}$. The last 2 months of data (1,488 observations) are used for testing and the remaining data (7,272 observations) are used for training. Additionally, the dataset has obvious 24-hour seasonality.

*2) **SST**:* The SST dataset is a 5×6 grid of sea-surface temperatures, where the observations were recorded every hour [8]. This dataset records hourly temperature readings for $T = 2000$ observations, giving a multilinear time series $\mathcal{Y} \in \mathbb{R}^{5\times2000\times6}$. The last 2 weeks of data (336 observations) are used for testing and the remaining data (1,664 observations) are used for training. Additionally, the dataset has obvious 24-hour seasonality.

*3) **MovingMNIST**:* The MovingMNIST dataset is a video of bouncing MNIST digits within a 50×50 grid [9], [10]. We generated 2000 frames of the digits 1, 3, and 5 bouncing within a 50×50 grid, giving a multilinear time series $\mathcal{Y} \in \mathbb{R}^{50\times2000\times50}$. The last 400 frames are used for testing and the remaining data (1,600 observations) are used for training.

*4) **NYC Trip Record**:* The NYC Trip Record dataset contains taxicab pickup and dropoff locations for the New York Manhattan area [11]. The original dataset splits the area into 263 pickup and dropoff zones. These zones can be represented as nodes in a dynamic graph. While the dataset spans between

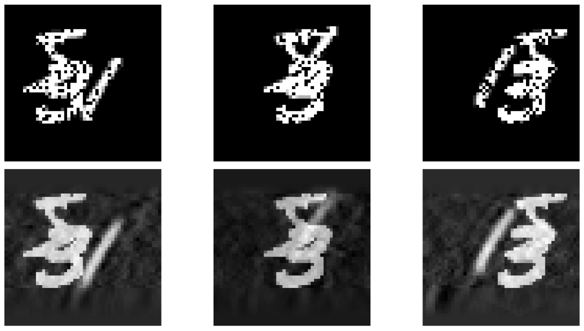


Fig. 3: Qualitative evaluation of our framework's ability to capture underlying temporal/spatial patterns in the MovingMNIST dataset. **Top row:** A sampling of 3 multilinear observations for $t = 1600, 1610, 1620$, which is part of the test set. Note the 20% missing entries (pixels) are set to 0 and colored as black. **Bottom row:** A single-step forecast for the above samples. We generate a forecast from the TTF model outlined in Table I.

January 2015 to the present (June 2024), we only use a subset between January 2015 to December 2019 as COVID-19 caused a dramatic decrease in the amount of trips. With daily taxicab trips, this gives a multilinear time series $\mathcal{Y} \in \mathbb{R}^{263\times1825\times263}$. We use the last year (365 observations) for testing and the remaining years (1,460 observations) are used for training.

C. Experimental Results

In this section, we report the experimental results of the forecasting performance for all baseline models. To optimize the performance of each model, we conducted hyperparameter tuning using a grid search approach. Grid search was chosen because it systematically explores a specified range of hyperparameters, ensuring that each model is thoroughly evaluated. The parameters selected for the grid search were based on domain knowledge and previous experiments outlined in [14], [15], [18]. For factorization methods (TMF, TRMF, NoTMF, & TTF), these parameters include the rank r and various regularization parameters λ. We evaluate the models for ranks $r = 3, 5, 10, 15, 20$ and $\lambda = 0.1, 0.5, 1, 1.5, 2$. For all methods, we also consider the order p and lagged differencing d. In the grid search, we evaluate the models for $p = 1, 2, \dots, 6$ and $d = 0, 1$. The JERICHO-E-usage, SST, and NYC Taxicab datasets may contain some seasonality, so we include lagged differencing for various seasonality patterns. For the JERICHO-E-usage and SST datasets, we include both weekly and daily seasonality differencing ($s = 24, 168$), and for the NYC Taxi Cab dataset, we include weekly and yearly seasonality differencing ($d = 7, 365$). All the final parameters are listed in Table I.

Fig. 3 demonstrates our framework's ability to learn underlying temporal/spatial patterns for the MovingMNIST dataset. As illustrated, our method can learn the underlying correlations from partially observed data. Additionally, this illustrates TTF's predictive capability. While not perfect, post-processing techniques can easily clean the image sequences. Since TTF performs dimensionality reduction to represent the partially observed data as a latent time series, the forecast appears smoother and more generalized, which effectively captures the essential features of the data while filtering out noise.

TABLE II: Experimental results for all baseline models. Errors are reported by average relative error outlined in Eqn. (18)

Dataset	TTF	TMF	TRMF	NoTMF	$\mathcal{L}$-TAR
Single-step Forecast					
JERICHO-E	**0.101**	0.654	0.610	0.424	0.357
SST	**0.128**	0.253	0.206	0.187	0.144
MovingMNIST	**0.107**	0.221	0.157	0.198	0.216
NYC Trip Record	**0.221**	0.386	0.314	0.289	0.247
Multi-step Forecast					
JERICHO-E	**0.286**	0.945	0.913	0.549	0.684
SST	**0.194**	0.291	0.292	0.207	0.291
MovingMNIST	**0.170**	0.376	0.360	0.312	0.235
NYC Trip Record	0.382	0.519	0.428	0.323	**0.319**

Table II provides the summary results for each of the forecasting methods. These results highlight the robustness and effectiveness of the TTF framework in both single-step and multi-step forecasting across all datasets. This also demonstrates that our framework performs well within a wide range of domains and applications. Our framework outperforms the baseline methods in almost all datasets, except for the multi-step forecasting in the NYC Trip Record dataset. Our framework outperforms the multivariate counterparts because we are able to represent observations $\mathcal{Y}_t$ as their natural tensor representation. We also outperform the $\mathcal{L}-$TAR model due to its inability to address missing entries. Additional, our method performs well when the time series contains seasonality, as shown in the JERICHO-E-usage, SST, and NYC Trip Record datasets.

V. Conclusions and Future Directions

We propose a novel temporal tensor factorization framework to forecast a partially observed multilinear time series. We factor the time series into two latent tensors in order to learn the underlying spatial/temporal relationships. This enables us to 1) maintain the natural tensor representation of the time series and 2) provide a latent time series where a forecast can be made by an $\mathcal{L}-$TAR process. We derive an alternating minimization framework that effectively and efficiently learns the latent factor and coefficient tensors in $O(T)$. We also demonstrate how we enable our framework to handle nonstationary observations and a rolling forecast mechanism. Compared to multivariate factorization techniques and $\mathcal{L}-$TAR, our framework outperforms the state of the art in geospatial datasets, video sequences, and dynamic graphs.

Future work will include developing a rank estimation scheme to learn the optimal rank r for the factor tensors $\mathcal{W}, \mathcal{X}$, as the rank in Section IV was picked via grid search. Future work will also investigate the advantages of running the update equations (14), (15), (16) in parallel. Additionally, we would like to explore how low-rank tensor factorization can extend other multivariate frameworks, such as non-negative matrix factorization which is commonly used for clustering [35]–[37].

References

[1] G. Box., "Understanding exponential smoothin-a simple way to forecast sales and inventory," *Quality Engineering*, vol. 4, no. 3, pp. 561–566, 1991.

[2] R. Brown, *Statistical Forecasting for Inventory Control*. New York: McGraw-Hill, 1959.

[3] ——, *Smoothing, Forecasting, and Prediction*. NJ: Pretice Hall, Englewood Cliffs, 1963.

[4] Box, P.J., GM, and R., *Time Series Analysis: Forecasting & Control*, 2008.

[5] S. Haykin, *Neural Networks and Learning Machines (3rd ed.)*. New York: Pearson, 2009.

[6] T. Hill, L. Marquez, M. O'Connor, and W. Remus, "Artificial neural network models for forecasting and decision making," *International Journal of Forecasting*, vol. 10, pp. 5–15, 1994.

[7] J. Priesmann, L. Nolting, C. Kockel, and A. Praktiknjo, "Time series of useful energy consumption patterns for energy system modeling," *Scientific Data*, vol. 8, no. 1, pp. 1–12, 2021.

[8] NOAA/Pacific Marine Environmental Laboratory, "Tropical atmosphere ocean project," http://www.pmel.noaa.gov/tao/data_deliv/deliv.html, accessed: May 23, 2013.

[9] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using LSTMs," 2015.

[10] J. Cates, R. C. Hoover, K. Caudle, D. Marchette, and C. Ozdemir, "Anomaly detection from multilinear observations via time-series analysis and 3dtpca," in *2022 21st IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2022, pp. 677–680.

[11] "Nyc trip record data," 2022. [Online]. Available: https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

[12] W. Lu, X.-Y. Liu, Q. Wu, Y. Sun, and A. Elwalid, "Transform-Based Multilinear Dynamical System for Tensor Time Series Analysis," in *Neural Information Processing (NIPS) Workshop on Spatiotemporal Data*, 2018.

[13] M. Rogers, L. Li, and S. J. Russell, "Multilinear dynamical systems for tensor time series," in *in Neural Information Processing Systems (NIPS)*, 2013, pp. 2634–2642.

[14] J. Cates, R. C. Hoover, and K. Caudle, "Transform-based tensor auto regression for multilinear time series forecasting," in *2021 IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2021.

[15] J. Cates, R. C. Hoover, K. Caudle, C. Ozdemir, K. Braman, and D. Machette, "Forecasting multilinear data via transform-based tensor autoregression," 2022.

[16] Y.-Y. Lo, W. Liao, C.-S. Chang, and Y.-C. Lee, "Temporal matrix factorization for tracking concept drift in individual user preferences," *IEEE Transactions on Computational Social Systems*, vol. 5, no. 1, pp. 156–168, 2018.

[17] H.-F. Yu, N. Rao, and I. S. Dhillon, "Temporal regularized matrix factorization for high-dimensional time series prediction," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016.

[18] X. Chen, C. Zhang, X.-L. Zhao, N. Saunier, and L. Sun, "Nonstationary temporal matrix factorization for multivariate time series forecasting," 2022.

[19] Q. Song, H. Ge, J. Caverlee, and X. Hu, "Tensor completion algorithms in big data analytics," *ACM Trans. Knowl. Discov. Data*, vol. 13, no. 1, jan 2019.

[20] 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*, vol. 35, no. 1, pp. 208–220, 2013.

[21] P. Zhou, C. Lu, Z. Lin, and C. Zhang, "Tensor factorization for low-rank tensor completion," *IEEE Transactions on Image Processing*, vol. 27, no. 3, pp. 1152–1163, 2018.

[22] M. E. Kilmer, C. D. Martin, and L. Perrone, "A third-order generalization of the matrix SVD as a product of third-order tensors," Tufts University, Department of Computer Science, Tech. Rep. TR-2008-4, October 2008.

[23] M. E. Kilmer and C. D. Moravitz Martin, "Factorization strategies for third-order tensors," *Linear Algebra and Its Applications*, no. Special Issue in Honer of G.W.Stewart's 75^{th} birthday, 2009.

[24] K. Braman, "Third-order tensors as linear operators on a space of matrices," *Linear Algebra and its Applications*, vol. 433, no. 7, pp. 1241 – 1253, 2010.

[25] N. Hao, M. E. Kilmer, K. S. Braman, and R. C. Hoover, "New tensor decompositions with applications in facial recognition," *SIAM Journal on Imaging Science (SIIMS)*, vol. 6, no. 1, pp. 437–463, Feb. 2013.

[26] M. E. Kilmer, K. S. Braman, N. Hao, and R. C. Hoover, "Third order tensors as operators on matrices: A theoretical and computational framework with applications in imaging," *SIAM Journal on Matrix Analysis and Applications (SIMAX)*, vol. 34, no. 1, pp. 148–172, Feb. 2013.

[27] R. C. Hoover, K. S. Braman, and N. Hao, "Pose estimation from a single image using tensor decomposition and an algebra of circulants," in *Int. Conf. on Intel. Robots and Sys.*, 2011.

[28] R. C. Hoover, K. Caudle, and K. Braman, "Multilinear discriminant analysis through tensor-tensor eigendecomposition," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2018, pp. 578–584.

[29] X.-Y. Liu and X. Wang, "Fourth-order tensors with multidimensional discrete transforms," 2017.

[30] C. Ozdemir, R. C. Hoover, K. Caudle, and K. Braman, "High-order multilinear discriminant analysis via order-*n* tensor eigendecomposition," 2022.

[31] E. Kernfeld, M. Kilmer, and S. Aeron, "Tensor–tensor products with invertible linear transforms," *Linear Algebra and its Applications*, vol. 485, pp. 545–570, 2015.

[32] G. H. Golub and C. F. van Loan, *Matrix Computations*, 4th ed. JHU Press, 2013.

[33] S. Gultekin and J. Paisley, "Online forecasting matrix factorization," *IEEE Transactions on Signal Processing*, vol. 67, no. 5, pp. 1223–1236, 2019.

[34] G. Hyndman R. J., & Athanasopoulous, *Forecasting: principles and practice*. OTexts, 2018.

[35] Y.-X. Wang and Y.-J. Zhang, "Nonnegative matrix factorization: A comprehensive review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 6, pp. 1336–1353, 2013.

[36] N. Gillis, "The why and how of nonnegative matrix factorization," *Regularization, optimization, kernels, and support vector machines*, vol. 12, no. 257, pp. 257–291, 2014.

[37] V. P. Pauca, J. Piper, and R. J. Plemmons, "Nonnegative matrix factorization for spectral data analysis," *Linear algebra and its applications*, vol. 416, no. 1, pp. 29–47, 2006.