Transform-Based Tensor Auto Regression for Multilinear Time Series Forecasting

Jackson Cates, Randy C. Hoover, Kyle Caudle, Riley Kopp, Cagri Ozdemir
Department of Computer Science & Engineering and Department of Mathematics
South Dakota Mines
Rapid City, South Dakota

***Abstract*—With the massive influx of 2-dimensional observational data, new methods for analyzing, modeling, and forecasting multidimensional data need to be developed. The current research aims to accomplish these goals through the intersection of time-series modeling and multi-linear algebraic systems. In particular, the current research, aptly named the $\mathcal{L}$-Transform Tensor Auto-Regressive ($\mathcal{L}$-TAR for short) model expands previous auto-regressive techniques to forecast data from multilinear observations as oppose to scalars or vectors. The approach is based on recent developments in tensor decompositions and multilinear tensor products. Transforming the multilinear data through invertible discrete linear transforms enables statistical Independence between observations. As such, can be reformulated to a collection of vector auto-regression problems for model learning. Experimental results are provided on benchmark datasets containing image collections, video sequences, sea surface temperature measurements, and stock closing prices.**

I. Introduction

Forecasting is among the most challenging and problematic of machine learning tasks in that it involves extrapolation—prediction of the future from only past data [1]. Historically, numerous methods have been created to meet the challenges of forecasting. Some of the more traditional forecasting methods include exponential smoothing [2]–[4] and Box-Jenkins Auto Regressive Integrated Moving Average (ARIMA) [5]. Neural networks have also been utilized to provide very competitive forecasts [6], [7], although these methods are much less interpretable.

Among the aforementioned forecasting methods, Box-Jenkins Auto Regressive Integrated Moving Average (ARIMA) [5] modeling is one of the more popular methods. In an auto-regressive model, we forecast future values of a time-series by creating a linear combinations of previous time series observations. The "order" of the time series is the number of previous values (commonly referred to as lags) that are used to forecast the present value. For example, given the univariate AR model of order p,

$$y_t = \beta + \alpha_1 y_{t-1} + \alpha_2 y_{t-2} + ... + \alpha_p y_{t-p} + \epsilon_t, \qquad (1)$$

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.

the goal is to estimate the model parameters $\theta = \{\alpha_1, \alpha_2, \ldots, \alpha_p, \beta\}$, $i = 1, \ldots, p$ the prior observations $y_j \in \mathbb{R}$, $j = 1, \ldots, n$ where in general $n >> p$.

˜ []Asteriou. For example, given the multivariate VAR model of order p

$$\mathbf{y}_t = \mathbf{c} + A_1 \mathbf{y}_{t-1} + ... + A_p \mathbf{y}_{t-p} + \boldsymbol{\epsilon}_t, \qquad (2)$$

similar to the univariate case, the goal is to estimate the model parameters $\boldsymbol{\theta} = \{A_1, A_2, \ldots, A_p, \mathbf{c}\}$, where $A_i \in \mathbb{R}^{k \times k}$, from the collection of observations $\mathbf{y}_j \in \mathbb{R}^k$, $j = 1, 2, \ldots, n$. Again, it is generally assumed that $n \gg p$.

With the massive influx of 2-dimensional (2D) sensors available today, extensions of such methods have been pursued where the observations are no longer vectors but can be viewed as a lateral slice of a tensor instead (e.g. $\mathcal{Y}_t \in \mathbb{R}^{i \times 1 \times j}$) [1]. Tensor in this context is a multi-dimensional array, often referred to as n-mode or n-way array as defined in Section II. Specific examples of where tensor-based forecasting may arise is spatiotemporal data such as dynamic networks, video sequencing, correlated image sets, and distributed sensing to name but a few. In [8] the authors developed a method to forecast higher-order tensors based on the Tucker decomposition and n-mode products (referred to as multi-linear dynamical systems (MLDS)) [9], [10]. The MLDS approach (based on dynamical systems theory and system identification methods) was extended in [11] by transitioning from the Tucker decomposition to a recently defined tensor product based on discrete transforms and mod-n convolution, referred to as the $\mathcal{L}$-transform [12]–[16] (the details of which are outlined in Section II). While both methods outlined in [8] and [11] (MLDS and $\mathcal{L}$-MLDS respectively) obtained promising performance, they are both based on multi-linear dynamical systems modeling as opposed to an auto-regressive model as defined above. In other words, they attempt to find a single state-transition tensor to obtain their forecast.

The current paper extends these results in two fundamental ways: First, we transit from a traditional dynamical systems model and approach the problem from an auto-regressive perspective. Building on [15]–[18], and the $\mathcal{L}$-transform outlined in [11] we show that extensions to the VAR model can be achieved by estimating the model parameters $\Theta =$

[1]Note: It's customary in the literature to represent tensors with upper-case calligraphic letters

$\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_p, \mathcal{C}\}$ of the tensor auto-regressive model ($\mathcal{L}$-TAR)

$$\mathcal{Y}_t = \mathcal{C} + \mathcal{A}_1 \bullet \mathcal{Y}_{t-1} + \cdots + \mathcal{A}_p \bullet \mathcal{Y}_{t-p} + \mathcal{E}_t$$

where $\bullet$ denotes the $\mathcal{L}$-**product** outlined in definition 3, $\mathcal{A}_i \in \mathbb{R}^{i\times i\times j}$ is the state-transition tensor for lag i, and $\mathcal{C} \in \mathbb{R}^{i\times 1\times j}$ is a tensor of centers. Second, we add the capability for modeling non-stationary tensor data by adding an integration (differencing) step that results in an extension to the classical auto-regressive integrated (ARI) model in a tensor framework. We refer to this model as an $\mathcal{L}$-TARI model. Finally, to account for the potential of seasonality in the data, we extend the classical seasonal auto-regressive model to the tensor framework. We refer to this model as an $\mathcal{L}$-STAR. Experimental results are presented on some benchmark datasets as outlined in [8], [11] and compared against both the traditional MLDS models, a long-short term memory artificial neural network (LSTM-ANN), and different variations of the proposed approach (both different models and different discrete transforms).

The remainder of the paper is organized as follows: In Section II we provide some mathematical background for the tensor linear algebra. In Section III we provide some preliminary information regarding the $\mathcal{L}$-TAR method and outline the different variants of $\mathcal{L}$-TAR ($\mathcal{L}$-TARI, $\mathcal{L}$-STAR, and $\mathcal{L}$-STARI) . In Section IV we provide some experimental results of our proposed method first with synthetic data and then with 3 standard benchmark data sets. Section V contains our conclusions and some interpretive remarks.

II. Mathematical Preliminaries

In the current section we discuss the mathematical foundations of the tensor decompositions used in the current work. While much of the theory in this section is outlined in [12]–[16], [19], we summarize this theory here to keep the current work self-contained.

A. Mathematical Preliminaries

The term *tensor*, as used in the context of this paper, 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.

First, we review the basic definitions from [13] and [12] and introduce some basic notation. It will be convenient to break a tensor $\mathcal{A}$ in $\mathbb{R}^{\ell\times m\times n}$ up into various slices and tubal elements, and to have an indexing on those. The i^{th} lateral slice will be denoted $\mathcal{A}_{(i)}$ whereas the j^{th} frontal slice will be denoted $\mathcal{A}^{(j)}$. In terms of *Python* slicing, this means $\mathcal{A}_{(i)} \equiv \mathcal{A}[:,:,i]$ while $\mathcal{A}^{(j)} \equiv \mathcal{A}[j,:,:]$. We use the notation $\mathbf{a}_{ik}$ to denote the i,k^{th} *frontal* tube in $\mathcal{A}$; i.e., $\mathbf{a}_{ik} = \mathcal{A}[:,i,k]$, and $\mathbf{a}^{ik}$ as the i,k^{th} *vertical* tube in $\mathcal{A}$; i.e., $\mathbf{a}^{ik} = \mathcal{A}[i,:,k]$. Indeed, these tubes have special meaning for us in the present work, as they will play a role similar to scalars in $\mathbb{R}$. Thus, we make the following definition:

Definition 1: An element $\mathbf{c} \in \mathbb{R}^{1\times 1\times n}$ is called a **tubal-scalar** of length n.

Fundamental to the results presented in this paper is a recently defined multiplication operation on third-order tensors which itself produces a third-order tensor [12], [13] (referred to as the **t-product**). While the original formulation outlined in [12], [13] was built around the discrete Fourier transform (DFT) and an algebra of circulants, the resulting complex arithmetic associated with the **t-product** becomes computationally prohibitive for large datasets. Therefore, the research community began searching for alternative solutions and arrived at two variations on the original formulation that utilize either the discrete cosine transform (DCT), or the discrete wavelet transform (DWT) [19]. Combining the notation outlined in [19] with the prior work outlined in [12]–[16], we define the following operators:

We anchor the `MatVec` command to the frontal slices of the tensor such that $\texttt{MatVec}(\mathcal{A})$ takes an $\ell \times m \times n$ tensor and returns a block $\ell n \times m$ matrix

$$\texttt{MatVec}(\mathcal{A}) = \begin{bmatrix} A^{(1)} \\ A^{(2)} \\ \vdots \\ A^{(n)} \end{bmatrix}.$$

We anchor the `MatView` command to the frontal slices of the tensor such that $\texttt{MatView}(\mathcal{B})$ takes an $\ell \times m \times n$ tensor and returns a block diagonal $\ell n \times mn$ matrix

$$\texttt{MatView}(\mathcal{B}) = \begin{bmatrix} B^{(1)} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & B^{(2)} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & B^{(n)} \end{bmatrix}.$$

where the $\mathbf{0}$'s in the previous matrix represent an $\ell \times n$ zero matrix.

The operation that takes both $\texttt{MatVec}(\mathcal{A})$ and/or $\texttt{MatView}(\mathcal{B})$ back to tensor form is the `fold` command:

$$\texttt{fold}(\texttt{MatVec}(\mathcal{A})) = \mathcal{A} \text{ and/or } \texttt{fold}(\texttt{MatView}(\mathcal{B})) = \mathcal{B}.$$

Finally, we anchor the `Collect` command to the collection of either mode-1 or mode-2 tensors (i.e., vectors and matrices respectively) such that $\texttt{Collect}(A_i)$, $i = 1, 2, \ldots, m$ and $A_i \in \mathbb{R}^{k\times k}$ returns a tensor $\mathcal{A} \in \mathbb{R}^{k\times k\times m}$ with the A_i as its frontal slices with increasing i from front to back.

The above operators enable a generalized tensor product to be defined via any invertible discrete transform $\mathcal{L} : \mathbb{C}^n \to \mathbb{C}^n$. As such, we have the following definition:

Definition 2: 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 of your choice along the tubes $\mathbf{a}_{ik}$ of $\mathcal{A}$.[2]

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

Using this formulation, given two third order tensors $\mathcal{A} \in \mathbb{C}^{\ell \times m \times n}$ and $\mathcal{B} \in \mathbb{C}^{m \times p \times n}$, we have:

Definition 3: The $\mathcal{L}$**-product** between $\mathcal{A}$ and $\mathcal{B}$ can be defined via traditional convolution as

$$\mathcal{C} = \mathcal{A} \bullet \mathcal{B} = \mathcal{L}^{-1}(\texttt{fold}(\texttt{MatView}(\tilde{\mathcal{A}}) \cdot \texttt{MatVec}(\tilde{\mathcal{B}}))),$$

where we denote $\bullet$ as the $\mathcal{L}$**-product**, $\cdot$ is computed via classical matrix multiplication, and the resulting tensor $\mathcal{C} = \mathcal{A} \bullet \mathcal{B} \in \mathbb{C}^{\ell \times m \times n}$.

III. Proposed Approach to Multilinear Time-Series Forecasting

In the current section we discuss the details of building the proposed extensions to the classical AR, ARI, and SARI models using the $\mathcal{L}$-transform and $\mathcal{L}$**-product**. Namely, we illustrate how using the $\mathcal{L}$-transform we can divide and conquer by recasting the multilinear time-series problem into a subset of linear VAR problems.

A. Mathematical Model Overview

Our overarching goal is to construct the p^{th} order tensor auto-regressive model (referred to as a $\mathcal{L}$-TAR(p) given by

$$\mathcal{Y}_t = \mathcal{C} + \mathcal{A}_1 \bullet \mathcal{Y}_{t-1} + \cdots + \mathcal{A}_p \bullet \mathcal{Y}_{t-p} + \mathcal{E}_t, \qquad (3)$$

by estimating the model parameters $\Theta = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_p, \mathcal{C}\}$ from a collection of multilinear observations $\mathcal{Y}_j \in \mathbb{R}^{\ell \times 1 \times m}$, $j = 1, 2, \ldots, n$ with $n >> p$. In Eqn. (3), $\bullet$ denotes the $\mathcal{L}$**-product** outlined in definition 3, $\mathcal{A}_i \in \mathbb{R}^{\ell \times \ell \times m}$ is the model coefficient tensor for lag i, $\mathcal{C} \in \mathbb{R}^{\ell \times 1 \times m}$ is a tensor of centers, and $\mathcal{E}_t$ represents the model errors. We assume the model errors have zero mean, with constant variance, and are uncorrelated (i.e., $\text{E}\{\mathcal{E}\} = 0$, $\text{E}\{\mathcal{L}(\mathcal{E}, \mathcal{E}^T)\} = \Psi$, and $\text{E}\{\mathcal{E}_i, \mathcal{E}_j\} = 0$ for $i \neq j$). Because our aim is to learn model parameters from real data, the moving average (MA) portion of the ARMA process contains unobservable error terms. As a result, the formulation of the $\mathcal{L}$-TAR(p) models (and their non-stationary and seasonal counterparts) focus on estimating the auto-regression coefficients only. A graphical illustration of the $\mathcal{L}$-TAR(p) model is shown in Fig. **??**.

B. Training Methodology

We proceed by computing $\tilde{\mathcal{Y}}_j = \mathcal{L}(\mathcal{Y}_j)$ for each $j = 1, 2, \ldots, n$. We note that, although it is assumed the multilinear observations $\mathcal{Y}_j$ are correlated in the sampling domain, the vertical tubes $\mathbf{y}^{ik}$ are uncorrelated in the transform domain. As such, $\mathcal{L}(\mathcal{Y}_j)$ transforms a single multilinear observation $\mathcal{Y}_j$ into a collection of m vector observations $\mathbf{y}_j^i \in \mathbb{C}^{\ell \times 1 \times 1}$ for $j = 1, 2, \ldots, n$ and $i = 1, 2, \ldots, m$. In other words, we now have a collection of n multivariate observations $\mathbf{y}_j^i$ sampled from m different VAR processes in the transform domain. As such, we can apply the techniques of standard VAR process learning (least squares regression, maximum likelihood, or expectation maximization) to learn m different VAR model parameters $\boldsymbol{\theta}_i = \{A_1^i, A_2^i, \ldots, A_p^i, \mathbf{c}^i\}$, $i = 1, 2, \ldots, m$ as outlined in Eqn. (1). Applying the $\texttt{Collect}(\cdot)$ command to each of the parameter matrices/vectors in $\boldsymbol{\theta}_i$ for each i results in the parameter tensors $\{\tilde{\mathcal{A}}_1, \tilde{\mathcal{A}}_2, \ldots, \tilde{\mathcal{A}}_p, \tilde{\mathcal{C}}\}$. Finally, the inverse of the $\mathcal{L}$-transform is applied resulting in the $\mathcal{L}$-TAR(p) model parameters

$$\mathcal{L}^{-1}\{\tilde{\mathcal{A}}_1, \tilde{\mathcal{A}}_2, \ldots, \tilde{\mathcal{A}}_p, \tilde{\mathcal{C}}\} \rightarrow \Theta = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_p, \mathcal{C}\}.$$

The entire process for constructing the $\mathcal{L}$-TAR(p) model is illustrated graphically in Fig. 1.

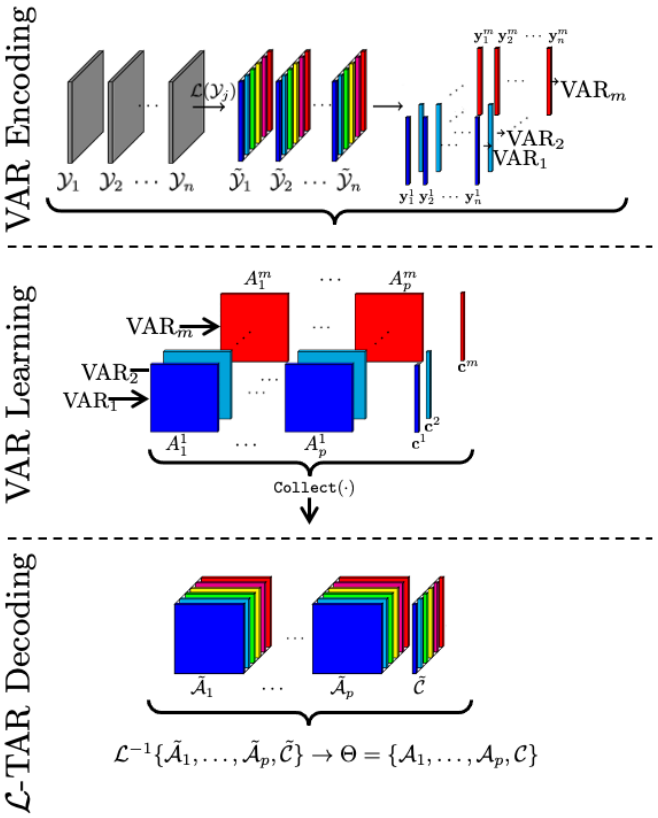

Fig. 1: Graphical illustration of the overall process for computing the $\mathcal{L}$-TAR(p) model. The top row illustrates how the original observations $\mathcal{Y}_j$ are encoded using the $\mathcal{L}$-transform to construct the individual p observations $\mathbf{y}_j^i$ for learning the m VAR models in the transform domain. The middle row illustrates how the m VAR model parameters $\boldsymbol{\theta}_i$ are learned and collected back to tensor form using the $\texttt{Collect}(\cdot)$ command. The bottom row illustrates how the inverse $\mathcal{L}$-transform is applied to compute the $\mathcal{L}$-TAR(p) model parameters $\Theta = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_p, \mathcal{C}\}$.

C. Dealing with seasonality and non-stationarity

The formulation of the $\mathcal{L}$-TAR(p) assumes two conditions: 1) that the observations $\mathcal{Y}_j$ are stationary and 2) there is no seasonal trend within the observations. While these assumptions are valid in many real-world applications, there are many instances where these assumptions are violated. Luckily, the statistics community has overcome these hurdles but extending the traditional VAR(p) models to account for seasonality,

non-stationarity, or both. This subsection illustrates how such extensions can be capitalized on in a multilinear framework.

1) Dealing with non-stationary data ($\mathcal{L}$-TARI): It's common practice to use integration within an autoregressive model in order to enforce stationarity within a time series. Similar to a traditional VAR(p) process, the proposed $\mathcal{L}$-TAR(p) process requires the tensor time-series to be stationary [20]. A time series is stationary if the observations $\mathcal{Y}_j$ have constant mean and variance, i.e., $\mathrm{E}\{\mathcal{Y}\} = \mathcal{M}$ and $\mathrm{E}\{(\mathcal{Y} - \mathcal{M})^2\} = \Psi$, where $\mathcal{M} \in \mathbb{R}^{\ell \times 1 \times m}$ is the mean tensor. Enforcing stationarity within the tensor time-series can be performed similar to how it's enforced in the VAR process, by applying a number of lagged differences to our observations, $\mathcal{Y}_j$. The resulting multilinear model is referred to as a $\mathcal{L}$-TARI(p, d) model, where $d \in \mathbb{Z}^+$ is the order of differencing. The order of differencing d is the amount of times that equation 4 is applied to the observations $\mathcal{Y}_j$. We apply the lagged difference d times as

$$\mathcal{Y}'_j = \mathcal{Y}_j - \mathcal{Y}_{j-1} \tag{4}$$

Then the model is constructed using the differenced observations $\mathcal{Y}'_j$, and the forecast is performed for $w \in \mathbb{Z}^+$ steps. This obtains a multilinear response $\hat{\mathcal{Y}}'_k$ for $k = n+1, n+2, \ldots, n+w$. The differencing must be removed to recover the response using

$$\hat{\mathcal{Y}}_k = \sum_{q=1}^{w} \hat{\mathcal{Y}}'_{k-q} + \mathcal{Y}_n \tag{5}$$

Equation 5 must be applied to the forecast $\hat{\mathcal{Y}}'_k$ for d times.

2) Dealing with seasonal data ($\mathcal{L}$-STAR): It's also common practice to consider seasonality within the observations $\mathcal{Y}_t$ to enforce stationarity within a time series. In addition to requiring constant mean and variance, the time-series should not have any seasonality or periods within the data [20]. In a similar fashion of how non-stationary data enforces stationarity, We can account of seasonality by applying a seasonal difference to our observations $\mathcal{Y}_t$. The resulting multilinear model is referred to as $\mathcal{L}$-STAR(p, s), where we consider $1 < s < n$ as the period of the seasonality. We apply the seasonal difference to our observations $\mathcal{Y}_j$ as

$$\mathcal{Y}'_j = \mathcal{Y}_j - \mathcal{Y}_{j-s} \tag{6}$$

As before, the model is constructed using the differenced observations $\mathcal{Y}'_j$, and the forecast is performed for w steps to obtain the multilinear response $\hat{\mathcal{Y}}'_k$. The difference must be removed to recover the response $\hat{\mathcal{Y}}_k$ using

$$\hat{\mathcal{Y}}_k = \sum_{q=1}^{w} \hat{\mathcal{Y}}'_{k-q*s} + \mathcal{Y}_{n-s+j \mod s} \tag{7}$$

3) Dealing with both non-stationary and seasonal data ($\mathcal{L}$-STARI): When presented with non-stationary observations after applying a seasonal difference, a combination of the methods mentioned in $\mathcal{L}$-TARI and $\mathcal{L}$-STAR can be done to remove trends within seasonal observations. This results in a multilinear model that is referred as $\mathcal{L}$-STARI(p, d, s), where we consider both the order of difference d and the period of seasonality s. This is done in the similar fashion of the two models, where we apply a number of differences in the observations $\mathcal{Y}_j$ to form $\mathcal{Y}'_j$, construct the model based on the differenced observations $\mathcal{Y}'_j$, then once the forecasted response $\hat{\mathcal{Y}}'_k$ is made, remove the differencing. The order of which difference to apply is free for choice, where we can apply the seasonal difference first from equation 6 then the lagged difference from equation 4, or vice-versa.

IV. Experimental Results

A. Quantitative Analysis

As a quantitative evaluation, we compare the proposed approach against the $\mathcal{L}$-MLDS model proposed in [11] as well as a convolutional Long Short-Term Memory Neural Network model due their recent explosion in time-series modeling. We use a subset of the same datasets proposed in the $\mathcal{L}$-MLDS model in [11] in an effort to compare and contrast both methods. A tabulated list of all models used in our evaluation are outlined in Table I, with the information pertaining to each dataset outlined in Table II (additional details on the datasets can be found in [11]).

TABLE I: Models Used in Experiments

Model	***Notes***
$\mathcal{L}$-TAR	$\mathcal{L}$-transform computed using the DWT and DCT
$\mathcal{L}$-TARI	$\mathcal{L}$-TAR for non-stationary data
$\mathcal{L}$-STAR	$\mathcal{L}$-TAR for seasonal data
LSTM	A Long Short-Term Memory Neural Network
$\mathcal{L}$-MLDS	Using DWT, DCT, and DFT

TABLE II: Datasets Used in Experiments

Dataset	***Notes***
SST	A 5×6 grid of sea-surface temperatures. The first 1800 hours are used as observations and the last 200 hours are used for testing. [11]
Video	A 10×10 video of the ocean. The first 1000 hours are used as observations and the last 171 hours are used for testing. [11]
NASDAQ-100	Opening, closing, high, and low for 50 randomly-chosen NASDAQ-100 companies (50×4). The first 2000 hours are used as observations and the last 186 hours are used for testing. [11]

=

In our single-step forecasting, we evaluate the original $\mathcal{L}$-MLDS and $\mathcal{L}$-TAR. The other methods are not considered because both $\mathcal{L}$-TARI and $\mathcal{L}$-STARI models are more suited for multi-step forecasting. We also show multi-step forecasting to illustrate the model's ability to make long-term predictions.

Fig. 2: Quantitative Evaluation of the proposed models. **Top:** Single Step with SST, Video, and NASDAQ100. **Bottom:** Multi-step (same order). The error is calculated by the percent error of the observation's Frobenius norm.

In our multi-step forecasting, we evaluate all methods in table I. $\mathcal{L}$-MLDS is modified in the multi-step forecasting to where the LDS models generated consider no error terms, unlike the authors original model [11] (i.e., the error terms are considered to be unobservable in the MLDS literature). The results of the experiment can be seen in figure 2.

1) SST: The SST dataset is a 5×6 grid of sea-surface temperatures, where the observations were recorded every hour [11]. Each observation can be represented as a multilinear observation $\mathcal{Y}_j \in \mathbb{R}^{5\times1\times6}$ where we have $n = 2000$. The first 1800 hours are used to construct the models and the last 200 hours are used for testing. By construction, we noticed that the data contains seasonality.

For single step, when estimating the model parameters $\Theta = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_p, \mathcal{C}\}$ for the $\mathcal{L}$-TAR(p), we found that $p = 5$ and gave the best result for the model. We compared our propose method to $\mathcal{L}$-MLDS and as seen on the top left of Fig. 2 $\mathcal{L}$-TAR(p) obviously performs better.

For multi-step, when estimating the model parameters $\Theta = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_p, \mathcal{C}\}$ for the models, we found that $p = 19$ gave the best result for L-TAR(p), $p = 19$ and $d = 1$ gave the best result for $\mathcal{L}$-TARI(p,d), and $p = 3$ and $s = 24$ gave the best result for $\mathcal{L}$-STAR(p, s). For the LSTM, two recurrent layers are used, both with a 100 nodes and relu activation functions. The LSTM was trained using stochastic gradient descent. As seen on the bottom left of Fig. 2 $\mathcal{L}$-TAR(p) is the best performer throughout.

2) Video: The video dataset is a 10×10 gray-scale video of the ocean, where the observations were recorded every frame [11]. Each observation can be represented as a multilinear observation $\mathcal{Y}_j \in \mathbb{R}^{10\times1\times10}$ where we have $n = 1171$. The first 1000 frames are used to construct the models and the last 171 frames are used for testing. By construction, we noticed that the data contains seasonality and is non-stationary.

For single step, when estimating the model parameters Θ for the $\mathcal{L}$-TAR(p), we found that $p = 10$ gave the best result for the model. We can see on the top middle of Fig. 2 both $\mathcal{L}$-TAR(p) and $\mathcal{L}$-MLDS virtually performs the same.

For multi-step, when estimating the model parameters Θ for the models, we found that $p = 13$ gave the best result for $\mathcal{L}$-TAR(p), $p = 9$ and $d = 1$ gave the best result for $\mathcal{L}$-TARI(p,d), and $p = 9$ and $s = 10$ gave the best result for $\mathcal{L}$-STAR(p, s). For the LSTM, two recurrent layers are used, both with a 125 nodes and sigmoid activation functions. The LSTM was trained using stochastic gradient descent. As seen on the bottom middle of Fig. 2 $\mathcal{L}$-TARI(p, d) performs the best until around the 40th frame, then the LSTM performs the best on forth.

3) NASDAQ-100: The NASDAQ-100 dataset contains the opening, closing, high and low stock price of the day for 50 random NASDAQ-100 companies, resulting in a 50×4 grid [11]. Each observation can be represented as a multilinear observation $\mathcal{Y}_j \in \mathbb{R}^{50\times1\times4}$ where we have $n = 2186$. The first 2000 days are used to construct the models and the last 186 days are used for testing. By construction, we noticed that the data does not contains seasonality, but it is non-stationary. As such, a $\mathcal{L}$-STAR(p, s) was not trained since there was no seasonality.

For single step, when estimating the model parameters Θ for the $\mathcal{L}$-TAR(p), we found that $p = 10$ gave the best result

for the model. We can see on the top right of Fig. 2 that while $\mathcal{L}$-MLDS performs slightly better throughout, the error between both models is extremely small and very comparable.

For multi-step forecasting, when estimating the model parameters Θ for the models, we found that $p = 5$ gave the best result for $\mathcal{L}$-TAR(p), $p = 16$ and $d = 1$ gave the best result for $\mathcal{L}$-TARI(p,d). For the LSTM, two recurrent layers are used, both with a 200 nodes and relu activation functions. The LSTM was trained using stochastic gradient descent. As seen on the bottom right of Fig. 2 $\mathcal{L}$-MLDS performs the best until around the 50^{th} day when the $\mathcal{L}$-LTARI(p, d) begins to outperform all other models. Similar to the single-step forecasting however, both models perform so similar up to day 100 neither appears to outperform the other long term.

V. Conclusions and Future Directions

Overall, with the experimental results $\mathcal{L}$-TAR(p), $\mathcal{L}$-TARI(p, d), and $\mathcal{L}$-STAR(p, s) are worthy models to forecast a multilinear time series. While not necessarily performing the best in every data set, there are situations where the proposed methods outperform the current state of the art. When considering speed, this is an area we are unable to truly evaluate at the time, because the original $\mathcal{L}$-MLDS is coded into MATLAB, and our proposed methods are coded in Python. Thus, a true evaluation between the two methods is difficult to make. However, when compared, the proposed methods was achieving close to a 100-fold speedup. With the current state, our proposed methods are much faster and much more tune-able to a given problem set (i.e., we can control the seasonality, non-stationarity, and number of lags in the auto-regression).

Future work includes implementing the MLDS in Python to get a true computational comparison. Future work also includes applying extensions to $\mathcal{L}$-TAR in a similar fashion to its auto-regressive predecessors, such as applying moving averages ($\mathcal{L}$-TARMA, $\mathcal{L}$-STARMA, $\mathcal{L}$-TARIMA, $\mathcal{L}$-STARIMA) and considering non-linearity with exogenous observations ($\mathcal{L}$-NTARX). Also, in the section IV, p. d, and s was picked via trial and error, future work will also include creating similar tensor versions of auto-correlation factor (ACF) and partial auto-correlation factor (PACF) plots to have a more precise method of estimating these parameters.

References

[1] J. G. De Gooijer and R. J. Hyndman, "25 years of time series forecasting," *International journal of forecasting*, vol. 22, no. 3, pp. 443–473, 2006.

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

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

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

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

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

[7] 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.

[8] 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.

[9] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, Sept. 1966.

[10] L. D. Lathauwer, B. D. Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM J. Matrix Anal. Appl.*, vol. 21, no. 4, pp. 1253–1278, March 2000.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

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

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