Block Hankel Tensor ARIMA for Multiple Short Time Series Forecasting

Qiquan Shi,[1] **Jiaming Yin,**[2] * **Jiajun Cai,**[3] * **Andrzej Cichocki,**[4] **Tatsuya Yokota,**[5]
Lei Chen,[1] **Mingxuan Yuan,**[1] **Jia Zeng**[1]

[1]Huawei Noah's Ark Lab, Hong Kong, China, [2]Tongji University, Shanghai, China
[3]The University of Hong Kong, Hong Kong, China, [4]The Skolkovo Institute of Science and Technology, Moscow, Russia
[5]Nagoya Institute of Technology, Nagoya, Japan; RIKEN Center for Advanced Intelligence Project, Japan
{shiqiquan, lc.leichen, yuan.mingxuan, zeng.jia}@huawei.com, 14jiamingyin@tongji.edu.cn, jjcai@connect.hku.hk,
a.cichocki@skoltech.ru, t.yokota@nitech.ac.jp

Abstract

This work proposes a novel approach for multiple time series forecasting. At first, multi-way delay embedding transform (MDT) is employed to represent time series as low-rank block Hankel tensors (BHT). Then, the higher-order tensors are projected to compressed core tensors by applying Tucker decomposition. At the same time, the generalized tensor Autoregressive Integrated Moving Average (ARIMA) is explicitly used on consecutive core tensors to predict future samples. In this manner, the proposed approach tactically incorporates the unique advantages of MDT tensorization (to exploit mutual correlations) and tensor ARIMA coupled with low-rank Tucker decomposition into a unified framework. This framework exploits the low-rank structure of block Hankel tensors in the embedded space and captures the intrinsic correlations among multiple TS, which thus can improve the forecasting results, especially for multiple short time series. Experiments conducted on three public datasets and two industrial datasets verify that the proposed BHT-ARIMA effectively improves forecasting accuracy and reduces computational cost compared with the state-of-the-art methods.

Introduction

Time series forecasting (TSF) is one of the most sought-after and yet arguably the most challenging tasks. It has played an important role in a wide range of areas including statistics, machine learning, data mining, econometrics, operations research for several decades. For example, forecasting the supply and demand of products can be used for optimizing inventory management, vehicle scheduling and topology planning, which are crucial for most aspects of supply chain optimization (**?**).

Among existing TSF approaches, autoregressive integrated moving average (ARIMA) (**?**) is one of the most popular and widely used linear models due to its statistical properties and great flexibility (**?**). ARIMA merges the autoregressive model (AR) and the moving average model (MA) with differencing techniques for non-stationary TSF. However, most existing ARIMA models need to predict multiples TS one by one and thus suffer from high computa-

tional cost, especially for a large number of TS. In addition, ARIMA models do not consider the intrinsic relationships among correlated TS, which may limit their performance. Recently, researchers from Facebook developed Prophet (**?**). It can estimate each TS well based on an additive model where non-linear trends are fit with seasonality and holidays. However, its computational expense grows steeply with increasing number of TS due to it estimates single TS separately.

Multiple TS arising from real applications can be reformulated as a matrix or even a high-order tensor (multi-way data) naturally. For example, the spatio-temporal grid of ocean data in meteorology can be shaped as a fourth-order tensor TS, wherein four factors are jointly represented as latitude, longitude, grid points and time (**?**). When dealing with tensors, traditional linear TSF models require reshaping TS into vectors. This vectorization not only causes a loss of intrinsic structure information but also results in high computational and memory demands.

Tensor decomposition is a powerful computational technique for extracting valuable information from tensorial data (**?**; **?**; **?**; **?**). Taking this advantage, tensor decomposition-based approaches can handle multiple TS simultaneously and achieve good forecasting performance (**?**; **?**; **?**; **?**; **?**). For example, Tucker decomposition integrated with AR model was proposed (**?**) to obtain the multilinear orthogonality AR (MOAR) model and the multilinear constrained AR model for high-order TSF. Moreover, some works incorporate decomposition with neural networks for more complex tensorial TS (**?**; **?**).

However, instead of being high-dimensional tensors in nature, many real-world TS are relatively short and small (**?**). For example, since the entire life cycle of electronic products like smartphones or laptops is usually quite short (around one year for each generation), their historical demand data of product materials and sales records are very small. Due to limited information, the future demands or sales of these products cannot be effectively predicted by *either* existing linear models *or* tensor methods and deep learning approaches.

Multi-way delay embedding transform (MDT) (**?**) is an emerging technique to Hankelize available data to a high-

---

order block Hankel tensor. In this paper, we conduct MDT on multiple TS along the temporal direction, and thus get a high-order block Hankel tensor (*i.e. a block tensor whose entries are Hankel tensors (?)*), which represents all the TS at each time point as a tensor in a high-dimensional embedded space. The transformed high-order data are assumed to have a low-rank or smooth manifold in the embedded space (**?**; **?**). With the block Hankel tensor, we then employ low-rank Tucker decomposition to learn compressed core tensors by orthogonal factor (projection) matrices. These projection matrices are jointly used to maximally preserve the temporal continuity between core tensors which can better capture the intrinsic temporal correlations than the original TS data. At the same time, we generalize classical ARIMA to tensor form and directly apply it on the core tensors *explicitly*. Finally, we predict a new core tensor at the next time point, and then obtain forecasting results for all the TS data simultaneously via inverse Tucker decomposition and inverse MDT. In short, we incorporate block Hankel tensor with ARIMA (**BHT-ARIMA**) via low-rank Tucker decomposition into a unified framework. This framework exploits low-rank data structure in the embedded space and captures the intrinsic correlations among multiple TS, leading to good forecasting results. We evaluate BHT-ARIMA on three public datasets and two industrial datasets with various settings. In a nutshell, **the main contributions of this paper are threefold:**

1) We employ MDT along the temporal direction to transform multiple TS to a high-order block Hankel tensor. To the best of our knowledge, we are the first to introduce MDT together with tensor decomposition into the field of TSF. In addition, we empirically demonstrate this strategy is also effective for existing tensor methods.

2) We propose to apply the orthogonal Tucker decomposition to explore intrinsic correlations among multiple TS represented as the compressed core tensors in the low-rank embedded space. Furthermore, we empirically study BHT-ARIMA with relaxed-orthogonality (*impose the orthogonality constraint on all the modes **except** the temporal mode*) which can obtain even slightly better results with reduced sensitivity to parameters.

3) We generalize the classical ARIMA to tensor form and incorporate it into Tucker decomposition in a unified framework. We explicitly use the learned informative core tensors to train the ARIMA model. Extensive experiments on both public and industrial datasets validate and illustrate that BHT-ARIMA significantly outperforms nine state-of-the-art (SOTA) methods in both efficiency and effectiveness, especially for multiple short TS.

Preliminaries and Related Work

Notations   The number of dimensions of a tensor is the *order* and each dimension is a *mode* of it. A vector is denoted by a bold lower-case letter $\mathbf{x} \in \mathbb{R}^I$. A matrix is denoted by a bold capital letter $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$. A higher-order ($N \geq 3$) tensor is denoted by a bold calligraphic letter $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$. The i_{th} entry of $\mathbf{x}$ is denoted by $\mathbf{x}_i$, and the (i, j)th entry of $\mathbf{X}$ is denoted by $\mathbf{X}_{i_1, i_2}$. The $(i_1, \ldots, i_N)$th entry of $\boldsymbol{\mathcal{X}}$ is denoted by $\boldsymbol{\mathcal{X}}_{i_1, \ldots, i_N}$. The Frobenius norm of a tensor

$\boldsymbol{\mathcal{X}}$ is defined by $\|\boldsymbol{\mathcal{X}}\|_F = \sqrt{<\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{X}}>}$, where $<\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{X}}> = \sum_{i_1} \sum_{i_2} \cdots \sum_{i_N} \boldsymbol{\mathcal{X}}_{i_1, \ldots, i_N}^2$ denotes inner product.

A mode-n product of $\boldsymbol{\mathcal{X}}$ and $\mathbf{U} \in \mathbb{R}^{I_n \times R_n}$ is denoted by $\boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}} \times_n \mathbf{U}^\top \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times R_n \times I_{n+1} \times \cdots \times I_N}$. Mode-$n$ unfolding is the process of reordering the elements of a tensor into matrices along each mode. A mode-n unfolding matrix of $\boldsymbol{\mathcal{X}}$ is denoted as $\mathbf{X}^{(n)} = \text{Unfold}(\boldsymbol{\mathcal{X}}) \in \mathbb{R}^{I_n \times \Pi_{i \neq n} I_i}$. Its inverse operation is fold: $\boldsymbol{\mathcal{X}} = \text{Fold}(\mathbf{X}^{(n)})$.

Tucker Decomposition   It represents a tensor $\boldsymbol{\mathcal{X}}_t \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ as a core tensor with factor (projection) matrices (**?**):

$$\boldsymbol{\mathcal{X}}_t = \boldsymbol{\mathcal{G}}_t \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \cdots \times_N \mathbf{U}^{(N)}, \quad (1)$$

where $\{\mathbf{U}^{(n)} \in \mathbb{R}^{I_n \times R_n}, n = 1, \ldots, N, \text{ and } R_n < I_n\}$ are projection matrices which usually has orthonormal columns and $\boldsymbol{\mathcal{G}}_t \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ is the core tensor with lower dimensions. The Tucker-rank of the $\boldsymbol{\mathcal{X}}_t$ is an N-dimensional vector: $[R_1, \ldots, R_n, \ldots, R_N]$, where n-th entry R_n is the rank of the mode-n unfolded matrix $\mathbf{X}_t^{(n)}$.

ARIMA

Let $\mathbf{x}_t$ as the actual data value at any time point t. $\mathbf{x}_t$ can be considered as a linear function of the past p values and past q observation of random errors, i.e., a ARMA (p, q) model:

$$\mathbf{x}_t = \sum_{i=1}^{p} \alpha_i \mathbf{x}_{t-i} - \sum_{i=1}^{q} \beta_i \boldsymbol{\epsilon}_{t-i} + \boldsymbol{\epsilon}_t, \quad (2)$$

where the random errors $\{\boldsymbol{\epsilon}_t\}$ are identically distributed with a mean of zero and a constant variance. $\{\alpha_i\}_{i=1}^p$ and $\{\beta_i\}_{i=1}^q$ are the coefficients of AR and MA, respectively. In practice, TS data are usually not stationary. The ARIMA model integrates a differencing method to deal with non-stationary TS data. Let $\Delta^d \mathbf{x}_t$ denote as the order-d differencing of $\mathbf{x}_t$ and an ARIMA(p, d, q) model is given by:

$$\Delta^d \mathbf{x}_t = \sum_{i=1}^{p} \alpha_i \Delta^d \mathbf{x}_{t-i} - \sum_{i=1}^{q} \beta_i \boldsymbol{\epsilon}_{t-i} + \boldsymbol{\epsilon}_t. \quad (3)$$

It has been one of the most popular TSF models and has many variants (**?**; **?**; **?**). ARIMA models are usually used for single TSF.

## Tensor decomposition-based methods

These tensor approaches preserve the structure of high-order data and can handle multiple TS simultaneously (**?**; **?**; **?**; **?**; **?**). The closest related work is the MOAR which collectively integrates Tucker decomposition with AR for high-order TSF (**?**):

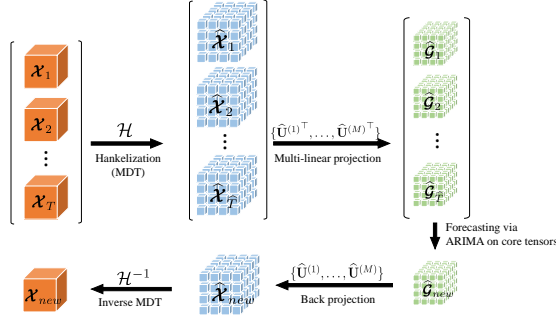$$\min_{\{\mathbf{U}^{(i)}\}, \{\alpha_i\}} \sum_{t=2}^{T} \left\| \left(\boldsymbol{\mathcal{X}}_t - \sum_{i=1}^{p} \alpha_i \boldsymbol{\mathcal{X}}_{t-i}\right) \prod_{n=1}^{N} \times_n \mathbf{U}^{(n)\top} \right\|_F^2 \quad (4)$$

$$\text{s.t. } \mathbf{U}^{(n)\top} \mathbf{U}^{(n)} = \mathbf{I}, n = 1, \ldots, N,$$

where $\mathbf{I}$ is an identity matrix and the core tensors are formulated in Tucker model and *implicitly* used to train the AR model. Our BHT-ARIMA differs from this closest related MOAR in three key aspects:1) BHT-ARIMA utilizes
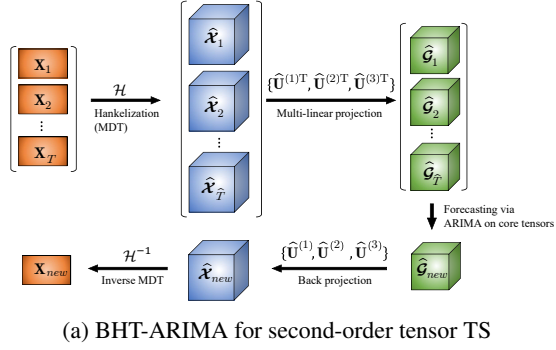
(a) BHT-ARIMA for second-order tensor TS

(b) BHT-ARIMA for third-order tensor TS

Figure 1: Illustration of the proposed method for prediction of multiple TS represented by a set of second/third-order tensors. Extension to higher-order tensors is straightforward.

the BHT as input which can be better estimated than original data, especially for shorter univariate TS. This will be also verified by using BHT to improve MOAR; 2) BHT-ARIMA explicitly uses low-rank core tensors to train its model while MOAR does not directly use them. Since the core tensors are smaller and can better capture the intrinsic temporal correlations than original TS data. BHT-ARIMA can improve computational speed and forecasting accuracy than MOAR; 3) BHT-ARIMA removes non-stationarity of TS data by using differencing technique which is not considered in MOAR.

Besides, some works integrate decomposition with neural networks (**?**; **?**; **?**), like tensor-train decomposition with recurrent neural network (TTRNN) (**?**) can achieve promising results for higher-order non-linear TS. However, these tensor methods are not applicable for short univariate TS which cannot be presented as high-order tensors naturally.

MDT for multiple TS

Hankelization is an effective way to transform lower-order data to higher-order tensors. MDT is a multi-way extension of Hankelization and show good results in tensor completion (**?**; **?**). It combines multi-linear duplication and multi-way folding operations. By denoting $\widehat{\mathcal{X}} \in \mathbb{R}^{J_1 \times J_2 \cdots \times J_M}$ as the block Hankel tensor of $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}, N < M$, the MDT for $\mathcal{X}$ is defined by

$$\widehat{\mathcal{X}} = \mathcal{H}_\tau(\mathcal{X}) = \text{Fold}_{(\mathbf{I},\tau)}(\mathcal{X} \times_1 \mathbf{S}_1 \cdots \times_N \mathbf{S}_N), \quad (5)$$

Figure 2: Illustration of applying MDT along the temporal mode of multiple TS represented as a matrix $\mathbf{X} \in \mathbb{R}^{I \times T}$. As we view the last mode N ($N = 2$ in this example) of multiple TS as the temporal mode, i.e., $I_N = T$, the last mode duplication matrix $\mathbf{S}_N \in \mathbb{R}^{\tau_N(I_N - \tau_N + 1) \times I_N} = \mathbf{S} \in \mathbb{R}^{\tau(T - \tau + 1) \times T}$. Note that $\mathbf{X} \times_N \mathbf{S} = \mathbf{X}\mathbf{S}^\top$ for matrix case. Extension to multi-modes is straightforward for tensors.

where $\mathbf{S}_n \in \mathbb{R}^{\tau_n(I_n - \tau_n + 1) \times I_n}$ is a duplication matrix and $\text{fold}_{(\mathbf{I},\tau)}$: $\mathbb{R}^{\tau_1(I_1 - \tau_1 + 1) \times \cdots \times \tau_N(I_N - \tau_N + 1)} \rightarrow \mathbb{R}^{\tau_1 \times (I_1 - \tau_1 + 1) \times \cdots \times \tau_N \times (I_N - \tau_N + 1)}$ constructs a higher order block Hankel tensor $\widehat{\mathcal{X}}$ from the input tensor $\mathcal{X}$. The inverse MDT for $\widehat{\mathcal{X}}$ is given by

$$\mathcal{X} = \mathcal{H}_\tau^{-1}(\widehat{\mathcal{X}}) = \text{Unfold}_{(\mathbf{I},\tau)}(\widehat{\mathcal{X}}) \times_1 \mathbf{S}_1^\dagger \cdots \times_N \mathbf{S}_N^\dagger, \quad (6)$$

where $\dagger$ is the Moore-Penrose pseudo-inverse.

Proposed Block Hankel Tensor ARIMA

To effectively and efficiently address the multiple (short) TSF problem, we propose to incorporate block Hankel tensor with ARIMA (**BHT-ARIMA**) via low-rank Tucker decomposition. The main idea of the proposed method is illustrated in Fig. 1. It consists of three major steps.

Step 1: Block Hankel Tensor via MDT

In this step, we aim to employ MDT to transform multiple TS to a high-order block Hankel tensor. Let $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times T}$ be the input data where each fiber is one TS and the last mode T is the temporal mode. We conduct the MDT only along the temporal direction, i.e.,

$$\widehat{\mathcal{X}} = \mathcal{H}_\tau(\mathcal{X}) \in \mathbb{R}^{J_1 \times \cdots \times J_M \times \widehat{T}}, \quad (7)$$

where $J_n = I_n$ for $n = 1, ..., N$, $J_M = \tau$ and $\widehat{T} = T - \tau + 1$. In this way, we get a block Hankel tensor in high-dimensional embedded space, where each frontal slice $\widehat{\mathcal{X}}_t \in \mathbb{R}^{J_1 \times \cdots \times J_M}$ contains the data points of all the TS at t-th time point.

Fig. 2 illustrates conducting MDT only along the temporal mode of a second-order tensor, i.e., using the duplication matrix $\mathbf{S}$ with τ to transform $\mathbf{X} \in \mathbb{R}^{I \times T}$ to a third-order

block Hankel tensor $\widehat{\boldsymbol{\mathcal{X}}} \in \mathbb{R}^{I \times \tau \times (T-\tau+1)}$. The block Hankel tensor is assumed to be low-rank or smooth in the embedded space (**?**).

Remark 1: We only apply MDT along the temporal mode because the relationship between neighbor items of multiple TS is usually not stronger than their temporal correlation. Thus, it is unnecessary to conduct MDT on all the modes which probably not be meaningful while costing more time (we empirically study it, see Fig. 6 in the Supplementary). Nevertheless, our proposed method is capable to apply MDT on more modes to get a higher-order block Hankel tensor.

In this paper, we mainly handle second-order and third-order original tensors, see more details of applying MDT on them in Appendix A.1 of Supplementary Material (**Supp.**)[1].

Step 2: Tensor ARIMA with Tucker Decomposition

In this step, we generalize the classical ARIMA to tensor form and incorporate it into Tucker decomposition. Specifically, with the block Hankel tensor, we compute its order-d differencing to get $\{\Delta^d \widehat{\boldsymbol{\mathcal{X}}}_t\}_{t=d}^{\widehat{T}}$. We then employ Tucker decomposition for $\{\Delta^d \widehat{\boldsymbol{\mathcal{X}}}_t\}$ by projecting it to core tensors $\{\Delta^d \widehat{\boldsymbol{\mathcal{G}}}_t\}$ using joint orthogonal factor matrices $\{\widehat{\mathbf{U}}^{(m)}\}$, i.e.,

$$\Delta^d \widehat{\boldsymbol{\mathcal{G}}}_t = \Delta^d \widehat{\boldsymbol{\mathcal{X}}}_t \times_1 \widehat{\mathbf{U}}^{(1)\top} \cdots \times_M \widehat{\mathbf{U}}^{(M)\top}$$
$$\text{s.t. } \widehat{\mathbf{U}}^{(m)\top} \widehat{\mathbf{U}}^{(m)} = \mathbf{I}, m = 1, \ldots, M, \quad (8)$$

where the projection matrices $\{\widehat{\mathbf{U}}^{(m)} \in \mathbb{R}^{J_m \times R_m}\}$ maximally preserve the temporal continuity between core tensors and the low-rank core tensors $\{\Delta^d \widehat{\boldsymbol{\mathcal{G}}}_t \in \mathbb{R}^{R_1 \times \cdots \times R_M}\}$ represent the most important information of original Hankel tensors and reflect the intrinsic interactions between TS.

Then, it would be more promising to train a good forecasting model directly using core tensors *explicitly* instead of whole tensors. To retain the temporal correlations among core tensors, we generalize a (p, d, q)-order ARIMA model to tensor form and use it to connect the current core tensor $\Delta^d \widehat{\boldsymbol{\mathcal{G}}}_t$ and the previous core tensors $\Delta^d \widehat{\boldsymbol{\mathcal{G}}}_{t-1}, \Delta^d \widehat{\boldsymbol{\mathcal{G}}}_{t-2}, \ldots, \Delta^d \widehat{\boldsymbol{\mathcal{G}}}_{t-p}$ as follows:

$$\Delta^d \widehat{\boldsymbol{\mathcal{G}}}_t = \sum_{i=1}^{p} \alpha_i \Delta^d \widehat{\boldsymbol{\mathcal{G}}}_{t-i} - \sum_{i=1}^{q} \beta_i \widehat{\boldsymbol{\mathcal{E}}}_{t-i} + \widehat{\boldsymbol{\mathcal{E}}}_t, \quad (9)$$

where the $\{\alpha_i\}$ and $\{\beta_i\}$ are the coefficients of AR and MA, respectively, and $\{\widehat{\boldsymbol{\mathcal{E}}}_{t-i}\}$ are the random errors of past q observations. In model (9), $\widehat{\boldsymbol{\mathcal{E}}}_t$ is the forecast error at the current time point, which should be minimized to optimal zero. We thus can derive the following objective function:

$$\min_{\{\widehat{\boldsymbol{\mathcal{G}}}_t, \widehat{\mathbf{U}}^{(m)}, \widehat{\boldsymbol{\mathcal{E}}}_{t-i}, \alpha_i, \beta_i\}} \sum_{t=s+1}^{\widehat{T}} \left(\frac{1}{2} \left\| \Delta^d \widehat{\boldsymbol{\mathcal{G}}}_t - \sum_{i=1}^{p} \alpha_i \Delta^d \widehat{\boldsymbol{\mathcal{G}}}_{t-i} + \sum_{i=1}^{q} \beta_i \widehat{\boldsymbol{\mathcal{E}}}_{t-i} \right\|_F^2 \right.$$
$$\left. + \frac{1}{2} \left\| \Delta^d \widehat{\boldsymbol{\mathcal{G}}}_t - \Delta^d \widehat{\boldsymbol{\mathcal{X}}}_t \times_1 \widehat{\mathbf{U}}^{(1)\top} \cdots \times_M \widehat{\mathbf{U}}^{(M)\top} \right\|_F^2 \right)$$
$$\text{s.t. } \widehat{\mathbf{U}}^{(m)\top} \widehat{\mathbf{U}}^{(m)} = \mathbf{I}, m = 1, \ldots, M, \quad (10)$$

---

where $s = p+d+q$ is the sum of ARIMA orders, and is also the minimum input length of each TS. Next, we solve this problem using augmented Lagrangian methods. To facilitate the derivation of (10), we reformulate the optimization problem by unfolding each tensor variable along mode-m:

$$\min_{\{\widehat{\mathbf{G}}_t^{(m)}, \widehat{\mathbf{E}}_{t-i}^{(m)}, \widehat{\mathbf{U}}^{(m)}, \alpha_i, \beta_i\}} \sum_{t=s+1}^{\widehat{T}} \sum_{m=1}^{M} \left(\frac{1}{2} \left\| \Delta^d \widehat{\mathbf{G}}_t^{(m)} - \sum_{i=1}^{p} \alpha_i \Delta^d \widehat{\mathbf{G}}_{t-i}^{(m)} \right. \right.$$
$$\left. + \sum_{i=1}^{q} \beta_i \widehat{\mathbf{E}}_{t-i}^{(m)} \right\|_F^2 + \frac{1}{2} \left\| \Delta^d \widehat{\mathbf{G}}_t^{(m)} - \widehat{\mathbf{U}}^{(m)\top} \widehat{\mathbf{X}}_t^{(m)} \widehat{\mathbf{U}}^{(-m)\top} \right\|_F^2 \right)$$
$$\text{s.t. } \widehat{\mathbf{U}}^{(m)\top} \widehat{\mathbf{U}}^{(m)} = \mathbf{I}, m = 1, \ldots, M, \quad (11)$$

where $\widehat{\mathbf{U}}^{(-m)} = \widehat{\mathbf{U}}^{(M)\top} \otimes \cdots \widehat{\mathbf{U}}^{(m+1)\top} \otimes \widehat{\mathbf{U}}^{(m-1)\top} \otimes \cdots \widehat{\mathbf{U}}^{(1)\top} \in \mathbb{R}^{\prod_{j \neq m} R_j \times \prod_{j \neq m} I_j}$. In the following, we can update each target variable using closed-form solutions [2].

Update $\widehat{\mathbf{G}}_t^{(m)}$ Equation (11) with respect to $\Delta^d \widehat{\mathbf{G}}_t^{(m)}$ is:

$$\min_{\{\widehat{\mathbf{G}}_t^{(m)}\}} \sum_{t=s+1}^{\widehat{T}} \sum_{m=1}^{M} \left(\frac{1}{2} \left\| \Delta^d \widehat{\mathbf{G}}_t^{(m)} - \sum_{i=1}^{p} \alpha_i \Delta^d \widehat{\mathbf{G}}_{t-i}^{(m)} + \sum_{i=1}^{q} \beta_i \widehat{\mathbf{E}}_{t-i}^{(m)} \right\|_F^2 \right.$$
$$\left. + \frac{1}{2} \left\| \Delta^d \widehat{\mathbf{G}}_t^{(m)} - \widehat{\mathbf{U}}^{(m)\top} \widehat{\mathbf{X}}_t^{(m)} \widehat{\mathbf{U}}^{(-m)\top} \right\|_F^2 \right). \quad (12)$$

Computing the partial derivation of this cost function with respect to $\widehat{\mathbf{G}}_t^{(m)}$ and equalize it to zero, we update $\widehat{\mathbf{G}}_t^{(m)}$ by:

$$\Delta^d \widehat{\mathbf{G}}_t^{(m)} = \frac{1}{2} \left(\widehat{\mathbf{U}}^{(m)\top} \widehat{\mathbf{X}}_t^{(m)} \widehat{\mathbf{U}}^{(-m)\top} \right.$$
$$\left. + \sum_{i=1}^{p} \alpha_i \Delta^d \widehat{\mathbf{G}}_{t-i}^{(m)} - \sum_{i=1}^{q} \beta_i \widehat{\mathbf{E}}_{t-i}^{(m)} \right). \quad (13)$$

Update $\widehat{\mathbf{U}}^{(m)}$ Equation (11) with respect to $\widehat{\mathbf{U}}^{(m)}$ is:

$$\min_{\{\widehat{\mathbf{U}}^{(m)}\}} \sum_{t=s+1}^{\widehat{T}} \sum_{m=1}^{M} \frac{1}{2} \left\| \Delta^d \widehat{\mathbf{G}}_t^{(m)} - \widehat{\mathbf{U}}^{(m)\top} \widehat{\mathbf{X}}_t^{(m)} \widehat{\mathbf{U}}^{(-m)\top} \right\|_F^2 \quad (14)$$
$$\text{s.t. } \widehat{\mathbf{U}}^{(m)\top} \widehat{\mathbf{U}}^{(m)} = \mathbf{I}, m = 1, \ldots, M.$$

The minimization of (14) over $\widehat{\mathbf{U}}^{(m)}$ with orthonormal columns is equivalent to the maximization of the well-known orthogonality Procrustes problem (**?**), whose global optimal solution is,

$$\widehat{\mathbf{U}}^{(m)} = \widehat{\mathbf{U}}^{*(m)} (\widehat{\mathbf{V}}^{*(m)})^\top, \quad (15)$$

where $\widehat{\mathbf{U}}^{*(m)}$ and $\widehat{\mathbf{V}}^{*(m)}$ are the left and right singular vectors of SVD of $\sum_{t=s+1}^{\widehat{T}} \widehat{\mathbf{X}}_t^{(m)} \widehat{\mathbf{U}}^{(-m)\top} \Delta^d \widehat{\mathbf{G}}_t^{(m)\top}$, respectively.

Discussion 1: ***Relaxed-orthogonality*** We empirically explored the effect of relaxing the full-orthogonality in (14) by **removing** the orthogonal constraints along the last mode (viewed as the temporal mode of each $\widehat{\boldsymbol{\mathcal{X}}}_t$ in the embedded space). This strategy probably relaxes the heavy constraints on temporal smoothness and thus would make the proposed

---

[1]Available at https://github.com/yokotatsuya/BHT-ARIMA.

[2]The detailed derivation is presented in *Appendix A of the Supp.*.

model more flexible and robust to variability of parameters, observed from our experimental results. We relax the last mode $\widehat{\mathbf{U}}^{(M)}$ without orthogonality constraint, and then compute the partial derivation of Eq. (14) with respect to $\widehat{\mathbf{U}}^{(M)}$ and equalize it to zero. Thus, we can update it by

$$
\widehat{\mathbf{U}}^{(M)} = \Big(\sum_{t=s+1}^{\widehat{T}} \widehat{\mathbf{X}}_t^{(M)} \widehat{\mathbf{U}}^{(-M)\dagger} (\widehat{\mathbf{X}}_t^{(M)} \widehat{\mathbf{U}}^{(-M)\dagger})^\top \Big)^{-1}
$$
$$
\Big(\sum_{t=s+1}^{\widehat{T}} \widehat{\mathbf{X}}_t^{(M)} \widehat{\mathbf{U}}^{(-M)\dagger} \Delta^d \widehat{\mathbf{G}}_t^{(M)\top} \Big). \tag{16}
$$

Update $\widehat{\mathbf{E}}_{t-i}^{(m)}$ Equation (11) with respect to $\widehat{\mathbf{E}}_{t-i}^{(m)}$ is:

$$
\min_{\{\widehat{\mathbf{E}}_{t-i}^{(m)}\}} \sum_{t=s+1}^{\widehat{T}} \sum_{m=1}^{M} \frac{1}{2} \Big\| \Delta^d \widehat{\mathbf{G}}_t^{(m)} - \sum_{i=1}^{p} \alpha_i \Delta^d \widehat{\mathbf{G}}_{t-i}^{(m)} + \sum_{i=1}^{q} \beta_i \widehat{\mathbf{E}}_{t-i}^{(m)} \Big\|_F^2 . \tag{17}
$$

Computing the partial derivation of Eq. (17) with respect to $\widehat{\mathbf{E}}_{t-i}^{(m)}$ and equalize it to zero, we can update $\widehat{\mathbf{E}}_{t-i}^{(m)}$ by

$$
\widehat{\mathbf{E}}_{t-i}^{(m)} = \frac{\sum_{t=s+1}^{\widehat{T}} \Big(\Delta^d \widehat{\mathbf{G}}_t^{(m)} - \sum_{i=1}^{p} \alpha_i \Delta^d \widehat{\mathbf{G}}_{t-i}^{(m)} + \sum_{j \neq i}^{q} \beta_j \widehat{\mathbf{E}}_{t-j}^{(m)} \Big)}{(s+1-\widehat{T})\beta_i} . \tag{18}
$$

Update $\{\alpha_i\}, \{\beta_i\}$ Regarding the coefficient parameters $\{\alpha_i\}_{i=1}^{p}, \{\beta_i\}_{i=1}^{q}$ of AR and MA respectively in the objective function (11), we follow the classical ARIMA based on Yule-Walker method to estimate them. We generalize a least squares modified Yule-Walker technique to support tensorial data and then estimate the $\{\alpha_i\}_{i=1}^{p}$ from the core tensors. Then, we estimate the $\{\beta_i\}_{i=1}^{q}$ from the residual time series.

Step 3: Forecasting $\mathcal{X}_{T+1}$

Finally, we apply the learned model to get a new $\Delta^d \widehat{\mathcal{G}}_{\widehat{T}+1}$:

$$
\Delta^d \widehat{\mathcal{G}}_{\widehat{T}+1} = \sum_{i=1}^{p} \alpha_i \Delta^d \widehat{\mathcal{G}}_{\widehat{T}-i} - \sum_{i=1}^{q} \beta_i \widehat{\mathcal{E}}_{\widehat{T}-i} . \tag{19}
$$

After obtaining $\Delta^d \widehat{\mathcal{G}}_{\widehat{T}+1}$, we reconstruct a new tensor by Tucker model with optimized factor matrices: $\Delta^d \widehat{\mathcal{X}}_{\widehat{T}+1} = \Delta^d \widehat{\mathcal{G}}_{\widehat{T}+1} \prod_{m=1}^{M} \times_m \widehat{\mathbf{U}}^{(m)}$. We then conduct inverse order-d differencing for $\Delta^d \widehat{\mathcal{X}}_{\widehat{T}+1}$ and get $\widehat{\mathcal{X}}_{\widehat{T}+1} \in \mathbb{R}^{J_1 \times \cdots \times J_M \times (\widehat{T}+1)}$ in the embedded space. Finally, we apply inverse MDT to get $\mathcal{X}_{T+1} = \mathcal{H}_\tau^{-1}(\widehat{\mathcal{X}}_{\widehat{T}+1}) \in \mathbb{R}^{I_1 \times \cdots \times I_N \times (T+1)}$ to get the predicted values at the $T+1$-th time point for all TS *simultaneously*. Furthermore, we could do long-term forecasting by using prior foretasted values in last steps. Although this way may lead to error accumulation to some degree, it becomes more practical for real-world applications (**?**).

Remark 2:   We explicitly use the compressed core tensors to train the model including parameters estimation. That is different from existing tensor methods like MOAR which implicitly use the core tensors to train their models. In this

---

Algorithm 1 TSF using **BHT-ARIMA**

1: **Input:** A time series data $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times T}$, (p, d, q), τ, maximum iteration K, and stop criteria *tol*.
2: **Step 1: Block Hankel Tensor via MDT**
3: Use MDT to transform the original tensor as a block Hankel tensor: $\widehat{\mathcal{X}} = \mathcal{H}_\tau(\mathcal{X}) \in \mathbb{R}^{J_1 \times \cdots \times J_M \times \widehat{T}}$, where each frontal slice $\widehat{\mathcal{X}}_t \in \mathbb{R}^{J_1 \times \cdots \times J_M}$ refers to all the time series at t-th time point.
4: Set values for $\{R_m\}_{m=1}^{M}$ and initialize $\{\widehat{\mathbf{U}}^{(m)}\}_{m=1}^{M}$ randomly.
5: **Step 2: Tensor ARIMA with Tucker decomposition**.
6: Conduct order-d differencing for $\{\widehat{\mathcal{X}}_t\}_{t=1}^{\widehat{T}}$ and get $\{\Delta^d \widehat{\mathcal{X}}_t\}_{t=d+1}^{\widehat{T}}$
7: Initialize $\{\{\widehat{\mathcal{E}}_{t-i} \in \mathbb{R}^{R_1 \times \cdots \times R_M}\}_{i=1}^{q}\}_{i=s+1}^{\widehat{T}}$ randomly.
8: **for** $k = 1, ..., K$
9:    Compute all the latent low-dimensional core tensors $\{\Delta^d \widehat{\mathcal{G}}_t = \Delta^d \widehat{\mathcal{X}}_t \times_1 \widehat{\mathbf{U}}^{(1)\top} \cdots \times_M \widehat{\mathbf{U}}^{(M)\top}\}_{t=1}^{\widehat{T}}$;
10:    Estimate coefficients $\{\alpha_i\}_{i=1}^{p}, \{\beta_i\}_{i=1}^{q}$ of AR and MA based on Yule-Walker equations based on $\{\Delta^d \widehat{\mathcal{G}}_t\}_{t=d+1}^{\widehat{T}}$
11:    **for** $n = 1, ..., M$
12:        Update $\Delta^d \widehat{\mathbf{G}}_t^{(m)}$ by (13) and get $\Delta^d \widehat{\mathcal{G}}_t = \text{Fold}\Big(\Delta^d \widehat{\mathbf{G}}_t^{(m)} \Big)$
13:        Update $\widehat{\mathbf{U}}^{(m)}$ by (15)
14:        If applying relaxed-orthogonality, update $\widehat{\mathbf{U}}^{(M)}$ by (16).
15:        **for** $i = 1, ..., q$
16:            Update $\widehat{\mathbf{E}}_{t-i}^{(n)}$ by (18) and update $\widehat{\mathcal{E}}_{t-i}$ by $\text{Fold}\Big(\widehat{\mathbf{E}}_{t-i}^{(m)} \Big)$
17:    Convergence checking: if $\frac{\sum_{m=1}^{M} ||\widehat{\mathbf{U}}^{(m)^{k+1}} - \widehat{\mathbf{U}}^{(m)^k}||_F^2}{\sum_{m=1}^{M} ||\widehat{\mathbf{U}}^{(m)^{k+1}}||_F^2} <$ *tol*, break; otherwise, continue.
18: **Step 3: Forecasting**
19: Compute new observation by (19) to get $\Delta^d \widehat{\mathcal{G}}_{\widehat{T}+1}$ and then compute $\Delta^d \widehat{\mathcal{X}}_{\widehat{T}+1} = \Delta^d \widehat{\mathcal{G}}_{\widehat{T}+1} \prod_{m=1}^{M} \times_m \widehat{\mathbf{U}}^{(m)}$.
20: Conduct inverse differencing for $\Delta^d \widehat{\mathcal{X}}_{\widehat{T}+1}$ and get $\widehat{\mathcal{X}}_{\widehat{T}+1}$.
21: Conduct inverse MDT: $\mathcal{X}_{T+1} = \mathcal{H}_\tau^{-1}(\widehat{\mathcal{X}}_{\widehat{T}+1})$
22: **Output:** $\mathcal{X}_{T+1}, \{\widehat{\mathbf{U}}^{(1)}, \ldots, \widehat{\mathbf{U}}^{(m)}, \ldots, \widehat{\mathbf{U}}^{(M)}\}$.

---

manner, we not only reduce the computational cost since the size of core tensors $\{\widehat{\mathcal{G}}_t\}$ are much smaller than whole tensors $\{\widehat{\mathcal{X}}_t\}$ based on the low-rank assumption in the embedded space, but also improve the forecasting accuracy by utilizing the mutual correlations among multiple TS in the model building process.

Finally, we summarize the proposed **BHT-ARIMA** in **Algorithm 1** and further evaluate it in the following section.

Experiments

Due to limited space, we present the detailed experimental setup and parameter analysis with figures in the Supp..

Experimental Setup

Datasets   We evaluate the proposed BHT-ARIMA by conducting experiments on five real-world datasets, including: i)*Three publicly available datasets*: **Traffic** is originally collected from California department of transportation and describes the road occupy rate of Los Angeles County highway

network. We here use the same subset in (**?**) which selects 228 sensors randomly and we aggregate it to daily interval for each TS with 80 days data points. **Electricity** records 321 clients' hourly electricity consumption (**?**). We merge the every 24 time points to obtain a daily interval TS dataset with size 321×1096. **Smoke Video** records the Smoke from the chimney of a factory taken in Hokkaido in 2007. We sample and resize the images to obtain a third-order TS dataset of size $36 \times 64 \times 100$;

ii)*Two industrial datasets* from the supply chain of **Huawei**: **PC sales** has 105 weekly sales records of 9 personal computers from 2017 to 2019; **Raw materials** includes 2246 material items of making a product, each item has 24 monthly demand quantities. We illustrate these datasets in Fig. 1 and 2 in the Supp..

Compared methods   We compared *nine* competing methods: i) the classical **ARIMA**, Vector AR (**VAR**) and **XG-Boost** (**?**); ii) the two popular industrial forecasting methods: Facebook-**Prophet**, and Amazon-**DeepAR** (**?**); iii) Neural network based methods[3]: **TTRNN** (**?**) and Gated Recurrent Units (**GRU**) (**?**); iv) the two matrix/tensor-based methods: **TRMF** (**?**), and **MOAR**; In addition, we combine MDT with MOAR by using our obtained block Hankel tensor as the input of MOAR to get **BHT+MOAR** to evaluate the effectiveness of MDT together with tensor decomposition.

Parameter settings   All datasets are split into training sets (90%) and testing sets (10%). We conduct grid search over parameters for each model and dataset (See Appendix B.1 of Supp. in detail). For our BHT-ARIMA, we will show its analysis of parameters in the following. We measure the forecasting accuracy using the widely used Normalized Root Mean Square Error (**NRMSE**) metric.

Analysis of Parameters and Convergence

We here not only analyze the parameter sensitivity of BHT-ARIMA, but also study the effects of BHT-ARIMA with *relaxed-orthogonality* and *MDT on all modes* by testing on the Raw materials and Smoke video datasets, respectively.

Sensitivity Analysis of Parameters τ, $\{R_m\}$ and (p,d,q)
Fig. 3, 4 and 5 in the Supp. show the forecasting results using BHT-ARIMA with full-orthogonality (**FO**) versus (**vs.**) relaxed-orthogonality (**RO**) with different values of the parameters τ, $\{R_m\}$ and (p,d,q), respectively. Overall, BHT-ARIMA with RO is less sensitive and can achieve even slightly better forecasting accuracy than that with FO. Particularly, with very small τ (e.g. 1,2,5) and when the last mode rank $R_M = \tau$ (that maximally preserves the temporal dependencies among core tensors), BHT-ARIMA with FO can achieve better (similar) accuracy than that with RO. With the same differencing order d, the performance of BHT-ARIMA is relatively sensitive to too large p. Besides, the time cost of BHT-ARIMA with RO is larger than that of with FO due to the cost of computing Eq. (16) is larger than that of Eq. (15).

---

[3]We didn't show the comparison to Long Short-Term Memory (LSTM) as it yields similar results with GRU.
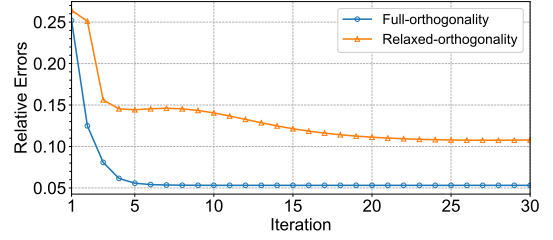

Figure 3: Convergence curves of BHT-ARIMA on the Raw materials dataset.

In short, we do not need to carefully tune the parameters for BHT-ARIMA while we usually can obtain better results by setting smaller values such as { $\tau = 2 - 4$, $R_M = \tau$ and small ($p = 1 - 5, d = 1, q = 1$)}. Moreover, the Tucker-rank can be estimated automatically (**?**; **?**).

Effect of applying MDT on all modes / temporal mode
As discussed in **Remark 1** about why we apply MDT only along the temporal mode, we here verify it by testing on the Smoke video. As shown in *Fig. 6 in the Supp.*: both types of applying MDT obtain similar forecasting accuracy while using MDT on all the modes costs more time due to computing higher-order tensors. This conclusion is also applicable for the cases of BHT-ARIMA with RO. These results support our assumption that these TS items usually do not have strong neighborhood relationships so it is unnecessary to applying MDT on other modes besides the temporal mode.

Convergence and maximum iteration K   We study the convergence of BHT-ARIMA in terms of the relative error of projection matrices $\frac{\sum_{m=1}^{M} ||\widehat{\mathbf{U}}^{(m)^{k+1}} - \widehat{\mathbf{U}}^{(m)^{k}}||_F^2}{\sum_{m=1}^{M} ||\widehat{\mathbf{U}}^{(m)^{k+1}}||_F^2}$. Fig. 3 shows that both BHT-ARIMA with FO and RO converge quickly while the FO version converges more smoothly and faster within 10 iterations. Furthermore, setting maximum iteration $K > 5$ is enough to get a sufficient forecasting accuracy, as shown in *Fig. 7(b) in the Supp.*. In this paper, we set $K = 10$ for BHT-ARIMA for all the tests.

Forecasting Accuracy Comparison

We report the average forecasting results of 10 runs in Tables 1, 2 and 3, where we highlight **the best results** in bold font and underline the second best results. Note that BHT-ARIMA used in the following comparisons is the full-orthogonality version.

Forecasting results of longer vs. shorter TS   To evaluate the capability of BHT-ARIMA for longer vs. shorter TS, we sample the first 40 time points of the Traffic dataset and thus get shorter **Traffic40** (228×40) and denote original whole set as **Traffic80**. For the Electricity, we sample the first 40 time points and get **Electricity40** (321×40), and the whole one is denoted as **Electricity1096**. The results are reported in **Table** 1: BHT-ARIMA outperforms all the existing competing methods in all the cases. Especially for shorter TS, BHT-ARIMA shows more advantage with 54.5% improvement on average on the Traffic data than that of on the Elec-

Table 1: Forecasting results comparison on the Electricity and Traffic (measured in NRMSE).

	Electricity1096 ($\times 10^{-2}$)	Electricity40 ($\times 10^{-2}$)	Traffic80 ($\times 10^{-3}$)	Traffic40 ($\times 10^{-3}$)
ARIMA	<u>1.316</u>	10.091	3.194	6.097
VAR	12.850	1.834	6.649	1.526
XGBoost	1.839	2.724	4.900	3.727
Prophet	13.799	5.948	4.343	1.992
DeepAR	4.742	4.857	8.178	5.358
TTRNN	2.686	3.565	5.723	3.432
GRU	1.574	<u>1.545</u>	<u>1.371</u>	<u>0.782</u>
TRMF	6.221	2.713	5.800	2.340
MOAR	4.731	7.677	10.689	12.200
BHT+MOAR	3.787	4.050	4.920	4.464
BHT-ARIMA	**1.114**	**1.456**	**0.599**	**0.493**

Table 2: Forecasting results comparison on the PC sales and Raw materials (measured in NRMSE).

	PC sales (9×105)	Raw materials_I (1533×24)	Raw materials_II (2246×24)
ARIMA	0.604	3.217	3.836
VAR	0.690	3. 387	4.033
XGBoost	0.618	3.834	4.231
Prophet	0.593	2.984	3.734
DeepAR	0.689	3.158	4.476
TTRNN	0.616	2.828	3.373
GRU	<u>0.524</u>	2.592	3.250
TRMF	0.689	3.167	4.362
MOAR	0.689	2.207	2.635
BHT+MOAR	0.683	<u>2.114</u>	<u>2.525</u>
BHT-ARIMA	**0.490**	**1.558**	**1.856**

tricity (7.9% improvement on average). GRU and ARIMA share the second best results. Note that BHT+MOAR performs consistently better than MOAR in all cases, which verifies the effectiveness of applying MDT together with tensor decomposition.

Forecasting results of industrial TS datasets In practice, industrial datasets are more complex with random irregular patterns and larger variance compared to these public datasets. As reported in the **Table** 2: 1) For PC sales with only nine TS items, BHT-ARIMA outperforms the second best performing method GRU by 6.5% on average; 2) For Raw materials dataset which has shorter length (24 months) than PC sales (105 weeks) while has much larger number of items (2246), we remove the items with missing values and get 1553 items namely **Raw materials_I** while the original one named as **Raw materials_II**. In such scenarios, BHT +
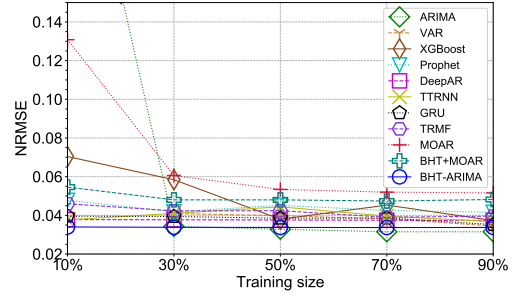

Figure 4: Forecasting accuracy comparison on the Smoke video. In the case of using 10% training size, the NRMSE of ARIMA is too large (0.277) so we truncate its curve there.

MOAR and GRU perform better than other existing methods, although their results are much worse than our BHT-ARIMA. These results further confirm the effectiveness of MDT together with tensor decomposition and also verified the improvement of explicitly using the core tensors to train our ARIMA model.

Forecasting results of Smoke video We further evaluate the performance of BHT-ARIMA on higher-order TS data using the Smoke video with 10% − 90% training set. Fig. 4 shows that BHT-ARIMA consistently forecasts the next video frame with smaller errors using even 10% training data (10 frames) where ARIMA, MOAR and other methods fail to keep their performance. Moreover, although ARIMA can achieve slightly better accuracy than ours with more than 50% training data, it surfers from extremely larger computational cost and memory requirements because like other linear models who cannot directly handle tensor TS data and need to reshape them into vectors.
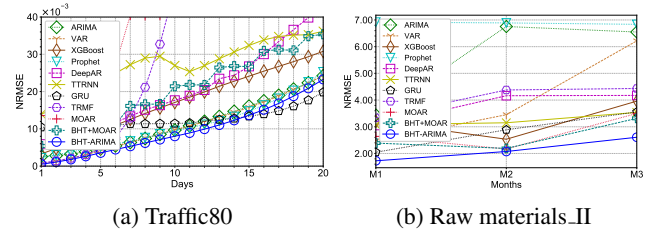

(a) Traffic80   (b) Raw materials_II

Figure 5: Long-term forecasting results for the Traffic and Raw materials (truncate curves with too large NRMSE).

Long-term Forecasting Comparison

Fig. 5 shows the comparison of long-term forecasting results, which further confirm the promising performance of BHT-ARIMA. Forecasting more steps, the errors of all the methods increase in general while BHT-ARIMA consistently keeps its best performance on the whole, especially on the shorter Raw materials dataset. Note that GRU slightly outperforms our method after 15 steps on Traffic80 dataset, but it requires near 900 times computational cost than ours.

Table 3: Computational cost (seconds) vs. SOTA algorithms on all five real-world TS datasets.

Time(s)	Electricity1096 (321×1096)	Electricity40 (321×40)	Traffic80 (228×80)	Traffic40 (228×40)	PC sales (9×105)	Raw materials_I (1533×24)	Raw materials_II (2246×24)	Smoke Video (32×64×100)
ARIMA	3322.62	327.60	453.63	382.32	13.01	1151.35	2067.48	1340.20
VAR	68.96	2.042	4.19	<u>0.92</u>	**0.27**	11.32	25.51	69.63
XGBoost	108.02	26.07	23.28	6.94	8.87	148.22	159.33	178.86
Prophet	2160.33	1304.07	1241.43	335.23	53.31	434.15	853.23	13813.21
DeepAR	175.56	171.17	119.02	112.04	78.43	136.80	286.66	106.65
TTRNN	165.84	104.38	88.77	95.47	22.83	25.49	28.05	27.82
GRU	4383.81	1077.09	1104.17	858.03	93.34	3353.43	3534.96	3142.46
TRMF	**16.03**	**0.57**	<u>1.90</u>	1.22	1.46	**0.27**	**0.28**	**0.77**
MOAR	28.96	10.21	6.75	5.78	1.68	566.19	1612.52	7.26
BHT+MOAR	42.86	12.28	8.28	5.93	1.70	865.91	1700.49	10.95
BHT-ARIMA	<u>23.24</u>	<u>0.89</u>	**1.28**	**0.58**	<u>1.41</u>	<u>0.91</u>	<u>1.13</u>	<u>3.67</u>

Time Cost Comparison

We report the average time cost of forecasting in **Table** 3. Although TRMF is slightly slower than VAR in a few cases, it is the fastest algorithm on the whole due to its core parts are implemented by C programming. BHT-ARIMA is the second fastest method while our implementations are not optimized for efficiency as our focus here is accuracy. MOAR is slower than ours mainly because it does not directly use the low-dimensional core tensors for training. ARIMA and GRU are the second best performing algorithms in a few cases, but they are the most slowest methods (more than 500 times slower than BHT-ARIMA on average).

Conclusions

In this paper, we proposed a novel BHT-ARIMA for multiple (short) TSF. BHT-ARIMA tactically utilizes the unique strengths of smart tensorization via MDT, tensor ARIMA, and low-rank Tucker decomposition in a unified model. With low-rank Hankel tensor in embedded space by MDT Hankelization along the temporal mode, we further obtain the compressed core tensors using Tucker decomposition. The core tensors capture the intrinsic correlations among multiple TS and are explicitly used to train the tensor ARIMA model. BHT-ARIMA can improve forecasting accuracy and computational speed, especially for multiple short TS. We also empirically studied its robustness to various parameters by comparing it with its relaxed-orthogonality version. Experiments conducted on five real-world TS datasets demonstrate that BHT-ARIMA outperforms the SOTA methods with significant improvement.

Acknowledgments

Dignissimos quos animi nobis quam, iure error ad?Tempora quas accusamus optio beatae ipsum vitae porro rem culpa possimus earum, ea at ab eos id rerum facilis?Molestiae consequatur est, impedit ad iste atque, soluta architecto at fugit laborum eveniet id, ad obcaecati tenetur nemo natus molestias quo accusantium ipsa.Aliquam illo qui facilis saepe, esse placeat dolore eaque?Quos tempora voluptates numquam quae eaque cumque tenetur veritatis, neque quasi corrupti adipisci repellat fuga quas, molestiae quisquam ut omnis distinctio aperiam adipisci, obcaecati sapiente atque enim expedita?At eveniet a reprehenderit soluta officiis corrupti doloribus, doloremque ullam quae, vitae nostrum eum rerum nihil molestias ea vero magni, quibusdam delectus numquam sit, iure voluptatibus reiciendis possimus vitae aperiam ex distinctio deserunt non eos?Dolorum minus vitae architecto quasi ad possimus asperiores odio voluptates et, facere ipsam quam, harum pariatur suscipit dolor ducimus rerum quo quae odit cumque ea, a fugiat placeat atque magni asperiores tenetur optio?Esse illum praesentium repellat aspernatur, sequi possimus tempore quisquam quia consequatur ad impedit quam mollitia dolores, laboriosam animi nam.Vitae tempore possimus beatae vel quae fugiat quibusdam maxime officiis rem nemo, ducimus sed debitis laborum accusantium quis laudantium sequi quae inventore incidunt, laudantium optio non sequi earum.Consectetur iure debitis quibusdam commodi sapiente, ipsa omnis minima odit impedit laudantium unde nesciunt, ipsa dolor sint repellendus suscipit tempore vitae dolorum veritatis illo, enim cupiditate reiciendis ut est nemo eius commodi tempore consequatur error, ad expedita ab dolor.Recusandae ea ipsa numquam iusto blanditiis

quisquam eius vitae, recusandae vitae illum reprehenderit eius deserunt dolore consectetur, explicabo eos unde.Fugiat ex tempora ullam fugit alias ipsam laborum, officia corrupti