Factor Augmented Tensor-on-Tensor Neural Networks

Guanhao Zhou, Yuefeng Han, Xiufan Yu

Department of Applied and Computational Mathematics and Statistics
University of Notre Dame
gzhou4, yuefeng.han, xiufan.yu@nd.edu

Abstract

This paper studies the prediction task of tensor-on-tensor regression in which both covariates and responses are multi-dimensional arrays (a.k.a., tensors) across time with arbitrary tensor order and data dimension. Existing methods either focused on linear models without accounting for possibly nonlinear relationships between covariates and responses, or directly employed black-box deep learning algorithms that failed to utilize the inherent tensor structure. In this work, we propose a Factor Augmented Tensor-on-Tensor Neural Network (FATTNN) that integrates tensor factor models into deep neural networks. We begin with summarizing and extracting useful predictive information (represented by the "factor tensor") from the complex structured tensor covariates, and then proceed with the prediction task using the estimated factor tensor as input of a temporal convolutional neural network. The proposed methods effectively handle nonlinearity between complex data structures, and improve over traditional statistical models and conventional deep learning approaches in both prediction accuracy and computational cost. By leveraging tensor factor models, our proposed methods exploit the underlying latent factor structure to enhance the prediction, and in the meantime, drastically reduce the data dimensionality that speeds up the computation. The empirical performances of our proposed methods are demonstrated via simulation studies and real-world applications to three public datasets. Numerical results show that our proposed algorithms achieve substantial increases in prediction accuracy and significant reductions in computational time compared to benchmark methods.

Introduction

Motivated by a wide range of industrial and scientific applications, forecasting tasks using one tensor series to predict another have become increasingly prevalent in various fields. In finance, tensor models are used to forecast future stock prices by analyzing multi-dimensional stock attributes over time (Tran et al. 2017). In meteorology, tensor models predict future weather conditions using historical data formatted as multi-dimensional tensors (latitude, longitude, time, etc.) (Bilgin et al. 2021). In neuroscience, tensor models are employed to process medical imaging like MRI/fMRI scans (Wei et al. 2023). Though predictive modeling of tensors has emerged rapidly over the past decade, most existing studies primarily focus on scenarios when either covariates or responses are tensors, such as scalar-on-tensor regression (Guo, Kotsia, and Patras 2011; Zhou, Li, and Zhu 2013; Wimalawarne, Tomioka, and Sugiyama 2016; Li et al. 2018; Ahmed, Raja, and Bajwa 2020), matrix-on-tensor regression (Hoff 2015; Kossaifi et al. 2020), and tensor-on-vector regression (Li and Zhang 2017; Sun and Li 2017). **Tensor-on-tensor**[1] **prediction**, when both covariates and responses are tensors, remains a challenging task due to the inherent complexities of tensor structures.

One commonly used strategy for tensor-on-tensor modeling is to flatten the tensors into matrices (or even vectors) (Kilmer and Martin 2011; Choi and Vishwanathan 2014) so that matrix-based analytical tools become applicable. However, flattening can lead to a loss of spatial or temporal information, especially for datasets that are inherently dependent on data structures. For instance, in image processing, the relative positions of pixels carry important information about object shapes, which is lost when the image is flattened.

Alternatively, there is a handful of recent works on tensor-on-tensor regression models that do not adopt flattening but deal with the tensor structures directly (Lock 2018; Liu, Liu, and Zhu 2020; Gahrooei et al. 2021; Luo and Zhang 2024). However, these methods posit a linear prediction model between covariates and responses, limiting their capacity to capture potential nonlinear relationships.

Another emerging trend in tensor-on-tensor prediction involves neural networks, such as recurrent neural networks (RNN), convolutional neural networks (CNN), recurrent convolutional neural networks (RCNN), and temporal convolutional networks (TCN), to name a few. Deep neural network is a powerful tool in prediction tasks owing to its ability to learn intricate patterns from complex datasets. Structured with multiple layers of interconnected neurons, neural networks excel in capturing nonlinearity in data. Nevertheless, they often operate in a black-box manner and typically require extensive computational resources.

[1] Following the literature, we distinguish "tensor-on-tensor regression" from "tensor regression". "Tensor regression" refers to regression models with tensor covariates and scalar responses. "Tensor-on-tensor regression" refers to regression models with both tensor covariates and tensor responses. As a matter of fact, tensor-on-tensor regression encompasses tensor regression.

In this work, we develop a Factor Augmented Tensor-on-Tensor Neural Network (FATTNN) for forecasting a sequence of tensor responses using tensor covariates of arbitrary tensor order and data dimensions. One key innovation of our method is the integration of tensor factor models into deep neural networks for tensor-on-tensor regression. Factor models have been a widely utilized tool in matrix/vector data analysis (Bai and Ng 2002; Stock and Watson 2002; Pan and Yao 2008; Lam and Yao 2012) for understanding common dependence among multi-dimensional covariates. In recent years, researchers have advanced the methodology of factor models to the context of tensor time series (Chen et al. 2022; Chen, Yang, and Zhang 2022; Han et al. 2024a,b; Han, Chen, and Zhang 2022; Han and Zhang 2023; Chen, Han, and Yu 2024; Yu et al. 2024). In our models, we borrow the strengths from tensor factor models, offering a more general approach than traditional vector factor analysis for prediction (Sen, Yu, and Dhillon 2019; Fan, Xue, and Yao 2017; Yu, Yao, and Xue 2022).

Our contributions can be summarized in three folds.

- We propose a FATTNN model for tensor-on-tensor prediction that integrates tensor factor models into deep neural networks. Via a tensor factor model, FATTNN exploits the latent factor structures among the complex structured tensor covariates, and extracts useful predictive information for subsequent modeling. The utilization of the factor model reduces the data dimension drastically while preserving the tensorial data structures, contributing to substantial increases in prediction accuracy and significant reductions in computational time.
- In addition to the improved prediction accuracy and reduced computation cost, FATTNN offers several advantages over existing methods. First, it preserves the inherent tensor structure without any flattening or tensor contraction operations, preventing the possible loss of spatial or temporal information. Second, the utilization of neural networks equips FATTNN with exceptional capability in modeling nonlinearity in response-covariate relationships.
- FATTNN provides a comprehensive framework for the general supervised learning question of predicting tensor responses using tensor covariates. Although initially introduced under the problem settings of tensor-on-tensor time series forecasting, the proposed framework is not limited to time series data, but can accommodate a variety of data types. With different types of factor models and choices of neural network architectures tailored to specific characteristics of the data, FATTNN can be naturally adapted to tensor-on-tensor regression for various data types.

Related Work

Neural networks are powerful tools for processing complex tensor data. Early works involve RNN and its variations (e.g., Long Short-Term Memory (LSTM) networks) to accommodate the temporal dependence, and CNN to capture the temporal and spatial relationships underlying the data structures. Recently, more advanced architectures have been proposed, integrating the strengths of traditional models to capture the increasingly complex data dependence in a more powerful and efficient manner. Notable examples include convolutional LSTM (ConvLSTM) (Shi et al. 2015), recurrent CNN (RCNN) (Liang and Hu 2015), and temporal convolutional networks (TCN)(Bai, Kolter, and Koltun 2018).

Another popular thread is to incorporate a tensor decomposition procedure in the model training to reduce computational complexity and potentially enhance interpretability. Recent examples include tensorizing neural networks (Novikov et al. 2015), convolutional tensor-train LSTM (Conv-TT-LSTM) (Su et al. 2020), tensor regression networks (Kossaifi et al. 2020) and its efficient variant using tensor dropout (Kolbeinsson et al. 2021). Methods may vary depending on how the tensor decomposition is conducted (Fang et al. 2022; Wang and Zhe 2022; Tao, Tanaka, and Zhao 2024).

Preliminaries

Notation. Before proceeding, we first set up some notation. Let $\mathcal{X} \in \mathbb{R}^{d_1 \times \cdots \times d_K}$ denote a K-th order tensor of dimension $d_1 \times d_2 \times \cdots \times d_K$. Let $\mathcal{X}_{i_1,\ldots,i_K}$ denote the $(i_1, \ldots, i_K)$-th entry of the tensor $\mathcal{X}$ and $\mathcal{X}[\mathcal{I}_1, \ldots, \mathcal{I}_K]$ denotes the subtensor of $\mathcal{X}$ for $\mathcal{I}_1 \subseteq \{1, \ldots, d_1\}, \ldots, \mathcal{I}_K \subseteq \{1, \ldots, d_K\}$. The matricization operation $\text{mat}_k(\cdot)$ unfolds an order-K tensor along mode k to a matrix, say $\mathcal{X} \in \mathbb{R}^{d_1 \times \cdots \times d_K}$ to $\text{mat}_k(\mathcal{X}) \in \mathbb{R}^{d_k \times d_{-k}}$ where $d_k = \prod_{j \neq k} d_j$ and its detailed definition is provided in the appendix. The Frobenius norm of tensor $\mathcal{X}$ is defined as $\|\mathcal{X}\|_{\text{F}} = (\sum_{i_1,i_2,\cdots,i_K} \mathcal{X}^2_{i_1,\ldots,i_K})^{1/2}$. The Tucker rank of an order-K tensor $\mathcal{X}$, denoted by $\text{Tucrank}(\mathcal{X})$, is defined as a K-tuple $(r_1, ..., r_K)$, where $r_k = \text{rank}(\text{mat}_k(\mathcal{X}))$. Any Tucker rank-$(r_1, ..., r_K)$ tensor $\mathcal{X}$ admits the following Tucker decomposition (Tucker 1966): $\mathcal{X} = \mathcal{S} \times_1 U_1 \times_2 \cdots \times_K U_K$, where $\mathcal{S} \in \mathbb{R}^{r_1 \times \cdots \times r_K}$ is the core tensor and $U_k = \text{SVD}_{r_k}(\text{mat}_k(\mathcal{X}))$ is the mode-k top r_k left singular vectors. Here, the mode-k product of $\mathcal{X} \in \mathbb{R}^{d_1 \times \cdots \times d_K}$ with a matrix $B \in \mathbb{R}^{d_k \times r_k}$, denoted by $\mathcal{X} \times_k B$, is a $d_1 \times \cdots \times d_{k-1} \times r_k \times d_{k+1} \times \cdots \times d_K$-dimensional tensor, and its detailed definition is provided in Appendix A. The following abbreviations are used to denote the tensor-matrix product along multiple modes: $\mathcal{X} \times_{k=1}^{K} U_k = \mathcal{X} \times_1 U_1 \times_2 \cdots \times_K U_K$ and $\mathcal{X} \times_{\ell \neq k} U_\ell = \mathcal{X} \times_1 U_1 \times_2 \cdots \times_{k-1} U_{k-1} \times_{k+1} U_{k+1} \times_{k+2} \cdots \times_K U_K$. For any two matrices $A \in \mathbb{R}^{m_1 \times r_1}, B \in \mathbb{R}^{m_2 \times r_2}$, we denote the Kronecker product $\odot$ as $A \odot B \in \mathbb{R}^{m_1 m_2 \times r_1 r_2}$. For any two tensors $\mathcal{A} \in \mathbb{R}^{m_1 \times m_2 \times \cdots \times m_K}, \mathcal{B} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_N}$, we denote the tensor product $\otimes$ as $\mathcal{A} \otimes \mathcal{B} \in \mathbb{R}^{m_1 \times \cdots \times m_K \times r_1 \times \cdots \times r_N}$, such that $(\mathcal{A} \otimes \mathcal{B})_{i_1,\ldots,i_K,j_1,\ldots,j_N} = (\mathcal{A})_{i_1,\ldots,i_K} (\mathcal{B})_{j_1,\ldots,j_N}$.

Problem Setup. Suppose we observe a time series dataset comprising n temporal samples $\mathcal{D} = \{(\mathcal{X}_t, \mathcal{Y}_t), 1 \leq t \leq n\}$, where $\mathcal{X}_t \in \mathbb{R}^{d_1 \times \cdots \times d_K}$ is a tensor of covariates and $\mathcal{Y}_t \in \mathbb{R}^{p_1 \times \cdots \times p_q}$ is the tensor response variable. For some newly observed covariates $\{\mathcal{X}_t\}_{t=n+1}^{n+m}$, *the forecasting task of tensor-on-tensor time series regression is to accurately predict the corresponding future tensor responses* $\{\mathcal{Y}_t\}_{t=n+1}^{n+m}$, *given the observed time series* $\{(\mathcal{X}_t, \mathcal{Y}_t)\}_{t=1}^{n}$ *and the new covariates* $\{\mathcal{X}_t\}_{t=n+1}^{n+m}$. Here, m is the forecast horizon. In subsequent discussions, we denote the series in training time range as $\mathcal{X}^{(tr)} = \{\mathcal{X}_t\}_{t=1}^{n}$ and $\mathcal{Y}^{(tr)} = \{\mathcal{Y}_t\}_{t=1}^{n}$, and the covariates in forecasting time range as $\mathcal{X}^{(te)} = \{\mathcal{X}_t\}_{t=n+1}^{n+m}$.

Methodology: Factor Augmented Tensor-on-Tensor Time Series Regression

In this section, we introduce our Factor Augmented Tensor-on-Tensor Neural Network, dubbed as FATTNN. Our work explores a broad category of challenges known as tensor-on-tensor regression, which seeks to elucidate the relationships between covariates and responses that may manifest as scalars, vectors, matrices, or higher-order tensors. As illustrated in Figure 1, our method consists of two components. First, factor tensor features are extracted by low-rank tensor factorization. Second, we develop a Tensor-on-Tensor Neural Network based on Temporal Convolutional Network (TCN) (Bai, Kolter, and Koltun 2018). In the first subsection, we present a tensor factor model. This model can capture global patterns among the observed time series covariate tensors, by representing each of the original covariate tensor $\mathcal{X}_t$ as a multilinear combination of $r_1 \times \cdots \times r_K$ basis across each tensor mode k plus noise, where $r_k \ll d_k$. It offers a more general approach than the traditional vector factor analysis for prediction (Sen, Yu, and Dhillon 2019; Fan, Xue, and Yao 2017; Luo et al. 2022; Yu, Yao, and Xue 2022; Fan and Gu 2023), and preserves the tensor structures of features. It also provides dimension reduction, which significantly reduces computational complexity. In the second subsection, we describe how the output from the global tensor factor model can be used as an input covariate for a TCN to predict future response tensors.

Tensor Factor Model (TFM)

Our representation learning module utilizes low rank tensor factorization for covariate tensor. The idea is to factorize the training covariate tensors $\mathcal{X}_t \in \mathbb{R}^{d_1 \times \cdots \times d_K}, 1 \le t \le n$, into low dimensional tensors $\mathcal{F}_t \in \mathbb{R}^{r_1 \times \cdots \times r_K}$ and linear combination matrices (loading matrices) $A_k \in \mathbb{R}^{d_k \times r_k}$, where $r_k \ll d_k$. Specifically, the model can be expressed as

$$\mathcal{X}_t = \mathcal{F}_t \times_1 A_1 \times_2 \ldots \times_K A_K + \mathcal{E}_t, \tag{1}$$

where $\mathcal{E}_t$ is idiosyncratic noise of the tensor $\mathcal{X}_t$, and t is a time point. Here the k-mode product of $\mathcal{X} \in \mathbb{R}^{d_1 \times d_2 \times \ldots \times d_K}$ with a matrix $U \in \mathbb{R}^{d'_k \times d_k}$, denoted as $\mathcal{X} \times_k U$, is an order K-tensor of size $d_1 \times \ldots \times d_{k-1} \times d'_k \times d_{k+1} \times \ldots \times d_K$ such that $(\mathcal{X} \times_k U)_{i_1,\ldots,i_{k-1},j,i_{k+1},\ldots,i_K} = \sum_{i_k=1}^{d_k} \mathcal{X}_{i_1,i_2,\ldots,i_K} U_{j,i_k}$. If $K = 2$, the matrix factor model can be written as $\mathcal{X}_t = A_1 \mathcal{F}_t A_2^\top + \mathcal{E}_t$. The latent core tensor $\mathcal{F}_t$, which typically encapsulates the most critical information, generally possesses dimensions significantly smaller than those of $\mathcal{X}_t$. Thus, the covariate tensor $\mathcal{X}_t$ can be thought of to be comprised of a basis tensor features $\mathcal{F}_t$ that capture the global patterns in the whole data-set, and all the original covariates are multilinear combinations of these basis tensor features. In the next subsection, we will discuss how a TCN can be utilized to leverage the temporal structure of the training data.

For predicting future response tensors, given a new covariate tensor $\mathcal{X}_t$, we can also extract low-dimensional feature tensors $\mathcal{F}_t$ using the estimated deterministic loading matrices A_k and model (1). This approach efficiently captures the essential characteristics from the high-dimensional input data. The reduced dimensionality allows for more efficient data processing and analysis while retaining essential features.

To capture the underlying tensor features of the covariates, we employ a Time series Inner-Product Unfolding Procedure (TIPUP). It utilizes the linear temporal dependence among the covariates. The TIPUP method performs singular value decomposition (SVD) for each tensor mode $k = 1, ..., K$:

$$\widehat{A}_k = \text{LSVD}_{r_k} \left(\frac{1}{n} \sum_{t=1}^{n} \text{mat}_k(\mathcal{X}_t) \text{mat}_k^\top(\mathcal{X}_t) \right). \tag{2}$$

The estimation methods in (2) can be further enhanced through an iterative refinement algorithm, detailed in the appendix. The factor tensors in the training data thus can be estimated by $\widehat{\mathcal{F}}_t = \mathcal{X}_t \times_{k=1}^{K} \widehat{A}_k^\top$, for $1 \le t \le n$.

In the next proposition, we demonstrate the provable benefits of using a tensor factor model with Tucker low-rank structure (1). Specifically, we establish the convergence rates of the linear space of the loading matrices, which in turn ensure the accuracy of the estimated factor tensors.

Proposition 1. *Assume each elements of the idiosyncratic noise $\mathcal{E}_t$ in (1) are i.i.d. $N(0,1)$. The ranks $r_1, ..., r_K$ are fixed. The factor process $\mathcal{F}_t$ is weakly stationary and its cross-outer-product process is ergodic in the sense of $\frac{1}{n}\sum_{t=1}^{n} \mathcal{F}_t \otimes \mathcal{F}_t \to \mathbb{E}(\mathcal{F}_t \otimes \mathcal{F}_t)$ in probability, where the elements of $\mathbb{E}(\mathcal{F}_t \otimes \mathcal{F}_t)$ are all finite. In addition, the condition numbers of $A_k^\top A_k$ ($k = 1, ..., K$) are bounded. Let $\lambda = \prod_{k=1}^{K} \|A_k\|_2$ and grow as dimensions d_k increases. Then, the iterative TIPUP estimator satisfies*

$$\begin{aligned} &\|\widehat{A}_k(\widehat{A}_k^\top \widehat{A}_k)^{-1}\widehat{A}_k^\top - A_k(A_k^\top A_k)^{-1}A_k\|_2 \\ =&O_{\mathbb{P}}\left(\frac{\max_k \sqrt{d_k}}{\lambda\sqrt{n}} + \frac{\max_k \sqrt{d_k}}{\lambda^2\sqrt{n}}\right), \qquad 1 \le k \le K. \end{aligned}$$

Tensor-on-Tensor Neural Network based TCN

If we are equipped with a TCN that identifies the temporal patterns in the training dataset $(\mathcal{X}^{(tr)}, \mathcal{Y}^{(tr)})$, we can use this model to enhance the temporal structures in $(\mathcal{F}^{(tr)}, \mathcal{Y}^{(tr)})$. Thus, a TCN is applied to the input sequence of estimated factor tensors $\widehat{\mathcal{F}}_1, \ldots, \widehat{\mathcal{F}}_t$ along with the corresponding output sequence of response tensors $\mathcal{Y}_1, \ldots, \mathcal{Y}_t$ to encapsulate the temporal dynamics. Let $\eta(\cdot)$ represent this network. The whole temporal network $\eta(\cdot)$ can be trained using the low-rank factor tensors $\widehat{\mathcal{F}}_1, \ldots, \widehat{\mathcal{F}}_n$ derived from training dataset. In the literature, factor-augmented linear regression has been well studied recently in statistics, economics and machine learning (Stock and Watson 2002; Bai and Ng 2006; Bair et al. 2006; Fan, Xue, and Yao 2017; Bing et al. 2021).

The trained temporal network $\eta(\cdot)$ can be effectively utilized for multi-step look-ahead prediction in a standard approach. The factor tensors for future time steps can be estimated using $\widehat{\mathcal{F}}_{n+h} = \mathcal{X}_{n+h} \times_{k=1}^{K} \widehat{A}_k^\top$, where $h > 0$. Using the historical data points of factor tensors $\widehat{\mathcal{F}}_t, 1 \le t \le n+h-1$, the prediction for the subsequent time step, $\mathcal{Y}_{n+h}$, is generated by $\widehat{\mathcal{Y}}_{n+h} = \eta(\widehat{\mathcal{F}}_1, ..., \widehat{\mathcal{F}}_{n+h})$.

Our hybrid forecasting model incorporates a TCN that inputs past data points from the original raw response time

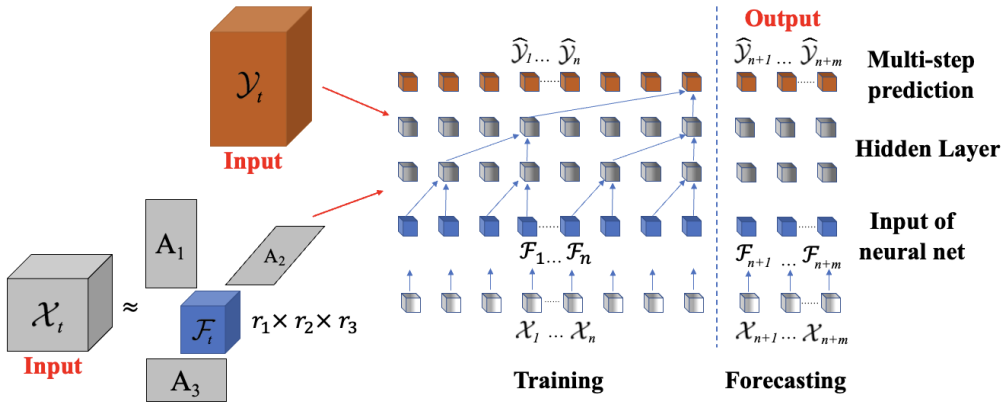

Figure 1: A graphical illustration of the proposed FATTNN. The input is the observed time series $\{(\mathcal{X}_t, \mathcal{Y}_t)\}_{t=1}^{n}$ and the new covariates $\{\mathcal{X}_t\}_{t=n+1}^{n+m}$. The output is the forecasted future tensor responses, denoted by $\{\widehat{\mathcal{Y}}_t\}_{t=n+1}^{n+m}$

series and the estimated factor tensors, enhancing prediction accuracy. The pseudo-code for our model is outlined in Algorithm 1. Due to space limitations, the determination of the rank of the factor process is discussed in Appendix A.

Algorithm 1: Factor Augmented Tensor-on-Tensor Neural Network (FATTNN)

Input: Observed tensor time series $(\mathcal{X}_1, \mathcal{Y}_1), \ldots, (\mathcal{X}_n, \mathcal{Y}_n), \mathcal{X}_{n+1}, \ldots, \mathcal{X}_{n+m}$, rank r_k for each mode k, forecasting horizon m.
Output: Forecasts $\widehat{\mathcal{Y}}_{n+1}, \ldots, \widehat{\mathcal{Y}}_{n+m}$.
Compute the factor tensor $\widehat{\mathcal{F}}_t$ for $t = 1, \ldots, n$, by TIPUP (2) or its iterative version in Algorithm S.1.
Fit a TCN $\eta(\cdot)$ to the input sequence of factor tensors $\widehat{\mathcal{F}}_1, \ldots, \widehat{\mathcal{F}}_n$ and the output sequence of response tensor $\mathcal{Y}_1, ..., \mathcal{Y}_n$ to capture the temporal dynamics.
Forecast $\mathcal{Y}_{n+1}, \ldots, \mathcal{Y}_{n+m}$ using the fitted TCN.
for $h = 1$ to m **do**
  $\widehat{\mathcal{F}}_{n+h} = \mathcal{X}_{n+h} \times_{k=1}^{K} \widehat{A}_k^\top$.
  $\widehat{\mathcal{Y}}_{n+h} = \eta(\widehat{\mathcal{F}}_1, \ldots, \widehat{\mathcal{F}}_{n+h})$.
end for
return $\widehat{\mathcal{Y}}_{n+1}, \ldots, \widehat{\mathcal{Y}}_{n+m}$.

Numerical Results

In this section, we investigate the finite-sample performance of our proposed FATTNN methods via simulation studies and real-world applications. Specifically, we implement the proposed **FATTNN**[2] with TIPUP algorithm to estimate the factor structures. For benchmark methods, we consider one traditional statistical model - the multiway tensor-on-tensor regression (Lock 2018) (denoted by **Multiway**), and four state-of-the-art deep learning approaches, including the temporal convolutional network (Bai, Kolter, and Koltun 2018) of $\mathcal{Y}$ regressed on $\mathcal{X}$ (denoted by **TCN**), the long short-term memory network (Hochreiter and Schmidhuber 1997) (denoted by **LSTM**) , the convolutional tensor-train LSTM (Su et al. 2020) (denoted by **Conv-TT-LSTM**), and tensor regression layer (Kossaifi et al. 2020) (denoted by **TRL**).

Evaluation Metrics. We evaluate the performance of various methods with emphasis on two aspects: prediction accuracy and computational efficiency. To evaluate prediction accuracy, we compute the Mean Squared Error (MSE) over the testing data, i.e., $\text{MSE} = (n_{\text{test}} \cdot p_1 \cdots p_q)^{-1} \sum_{i \in \mathcal{D}_{\text{test}}} \|\mathcal{Y}_i^{(\text{obs})} - \mathcal{Y}_i^{(\text{pred})}\|_F^2$, where $\mathcal{D}_{\text{test}}$ denotes the testing set, n_{test} is the number of samples in the testing set $\mathcal{D}_{\text{test}}$, and $(\mathcal{Y}_i^{(\text{obs})}, \mathcal{Y}_i^{(\text{pred})})$ are the observed and predicted values of the i-th tensor response. In addition, we record the computational time of different methods to evaluate computational efficiency.

Simulation Studies

Data Generating Process. We carry out simulation studies under different scenarios with various configurations of sample size, data dimensions, tensor ranks, and relationships between covariates and responses. Recall that $\mathcal{X}_t \in \mathbb{R}^{d_1 \times \cdots \times d_K}$, $\mathcal{F}_t \in \mathbb{R}^{r_1 \times \cdots \times r_K}$, $\mathcal{Y}_t \in \mathbb{R}^{p_1 \times \cdots \times p_q}$, for $1 \le t \le n$. We begin with generating the low-rank core tensor $\mathcal{F}_t$ from a tensor autoregressive model $\text{vec}(\mathcal{F}_t) = \Phi \cdot \text{vec}(\mathcal{F}_{t-1}) + \text{vec}(\mathcal{W}_t)$ for $t = 1, \ldots, n$. Here, $\text{vec}(\mathcal{F}_t)$ denotes the vectorization[3] of a tensor $\mathcal{F}_t$, $\mathcal{W}_t \in \mathbb{R}^{r_1 \times \cdots \times r_K}$ is an error tensor in which every element is randomly generated from a normal distribution. The coefficient matrix $\Phi \in \mathbb{R}^{(r_1 r_2 \cdots r_K) \times (r_1 r_2 \cdots r_K)}$ is constructed using the Kronecker product of a sequence of matrices Q_k, i.e., $\Phi = \odot_{k=1}^{K} Q_k$, in which each $Q_k \in \mathbb{R}^{r_k \times r_k}$ is a matrix with i.i.d. standard normal entries and orthonormalized through QR decomposition. The tensor time series $\mathcal{F}_t$ is randomly initiated in a distance past, and we discard the first 500 time points to stabilize the process.

Next, we generate the tensor covariates $\mathcal{X}_t$ from $\mathcal{F}_t$, following $\mathcal{X}_t = \lambda \mathcal{F}_t \times_1 A_1 \times_2 \ldots \times_K A_K + \mathcal{E}_t$, for $t = 1, \ldots n$, where λ is a scalar that controls the signal-to-noise ratio in

[2]Our code is available in the supplementary material.

[3]We'd like to clarify that our proposed FATTNN approach does not involve tensor vectorization in the model fitting. This vectorization is solely for the convenience of data generation in simulation.

the tensor factor model. In our experiments, we set $\lambda = (\prod_{k=1}^{K} r_k)^{1/2}$. The loading matrices $A_k \in \mathbb{R}^{d_k \times r_k}$ are generated with independent $N(0,1)$ entries, and then orthonormalized using QR decomposition. The error $\mathcal{E}_t \in \mathbb{R}^{d_1 \times \cdots \times d_K}$ is also generated with each element independently drawn from normal distributions.

Then, we generate the tensor responses $\mathcal{Y}_t$ from $\mathcal{F}_t$, using $\mathcal{Y}_t = \langle S(\mathcal{F}_t), \Lambda \rangle_L + \mathcal{U}_t$, $t = 1, \dots n$. Here $S(\mathcal{F}_t) := (s(\mathcal{F}_{t,i_1,\dots,i_K}))_{1 \le i_1 \le r_1, \dots, 1 \le i_K \le r_K} \in \mathbb{R}^{r_1 \times \cdots \times r_K}$ applies an element-wise transformation $s(\cdot)$ to each entry of $\mathcal{F}_t$. $\mathcal{U}_t \in \mathbb{R}^{p_1 \times \cdots \times p_q}$ is a noise matrix whose elements are i.i.d $N(0, \sigma_u^2)$. The coefficient tensor Λ is set as $\Lambda = [\![U_1, \cdots, U_K, V_1, \cdots V_d]\!] \in \mathbb{R}^{r_1 \times \cdots \times r_K \times p_1 \times \cdots \times p_q}$ where $U_k \in \mathbb{R}^{r_k \times R}$ for $k = 1, \cdots, K$ and $V_m \in \mathbb{R}^{p_m \times R}$ for $m = 1, \cdots, d$, each with independent $N(0,1)$ entries. In our experiments, we set $R = 6$. Details of the notation $[\![\cdot, \cdots, \cdot]\!]$ and $\langle \cdot, \cdot \rangle_L$ are provided in Equations (S.2) and (S.3).

We consider the following experimental configurations:
(1) $(d_1, d_2, d_3) = (25, 25, 12)$,$(r_1, r_2, r_3) = (3, 3, 2)$, $(p_1, p_2, p_3) = (6, 8, 6)$, $n = 500$, $s(z) = cos(z)$, $\sigma_u^2 = 1$.
(2) $(d_1, d_2, d_3) = (30, 6, 12)$, $(r_1, r_2, r_3) = (6, 3, 2)$, $(p_1, p_2, p_3) = (8, 6, 4)$, $n = 400$, $s(z) = log(|z|)$, $\sigma_u^2 = 1$.
(3) $(d_1, d_2, d_3) = (12, 3, 12)$, $(r_1, r_2, r_3) = (4, 3, 4)$, $(p_1, p_2, p_3) = (3, 3, 3)$, $n = 100$, $s(z) = log(e^{(z)} + 1)$, $\sigma_u^2 = 0.5$.

Simulation Evaluations. In each setting, we use 70% for training the models and 30% for model evaluation. Comparisons of MSE, and computational time are summarized in Table 1. All reported metrics are averaged over 20 replications. Experiments were run on the research computing cluster provided by the University Center for Research Computing. From Table 1, we can see that FATTNN consistently improves over the rest benchmark methods by a large margin. The improvements can be attributed to two facets: (a) the utilization of tensor factors, summarizing useful predictive information effectively and reducing the number of noise covariates, and (b) the utilization of neural networks, possessing extraordinary ability in capturing complex relationships in the data and therefore exhibiting exceptional predictive power. On a separate note, Table 1 reveals the evident computational advantage of FATTNN compared to other neural network models such as TCN, owing to the effective dimension reduction through tensor factor models.

Prediction Task	Simulation 1	Simulation 2	Simulation 3
	Test MSE		
TCN	9.812 (3.13)	670.5 (25.8)	1074 (32.6)
FATTNN	**6.353** (2.52)	**511.9** (22.5)	**696.7** (26.4)
MultiWay	12.31 (3.51)	805.1 (28.4)	1670 (40.3)
LSTM	10.48 (3.23)	878.3 (29.6)	807.8 (28.3)
TRL	12.29 (3.50)	852.6 (29.0)	2003 (38.4)
Conv-TT-LSTM	10.32 (3.15)	1643 (39.5)	2812 (52.8)
	Computational Time (hh:mm:ss)		
TCN	01:22:02	01:26:30	00:11:16
FATTNN	00:16:18	00:17:54	00:03:41
MultiWay	00:20:15	00:12:30	00:01:03
LSTM	00:03:36	00:03:01	00:02:09
TRL	00:02:25	00:01:30	00:00:50
Conv-TT-LSTM	03:02:15	05:31:25	00:35:37

Table 1: Comparisons of prediction accuracy (measured by MSE over the testing data) and computational time (in the format of hh:mm:ss) in simulation studies

Real Data Examples

We evaluate the performance of different methods using five prediction tasks on three real-world datasets. The information about sample sizes in the datasets, data dimensions of covariates $(d_1, \dots, d_K)$ and responses $(p_1, \dots, p_q)$, and the ranks of core tensors $(r_1, \dots, r_K)$ utilized in the analyses are summarized in Table 2. For the prediction tasks using the New York Taxi data and FMRI image datasets, we split the data into training sets (70%) and testing (30%). For the prediction tasks using the FAO dataset, we use 80% data for training and 20% for testing due to its relatively small sample size. We compare the prediction accuracy and computation time of six methods illustrated in the simulation section. The numerical prediction results along with the corresponding computational time are summarized in Table 3.

(1) The United Nations Food and Agriculture Organization (FAO) Crops and Livestock Products Data. The database provides agricultural statistics (including crop, livestock, and forestry sub-sectors) collected from countries and territories since 1961. It is publicly available at https://www.fao.org. Our analyses focus on 11 crops and 5 livestock products from 46 countries in East Asia, North America, South America, and Europe from 1961 to 2022. Detailed information about the types of crops, livestock, and countries are presented in Table S.1 of Appendix B. We study two prediction tasks: (i) using Yield and Production data of 11 different crops for 33 countries in East Asia, North America, and Europe (i.e., $\mathcal{X}_t \in \mathbb{R}^{33 \times 2 \times 11}$) to predict the Yield and Production quantities of the same crops for 13 selected countries in South America (i.e., $\mathcal{Y}_t \in \mathbb{R}^{13 \times 2 \times 11}$); and (ii) using four agricultural statistics (Producing Animals, Animals-slaughtered, Milk Animals, Laying) associated with 5 kinds of livestock of 26 selected countries in Europe (i.e., $\mathcal{X}_t \in \mathbb{R}^{26 \times 4 \times 5}$) to predict three metrics (Area-harvested, Production, and Yield) of 11 crops in the same countries (i.e., $\mathcal{Y}_t \in \mathbb{R}^{26 \times 3 \times 11}$). Observing high variability in the raw data, we adopt a log transformation on the raw series and fit our models using log-transformed data.

Table 3 shows that FATTNN brings in substantial reductions in MSE compared to other methods. In task (i), FATTNN yields **47.3%, 39.5%, 58.3%, 52.8%, and 86.1% reductions in MSE compared to TCN, Multiway, TRL, LSTM, and Conv-TT-LSTM**, respectively, and in task (ii), the improvements are **33.4%, 54.9%, 67.3%, 36.7%, and 68.9% reductions in MSE**, respectively. The table also shows that the upper bound of the bootstrap CI for FATTNN is smaller than the lower bounds of the CIs for almost all the benchmarks, indicating FATTNN possesses a statistically significant increase in prediction accuracy. What's more, FATTNN produces the shortest CI, implying the lowest variability among the methods compared. As reflected by the table, FATTNN **computes much faster than TCN, about 3.6 times faster in task (i) and 2 times faster in task (ii)**. We also provide graphical comparisons between ground truth and predictions from different methods; see Figure 2. The plots confirm that FATTNN yields results that most closely

Prediction Task	Sample Size	Dimension of $\mathcal{X}_t$	Dimension of $\mathcal{Y}_t$	Rank of $\mathcal{F}_t$
FAO crop $\sim$ crop	62	(33,2,11)	(13,2,11)	(6,2,4)
FAO crop $\sim$ livestock	62	(26,4,5)	(26,3,11)	(6,4,2)
NYC taxi – midtown	504	(2,12,12,8)	(12,12,8)	(2,4,4,2)
NYC taxi – downtown	504	(2,19,19,8)	(19,19,8)	(2,4,4,2)
FMRI data	1452	(25,64,64)	(1,64,64)	(8,23,23)

Table 2: Data information of the real-world applications

Prediction Task	TCN	FATTNN	Multiway	TRL	LSTM	Conv-TT-LSTM
			Test MSE			
FAO crop $\sim$ crop	14.11 [10.83, 17.58]	**7.437** [6.18, 11.44]	12.29 [11.86, 32.45]	17.84 [15.89, 22.74]	15.76 [13.16, 18.34]	53.39 [52.41, 63.95]
FAO crop $\sim$ livestock	32.53 [24.87, 36.10]	**21.65** [19.56, 25.10]	48.04 [45.56, 165.80]	66.25 [56.85, 76.13]	34.23 [30.47, 38.48]	69.61 [69.06, 78.44]
NYC taxi – midtown	12.87 [12.06, 15.52]	**7.399** [6.53, 8.03]	22.84 [20.86, 44.79]	8.565 [7.89, 10.12]	8.404 [7.95, 9.63]	25.80 [21.55, 28.62]
NYC taxi – downtown	14.58 [13.36, 17.42]	**9.493** [8.46, 11.03]	28.10 [23.99, 52.12]	12.43 [10.82, 14.50]	12.45 [11.09, 14.38]	42.18 [36.27, 47.66]
FMRI data	0.1236 [0.0997, 0.148]	0.06634 [0.053, 0.075]	0.1397 [0.135, 0.158]	0.07902 [0.072, 0.084]	0.1964 [0.163, 0.224]	**0.03871** [0.0337, 0.0914]
			Computational Time (hh:mm:ss)			
FAO crop $\sim$ crop	00:55:33	00:15:28	00:03:18	00:09:41	00:02:08	00:20:07
FAO crop $\sim$ livestock	00:24:34	00:12:57	00:03:21	00:07:53	00:01:55	00:16:18
NYC taxi – midtown	01:57:15	00:34:34	00:04:52	00:19:26	00:05:01	01:54:28
NYC taxi – downtown	03:25:55	00:32:46	00:25:35	00:22:09	00:05:34	03:07:05
FMRI data	03:54:50	00:20:58	10:01:20	00:43:51	00:06:27	17:27:05

Note: Numbers in the square brackets are bootstrap confidence intervals (CIs) of test MSE over 100 bootstrapping replications.

Table 3: Comparisons of prediction accuracy (measured by MSE over the testing data) and computational time (in the format of hh:mm:ss) of different methods in real-world applications

align with the ground-truth patterns.

(2) New York City (NYC) Taxi Trip Data. The data contains 24-hour taxi pick-up and drop-off information of 69 areas in New York City for all the business days in 2014 and 2015. It is publicly available at https://www.nyc.gov. Considering the autoregressive nature of this taxi data, we include the lag-1 response as an additional covariate when fitting the models. We focus on two prediction tasks: using the pick-up and drop-off data from 6:00-14:00 to predict the pick-up and drop-off outcomes from 14:00-22:00 in 12 districts in Midtown Manhattan and 19 districts in Downtown Manhattan, respectively. A detailed Manhattan district map is provided in Figure S.3 of Appendix B. The Midtown prediction task yields a tensor covariate and response $(\mathcal{X}_t, \mathcal{Y}_t) \in \mathbb{R}^{2\times12\times12\times8} \times \mathbb{R}^{12\times12\times8}$ for each business day, in which the 1st dimension stands for the observed information at time t, concatenated with lag 1 response, the 2nd and 3rd dimensions encode the pick-up and drop-off districts, and the 4th dimension represents the hour of day. Similarly, the Downtown prediction task involves $(\mathcal{X}_t, \mathcal{Y}_t) \in \mathbb{R}^{2\times19\times19\times8} \times \mathbb{R}^{19\times19\times8}$for each business day.

From Table 3, we observe FATTNN significantly outperforms all other approaches in terms of prediction accuracy, with **42.5%, 67.6%, 13.6%, 12.0%, and 71.3% reductions in MSE compared to TCN, Multiway, TRL, LSTM, and Conv-TT-LSTM**, respectively, in Midtown traffic prediction. In Downtown traffic prediction, FATTNN shows 34.9%, 66.2%, 23.6%, 23.7%, and 77.5% reductions in MSE compared to TCN, Multiway, TRL, LSTM, and Conv-TT-LSTM, respectively. As shown in the table, FATTNN is much more computationally efficient than TCN which directly feeds tensor observations into a convolutional network, about 6.4 times faster than TCN in Downtown prediction and 3.4 times faster in Midtown prediction. Figure 3 visualizes the comparisons using a 5-day moving average of ground-truth values and predicted pick-up and drop-off volumes by day (where we sum the number of pick-up and drop-off across 8-hour testing periods). FATTNN-predicted lines appear to be closer to the ground-truth lines than TCN-predicted lines in most plots.

(3) Functional Magnetic Resonance Imaging (FMRI) Data. We consider the Haxby dataset (Haxby et al. 2001), a well-known public dataset for research in brain imaging and cognitive neuroscience, which can be retrieved from the "NiLearn" Python library. The data contains slices of 64×64 FMRI brain images collected from 1452 samples. We use the first 25 slices of each sample to predict the 26th slice of that sample, i.e., $(\mathcal{X}_t, \mathcal{Y}_t) \in \mathbb{R}^{25\times64\times64} \times \mathbb{R}^{1\times64\times64}$.

Conv-TT-LSTM yields the lowest MSE in this task. This is not surprising as Conv-TT-LSTM is specifically designed for image prediction while FATTNN is more suitable for time-series tensor prediction. Other than Conv-TT-LSTM, FATTNN still outperforms other methods. Additionally, though Conv-TT-LSTM has the best prediction accuracy, it is quite computationally heavy - about 50 times slower than FATTNN. The FATTNN is also 2 times faster than the TRL method which has the 3rd lowest MSE in this task.

In summary, our proposed FATTNN performs the best in the four prediction tasks on FAO and NYC datasets. In the FMRI dataset, the FATTNN yields higher MSE than Conv-TT-LSTM, a method specialized in image prediction. That being said, FATTNN has a huge computational advantage over Conv-TT-LSTM. We also observe that Conv-TT-LSTM underperforms other methods in the FAO and NYC datasets by a large margin, when the prediction tasks are not focusing on image data. To summarize, the FATTNN demonstrates the overall best performance on all the prediction tasks, and strikes an ideal balance between prediction accuracy and computational cost. The FATTNN achieves substantial increases

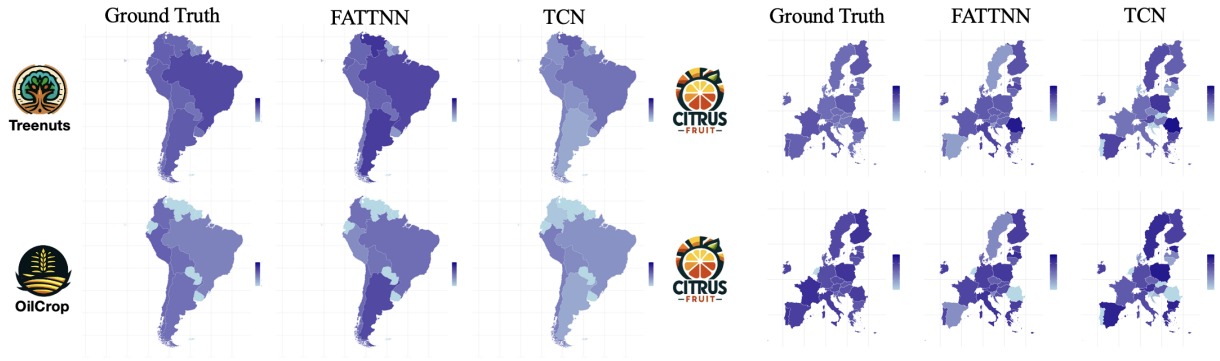

Figure 2: Left Panel: Comparisons between ground-truth values and predicted Production of Treenuts (top row) and Yield of Oilcrops (bottom row) in South America using the crop data of 33 countries in East Asia, North America, and Europe; Right Panel: Comparisons between ground-truth values and predicted Area-harvested (top row) and Yield (bottom row) of citrus fruit in 26 selected countries in Europe using livestock data of the same 26 countries. Values are plotted on the log-transformed scale. Numerical values are tabulated in Appendix B. In each panel, from left to right: Ground truth, FATTNN, and TCN.

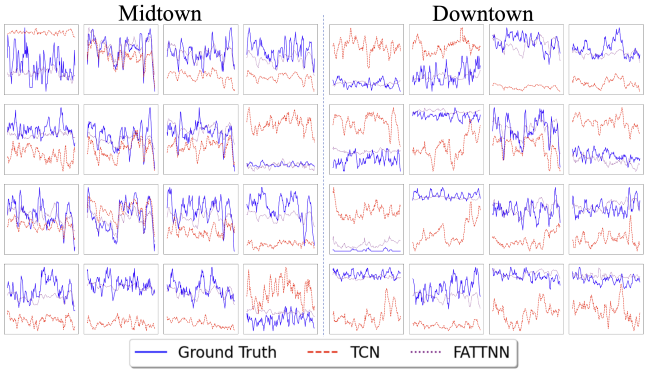

Figure 3: Comparisons between ground-truth values and predicted pick-up and drop-off volumes using various methods. "District A to B" denotes the traffic volume that passengers were picked up in District A and dropped off in District B. The district numbering is assigned according to the Manhattan district map shown in Figure S.3. An enlarged figure and more detailed descriptions are in Appendix B. Left: Midtown Manhattan; Right: Downtown Manhattan.

in prediction accuracy compared to benchmark methods. At the same time, the computational costs of FATTNN methods are much lower compared to TCN and Conv-TT-LSTM, ranging from 2 to 50 times faster.

Summary & Discussion

In this paper, we propose FATTNN, a hybrid method that integrates a tensor factor model into deep neural networks, for tensor-on-tensor time series forecasting. The key advantage of FATTNN is its ability to exploit and utilize the underlying tensor structure among the covariate tensors. Through a low-rank tensor factor representation, we preserve the tensor structures and, in the meantime, drastically reduce the data dimensionality, leading to enhanced prediction results and reduced computational costs. This approach unifies the strengths of both factor models and neural networks. Numerical studies confirm that our proposed methods successfully *achieve substantial increases in prediction accuracy and significant reductions in computational time*, compared to traditional linear tensor-on-tensor regression models (which often fail to capture the intricate nonlinearity in covariate-response relationships), conventional deep learning methods (that often operate in a black-box manner without making use of the inherent structures among observations), and state-of-the-art tensor-decomposition-based learning methods.

Limitation and Extension. Though we presented our models for the purpose of tensor-on-tensor time series forecasting, *our proposed FATTNN framework is not confined to temporal data but can accommodate a variety of data types*. The main idea of FATTNN is to use a low-rank tensor factor model to capture the intrinsic patterns among the observed covariates, and then proceed with a neural network to model the intricate relationships between covariates and responses. When handling time series tensors, we adopt TIPUP factorization and TCN to fully capitalize on their temporal nature. For non-temporal data, the TIPUP-TCN-based FATTNN is still applicable for tensor-on-tensor prediction tasks, but the strengths of TIPUP and TCN may not be fully utilized. For higher-order tensors (e.g. $K > 4$), Tensor-Train (Oseledets 2011) and Hierarchical Tucker decompositions (Lubich et al. 2013) may be better suited. This limitation can be easily rectified: with appropriate choices of factor models and neural network architectures, the proposed FATTNN can be naturally adapted to any simple (e.g., i.i.d observations) or complex tensor-type data (e.g., image, network, text, video). *To extend beyond time series data, different types of factor models and neural network architectures should be deployed, depending on the nature of the data.* For example, for graph data, Graph Neural Network is a more suitable alternative to TCN. For text data, Transformer models could substitute for the TCN in our FATTNN framework. An extended discussion on the generality of the FATTNN framework is presented in Appendix C. Our proposed FATTNN is flexible in accommodating different types of tensor factor models and deep learning architectures for tensor-on-tensor prediction using diverse data types.

Acknowledgments

We thank the anonymous reviewers for helpful suggestions. The work of Yuefeng Han is supported in part by National Science Foundation Grants DMS-2412578. The work of Xiufan Yu is supported in part by National Institutes of Health grant R01GM152812.

References

Ahmed, T.; Raja, H.; and Bajwa, W. U. 2020. Tensor regression using low-rank and sparse Tucker decompositions. *SIAM Journal on Mathematics of Data Science*, 2(4): 944–966.

Bai, J.; and Ng, S. 2002. Determining the number of factors in approximate factor models. *Econometrica*, 70(1): 191–221.

Bai, J.; and Ng, S. 2006. Confidence intervals for diffusion index forecasts and inference for factor-augmented regressions. *Econometrica*, 74(4): 1133–1150.

Bai, S.; Kolter, J. Z.; and Koltun, V. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Bair, E.; Hastie, T.; Paul, D.; and Tibshirani, R. 2006. Prediction by supervised principal components. *Journal of the American Statistical Association*, 101(473): 119–137.

Bilgin, O.; Maka, P.; Vergutz, T.; and Mehrkanoon, S. 2021. TENT: Tensorized encoder transformer for temperature forecasting. *arXiv preprint arXiv:2106.14742*.

Bing, X.; Bunea, F.; Strimas-Mackey, S.; and Wegkamp, M. 2021. Prediction under latent factor regression: Adaptive pcr, interpolating predictors and beyond. *Journal of Machine Learning Research*, 22(177): 1–50.

Chen, B.; Han, Y.; and Yu, Q. 2024. Estimation and Inference for CP Tensor Factor Models. *arXiv preprint arXiv:2406.17278*.

Chen, R.; Han, Y.; Li, Z.; Xiao, H.; Yang, D.; and Yu, R. 2022. Analysis of tensor time series: tensorts. *Journal of Statistical Software*.

Chen, R.; Yang, D.; and Zhang, C.-H. 2022. Factor models for high-dimensional tensor time series. *Journal of the American Statistical Association*, 117(537): 94–116.

Choi, J. H.; and Vishwanathan, S. 2014. DFacTo: Distributed factorization of tensors. *Advances in Neural Information Processing Systems*, 27.

Fan, J.; and Gu, Y. 2023. Factor augmented sparse throughput deep relu neural networks for high dimensional regression. *Journal of the American Statistical Association*, 1–15.

Fan, J.; Xue, L.; and Yao, J. 2017. Sufficient forecasting using factor models. *Journal of Econometrics*, 201(2): 292–306.

Fang, S.; Narayan, A.; Kirby, R.; and Zhe, S. 2022. Bayesian continuous-time Tucker decomposition. In *International Conference on Machine Learning*, 6235–6245. PMLR.

Gahrooei, M. R.; Yan, H.; Paynabar, K.; and Shi, J. 2021. Multiple tensor-on-tensor regression: An approach for modeling processes with heterogeneous sources of data. *Technometrics*, 63(2): 147–159.

Guo, W.; Kotsia, I.; and Patras, I. 2011. Tensor learning for regression. *IEEE Transactions on Image Processing*, 21(2): 816–827.

Han, Y.; Chen, R.; Yang, D.; and Zhang, C.-H. 2024a. Tensor factor model estimation by iterative projection. *The Annals of Statistics*, 52(6): 2641–2667.

Han, Y.; Chen, R.; and Zhang, C.-H. 2022. Rank determination in tensor factor model. *Electronic Journal of Statistics*, 16(1): 1726–1803.

Han, Y.; Yang, D.; Zhang, C.-H.; and Chen, R. 2024b. CP factor model for dynamic tensors. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 86(5): 1383–1413.

Han, Y.; and Zhang, C.-H. 2023. Tensor principal component analysis in high dimensional CP models. *IEEE Transactions on Information Theory*, 69(2): 1147–1167.

Haxby, J. V.; Gobbini, M. I.; Furey, M. L.; Ishai, A.; Schouten, J. L.; and Pietrini, P. 2001. Distributed and overlapping representations of faces and objects in ventral temporal cortex. *Science*, 293(5539): 2425–2430.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation*, 9(8): 1735–1780.

Hoff, P. D. 2015. Multilinear tensor regression for longitudinal relational data. *The Annals of Applied Statistics*, 9(3): 1169.

Kilmer, M. E.; and Martin, C. D. 2011. Factorization strategies for third-order tensors. *Linear Algebra and its Applications*, 435(3): 641–658.

Kolbeinsson, A.; Kossaifi, J.; Panagakis, Y.; Bulat, A.; Anandkumar, A.; Tzoulaki, I.; and Matthews, P. M. 2021. Tensor dropout for robust learning. *IEEE Journal of Selected Topics in Signal Processing*, 15(3): 630–640.

Kossaifi, J.; Lipton, Z. C.; Kolbeinsson, A.; Khanna, A.; Furlanello, T.; and Anandkumar, A. 2020. Tensor regression networks. *Journal of Machine Learning Research*, 21(123): 1–21.

Lam, C.; and Yao, Q. 2012. Factor modeling for high-dimensional time series: inference for the number of factors. *The Annals of Statistics*, 40(2): 694–726.

Li, L.; and Zhang, X. 2017. Parsimonious tensor response regression. *Journal of the American Statistical Association*, 112(519): 1131–1146.

Li, X.; Xu, D.; Zhou, H.; and Li, L. 2018. Tucker tensor regression and neuroimaging analysis. *Statistics in Biosciences*, 10(3): 520–545.

Liang, M.; and Hu, X. 2015. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3367–3375.

Liu, Y.; Liu, J.; and Zhu, C. 2020. Low-rank tensor train coefficient array estimation for tensor-on-tensor regression. *IEEE Transactions on Neural Networks and Learning Systems*, 31(12): 5402–5411.

Lock, E. F. 2018. Tensor-on-tensor regression. *Journal of Computational and Graphical Statistics*, 27(3): 638–647.

Lubich, C.; Rohwedder, T.; Schneider, R.; and Vandereycken, B. 2013. Dynamical approximation by hierarchical Tucker and tensor-train tensors. *SIAM Journal on Matrix Analysis and Applications*, 34(2): 470–494.

Luo, W.; Xue, L.; Yao, J.; and Yu, X. 2022. Inverse moment methods for sufficient forecasting using high-dimensional predictors. *Biometrika*, 109(2): 473–487.

Luo, Y.; and Zhang, A. R. 2024. Tensor-on-tensor regression: Riemannian optimization, over-parameterization, statistical-computational gap, and their interplay. *The Annals of Statistics*.

Novikov, A.; Podoprikhin, D.; Osokin, A.; and Vetrov, D. P. 2015. Tensorizing neural networks. *Advances in Neural Information Processing Systems*, 28.

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

Pan, J.; and Yao, Q. 2008. Modelling multiple time series via common factors. *Biometrika*, 95(2): 365–379.

Sen, R.; Yu, H.-F.; and Dhillon, I. S. 2019. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. *Advances in Neural Information Processing Systems*, 32.

Shi, X.; Chen, Z.; Wang, H.; Yeung, D.-Y.; Wong, W.-K.; and Woo, W.-c. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Advances in Neural Information Processing Systems*, 28.

Stock, J. H.; and Watson, M. W. 2002. Forecasting using principal components from a large number of predictors. *Journal of the American Statistical Association*, 97(460): 1167–1179.

Su, J.; Byeon, W.; Kossaifi, J.; Huang, F.; Kautz, J.; and Anandkumar, A. 2020. Convolutional tensor-train LSTM for spatio-temporal learning. *Advances in Neural Information Processing Systems*, 33: 13714–13726.

Sun, W. W.; and Li, L. 2017. Store: sparse tensor response regression and neuroimaging analysis. *Journal of Machine Learning Research*, 18(135): 1–37.

Tao, Z.; Tanaka, T.; and Zhao, Q. 2024. Undirected probabilistic model for tensor decomposition. *Advances in Neural Information Processing Systems*, 36.

Tran, D. T.; Magris, M.; Kanniainen, J.; Gabbouj, M.; and Iosifidis, A. 2017. Tensor representation in high-frequency financial data for price change prediction. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, 1–7. IEEE.

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

Wang, Z.; and Zhe, S. 2022. Nonparametric factor trajectory learning for dynamic tensor decomposition. In *International Conference on Machine Learning*, 23459–23469. PMLR.

Wei, B.; Peng, L.; Guo, Y.; Manatunga, A.; and Stevens, J. 2023. Tensor response quantile regression with neuroimaging data. *Biometrics*, 79(3): 1947–1958.

Wimalawarne, K.; Tomioka, R.; and Sugiyama, M. 2016. Theoretical and experimental analyses of tensor-based regression and classification. *Neural Computation*, 28(4): 686–715.

Yu, R.; Chen, R.; Xiao, H.; and Han, Y. 2024. Dynamic matrix factor models for high dimensional time series. *arXiv preprint arXiv:2407.05624*.

Yu, X.; Yao, J.; and Xue, L. 2022. Nonparametric estimation and conformal inference of the sufficient forecasting with a diverging number of factors. *Journal of Business & Economic Statistics*, 40(1): 342–354.

Zhou, H.; Li, L.; and Zhu, H. 2013. Tensor regression with applications in neuroimaging data analysis. *Journal of the American Statistical Association*, 108(502): 540–552.