# Multi-Antenna Channel Interpolation via Tucker Decomposed Extreme Learning Machine

Han Zhang, *Student Member, IEEE*, Bo Ai, *Senior Member, IEEE*, Wenjun Xu, *Senior Member, IEEE*, Li Xu, *Member, IEEE*, and Shuguang Cui, *Fellow, IEEE*

**_Abstract_—Channel interpolation is an essential technique for providing high-accuracy estimation of the channel state information for wireless systems design where the frequency-space structural correlations of multi-antenna channel are typically hidden in matrix or tensor forms. In this correspondence paper, a modified extreme learning machine (ELM) that can process tensorial data, or ELM model with tensorial inputs (TELM), is proposed to handle the channel interpolation task. The TELM inherits many good properties from ELMs. Based on the TELM, the Tucker decomposed extreme learning machine is proposed for further improving the performance. Furthermore, we establish a theoretical argument to measure the interpolation capability of the proposed learning machines. Experimental results verify that our proposed learning machines can achieve comparable mean squared error (MSE) performance against the traditional ELMs but with 15% shorter running time, and outperform the other methods for a 20% margin measured in MSE for channel interpolation.**

**_Index Terms_—Channel interpolation, extreme learning machine, tensor decomposition.**

## I. Introduction

High-quality channel estimation is crucial for many wireless applications, which is resource demanding in both time and frequency, where channel interpolation [1] and prediction [2] techniques are widely adopted to improve the estimation accuracy of channel state information (CSI). Meanwhile, using machine learning methods to tackle non-traditional problems in communications has become a new technology trend [3], [4]. Recently, as an innovative and efficient branch of the model-free machine learning methods, extreme learning machine (ELM) has gathered much interest from researchers in diversified areas. Owing to the unique properties such as fast training, solution uniqueness, and good generalization ability, ELM is promising to handle the channel interpolation tasks. However, the standard ELM was originally proposed to process vectorized data [5], which is not directly applicable to address the channel interpolation problem of multiple-input multiple-output (MIMO) channels. Specifically, MIMO channels exhibit frequency-space correlations, which are often recorded in the form of matrix or tensor, for which direct vectorization will lead to information loss. There have been attempts that tackles the channel interpolation problem with a tensor structure as in [6], a tensor filter is adopted for a 2D interpolation task. In [7], the MMSE method has been extended for similar task using tensor technique. To better resolve the high dimensional channel interpolation problem, a novel tensor based ELM, which is capable of handling tensorial inputs and learning through the CSI in MIMO channels, is needed.

In general, there have been great efforts in adapting ELM to tensorial inputs by applying certain matrix/tensor decomposition techniques [8], which are usually empirical. In this paper, we propose a novel ELM model with tensorial inputs (TELM) to extend the traditional ELM models for tensorial contexts while retaining the valuable ELM features. Moreover, we further propose a Tucker decomposed extreme learning machine (TDELM) based on the Tucker decomposition method [12] to reduce the computational complexity and establish a theoretical argument for its interpolation capability accordingly. Experimental results verify that our proposed methods can achieve comparable performance against the traditional ELMs but with reduced complexity, and outperform the other methods.

The remainder of this paper is organized as follows. Section II reviews the background of single-layer feedforward neural networks (SLFNs), traditional ELM, tensor operations and Tucker decomposition. Section III presents the proposed TELM and TDELM models and discusses how they will be applied to channel interpolation, Section IV investigates the properties of the considered models, and Section V demonstrates the experimental results. Finally, Section VI concludes the paper.

## II. Preliminaries

### A. Single-layer Feedforward Networks With Vector Inputs

Consider a dataset with $N$ data samples $(\mathrm{x}_i, t_i)$ for $i = 1, 2, \ldots, N$, where $\mathrm{x}_i \in \mathbb{R}^M$ is the feature vector of the $i$-th sample and $t_i$ is its label. Assume that an SLFN [5] contains $M$ input neurons and $L$ hidden neurons. The prediction $o_i$ of the label $t_i$ can be formulated as $o_i = \sum_{j=1}^{L} \beta_j \sigma(\mathrm{w}_j^T \mathrm{x}_i + b_j)$, where $\mathbf{W} = (\mathrm{w}_1, \mathrm{w}_2, \ldots, \mathrm{w}_L)$ is the weight matrix, whose $(i, j)$-th entry $w_{i,j}$ is the weight between the $i$-th input neuron and the $j$-th hidden neuron; $\mathrm{b} = (b_1, b_2, \ldots, b_L)^T$ is the bias vector from the input layer to the $j$-th hidden neuron; $\beta = (\beta_1, \beta_2, \ldots, \beta_L)^T$ is the weight vector between the hidden layer and the output neuron; and $\sigma$ is the sigmoid activation function, defined as $\sigma(x) \triangleq 1/(1 + e^{-x})$. In this setting, the bias between the hidden layer and the output layer is omitted. We then aim to solve the following optimization problem

$$\min_{\mathbf{W}, \mathrm{b}, \beta} \; f(\mathbf{W}, \mathrm{b}, \beta) = ||\mathbf{T} - \mathbf{O}||_2^2 = \sum_{i=1}^{N} (t_i - o_i)^2, \qquad (1)$$

where $\mathbf{T} = (t_1, t_2, \ldots, t_N)^T$ and $\mathbf{O} = (o_1, o_2, \ldots, o_N)^T$ are the label vector and its prediction vector, respectively. A typical algorithm finds the optimal values of $\mathbf{W}$, $\mathrm{b}$ and $\beta$ by propagating the errors backwards utilizing gradient or sub-gradient descent methods. However, the algorithm can be very sensitive to initialization and might be stuck at a local minimum due to the fact that the objective function is often

Manuscript received February 24, 2019; revised April 5, 2019; accepted April 23, 2019. Date of publication April 29, 2019; date of current version July 16, 2019. This work was supported in part by National Science Foundation with Grants CNS-1824553, DMS-1622433, AST-1547436, and ECCS-1659025. The review of this paper was coordinated by Dr. Y. Gao. *(Corresponding author: Shuguang Cui.)*

H. Zhang and S. Cui are with the Department of Electrical and Computer Engineering, University of California, Davis, CA 95616 USA (e-mail: hanzh@ucdavis.edu; sgcui@ucdavis.edu).

B. Ai is with the State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Beijing 100044, China (e-mail: aibo@ieee.org).

W. Xu is with the Key Lab of Universal Wireless Communications, Ministry of Education, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: wjxu@bupt.edu.cn).

L. Xu is with the Institute of Computing Technology, Chinese Academy of Science, Beijing 100190, China (e-mail: lixu@ict.ac.cn).

Digital Object Identifier 10.1109/TVT.2019.2913865

non-convex. In addition, the algorithm can be time-costing, which restricts its usage in practical applications.

## *B. Traditional Extreme Learning Machines With Vector Inputs*

Traditional ELMs are originally designed to train SLFNs [5], which improve the training speed remarkably by randomly assigning $\mathbf{w}$ and $\mathbf{b}$, transferring the aforementioned optimization into least square problems. More specifically, let $\mathbf{H}$ be an $N \times L$ matrix, whose $(i, j)$-th entry is $\sigma(\mathbf{w}_j^T \mathbf{x}_i + b_j)$. Instead of minimizing the objective function with respect to $\mathbf{W}$, $\mathbf{b}$, and $\beta$, each entry of $\mathbf{W}$ and $\mathbf{b}$ is drawn from a continuous random distribution. After the matrix $\mathbf{H}$ is calculated, the solution to $\min_\beta ||\mathbf{H}\beta - \mathbf{T}||_2^2$ is given by $\hat{\beta} = \mathbf{H}^\dagger \mathbf{T}$ according to the Gauss-Markov theorem [10], where $\mathbf{H}^\dagger$ is the Moore-Penrose pseudoinverse of $\mathbf{H}$.

## *C. Tensor Operations and Tucker Decomposition*

In this paper, we treat tensors as multi-dimensional arrays. Specifically, a tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_K}$ of order $K$ stores $I_1 \times I_2 \times \cdots \times I_K$ elements. The inner product of two tensors, e.g., $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_K}$, is defined as $\langle \mathbf{X}, \mathbf{Y} \rangle = \langle \text{vec}(\mathbf{X}), \text{vec}(\mathbf{Y}) \rangle$. The vectorization $\text{vec}(\mathbf{X})$ of a tensor $\mathbf{X}$ is obtained by stacking the elements of $\mathbf{X}$ into an $(\prod_{k=1}^K I_k)$-dimensional column vector in a fixed order [11], e.g., in a lexicographical order or reverse lexicographical order. Tensors can also be unfolded into matrices. Given a tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_K}$ and an index $k$, the $k$-mode matricization $\mathbf{X}_{(k)}$ of $\mathbf{X}$ is a matrix with $\prod_{j \neq k} I_j$ columns and $I_k$ rows obtained by unfolding $\mathbf{X}$ along the $k$-th coordinate[11]. For $1 \leq k \leq K$, the $k$-mode rank of tensor $\mathbf{X}$, denoted by $\text{rank}_k(\mathbf{X})$, is defined as the rank of $\mathbf{X}_{(k)}$, which satisfies $\text{rank}_k(\mathbf{X}) \leq I_k$[11].

The Tucker decomposition is a branch of the higher-order singular value decomposition [12]. Consider $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_K}$ and a vector $(D_1, D_2, \ldots, D_K)$. If $D_k \geq \text{rank}_k(\mathbf{X})$ for all $k \in \{1, 2, \ldots, k\}$, there exist a core tensor $\mathbf{X}' \in \mathbb{R}^{D_1 \times D_2 \times \cdots \times D_K}$ and $K$ factor matrices, i.e., $\mathbf{B}(1)$, $\mathbf{B}(2)$, $\ldots$, $\mathbf{B}(K)$ with each $\mathbf{B}(k) \in \mathbb{R}^{I_k \times D_k}$, to be column-wise orthogonal:

$$\mathbf{X}_{i_1 i_2 \cdots i_K} = \sum_{d_1=1}^{D_1} \sum_{d_2=1}^{D_2} \cdots \sum_{d_K=1}^{D_K} \mathbf{X}'_{d_1 d_2 \cdots d_K} \left[ \prod_{j=1}^{K} \mathbf{B}(j)_{i_j, d_j} \right],$$

which could be written compactly as $\mathbf{X} = [\![\mathbf{X}'; \mathbf{B}(1), \ldots, \mathbf{B}(K)]\!]$. If $D_K = I_K$, there exists a core tensor $\mathbf{X}' \in \mathbb{R}^{D_1 \times D_2 \times \cdots \times D_K}$ and $K$ factor matrices $\{\mathbf{B}(k)\}_{k=1}^K$ with $\mathbf{B}(k) \in \mathbb{R}^{I_k \times D_k}$ for $1 \leq k \leq K-1$ and $\mathbf{B}(K) = \mathbf{I}$ (an $I_K \times I_K$ identity matrix) such that $\mathbf{X} = [\![\mathbf{X}'; \mathbf{B}(1), \ldots, \mathbf{B}(K-1), \mathbf{I}]\!]$. If $D_k < \text{rank}_k(\mathbf{X})$ for some $k$, such a core tensor and factor matrices do not exist. As an alternative, we can use an approximation of $\mathbf{X}$ by the truncated Tucker decomposition [11].

We next introduce an important property of the Tucker decomposition mentioned in [13].

*Lemma 1 (Duality Lemma):* Given a tensor pair $\mathbf{W} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_K}$ and $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_K}$, and their corresponding tucker decomposed core tensor $\mathbf{W}'$ and $\mathbf{X}'$, where $\mathbf{W} = [\![\mathbf{W}'; \mathbf{B}(1), \ldots, \mathbf{B}(K)]\!]$ and $\mathbf{X} = [\![\mathbf{X}'; \mathbf{B}(1), \ldots, \mathbf{B}(K)]\!]$, we have $\langle \mathbf{W}', \mathbf{X}' \rangle = \langle \mathbf{W}, \mathbf{X} \rangle$.

Lemma 1 tells us that the inner product can be done on the tucker decomposed core pair instead of on the original pair without loss in accuracy. If the original tensor admits a low-rank structure, the computational cost could be drastically reduced by calculating the inner product of the core tensors.

**Algorithm 1:** TELM.

```
**Input:** Data samples $(\mathbf{X}_i, t_i), i = 1, 2, \ldots, N$.
**Output:** Weight vector $\beta$.
1  Draw each entry of the weight tensor $\mathcal{W}$ and the bias vector $\mathbf{b}$
   from a continuous random distribution.
2  Calculate matrix $\mathbf{H}$ from $\{\mathbf{X}_i\}_{i=1}^N$, $\mathcal{W}$, and $\mathbf{b}$ by (2).
3  Calculate parameter vector $\beta$ by $\beta = \mathbf{H}^\dagger \mathbf{T}$.
4  Return $\beta$.
```

## III. Frameworks

## *A. Channel Interpolation*

To conduct high-efficiency data transmission in a typical Multi-input Multi-output Orthogonal Frequency Division Multiplexing (MIMO-OFDM) setting, CSI for each sub-carrier is required. One way of acquiring CSI is inserting a pilot signal to each sub-carrier, and then calculating the CSI at receiver. To further reduce the overhead of probing, the following interpolation scheme is adopted. Let $S_i$ denote the $i$-th sub-carrier's CSI; pilots are inserted into sub-carriers with odd indices. The CSI of sub-carriers with even indices are then inferred as

$$S_i = f(S_{i-w+1}, \ldots, S_{i-3}, S_{i-1}, S_{i+1}, S_{i+3}, \ldots, S_{i+w-1}),$$

where $w$ is the window length and $f$ is the interpolation function. In our work, we adopt the modified ELM as $f$ and the detail will be explained in the next section. Notice that different window design can be adopted to achieve the balance between carrier usage and accuracy.

## *B. Extreme Learning Machines With Tensorial Inputs*

Consider a dataset with $N$ data samples $(\mathbf{X}_i, t_i)$ for $i = 1, 2, \ldots, N$, where $\mathbf{X}_i \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_K}$ is a tensor of order $K$, and $t_i$ is its label. For an SLFN with $I_1 \times I_2 \times \cdots \times I_K$ input neurons and $L$ hidden neurons, its prediction $o_i$ of label $t_i$ could be calculated as $o_i = \sum_{j=1}^L \beta_j \sigma(\langle \text{vec}(\mathbf{W}_j), \text{vec}(\mathbf{X}_i) \rangle + b_j)$, where $\mathcal{W} = (\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_L)$ with $\mathbf{W}_j$ as the weight tensor of the $j$-th hidden neuron; $\mathbf{b}$, $\beta$, $\mathbf{T}$, $\mathbf{O}$, and $\sigma$ are defined as in Section II. The goal is still to minimize $f(\mathcal{W}, \mathbf{b}, \beta) = ||\mathbf{T} - \mathbf{O}||_2^2 = \sum_{i=1}^N (t_i - o_i)^2$. Consistent with the traditional ELM, we draw each entry of the weight tensor $\mathcal{W}$ and the bias vector $\mathbf{b}$ from a continuous random distribution and solve the problem $\min_\beta ||\mathbf{H}\beta - \mathbf{T}||_2^2$, where

$$\mathbf{H} = \begin{bmatrix} \sigma(\langle \mathbf{W}_1, \mathbf{X}_1 \rangle + b_1) & \ldots & \sigma(\langle \mathbf{W}_L, \mathbf{X}_1 \rangle + b_L) \\ \vdots & \ddots & \vdots \\ \sigma(\langle \mathbf{W}_1, \mathbf{X}_N \rangle + b_1) & \ldots & \sigma(\langle \mathbf{W}_L, \mathbf{X}_N \rangle + b_L) \end{bmatrix}. \quad (2)$$

The problem has a unique least square solution $\hat{\beta} = \mathbf{H}^\dagger \mathbf{T}$, with $\mathbf{H}^\dagger$ is the Moore-Penrose pseudoinverse of $\mathbf{H}$ as defined above. Detailed procedures are summarized in Algorithm 1. It is noteworthy to point out that the TELM handles the tensorial inputs with the same computational cost as the traditional ELM with vectorized inputs.

## *C. Tucker Decomposed Extreme Learning Machines*

To improve the learning efficiency of TELM, we further propose the Tucker decomposed extreme learning machine (TDELM) based on the Tucker decomposition method. Generally, computing $\mathbf{H}$ in (2) requires $NL \prod_{i=k}^K I_k$ multiplication operations, which is computationally demanding when dealing with a large dataset. However, by employing the Tucker decomposition, the computational cost of computing $\mathbf{H}$ could be largely reduced when working with the datasets with a low $k$-mode rank.

**Algorithm 2:** TDELM.

```
Input: Data samples $(\mathbf{X}_i, t_i), i = 1, 2, \ldots, N$.
Output: Weight vector $\beta$.
1  Concatenate $\mathbf{X}_1, \ldots, \mathbf{X}_N$ into a tensor $\mathbf{X}$ of order $(K+1)$.
2  Find the $k$-mode rank $n_k$ of $\mathbf{X}$ for $1 \leq k \leq K$.
3  Conduct a Tucker decomposition such that
   $\mathbf{X} = [\![\mathbf{X}'; \mathbf{B}(1), \ldots, \mathbf{B}(K), \mathbf{I}]\!], \mathbf{X}' \in \mathbb{R}^{n_1 \times \cdots \times n_K \times N}$,
   $\mathbf{B}(k) \in \mathbb{R}^{I_k \times n_k}$ for $1 \leq k \leq K$, and $\mathbf{I} \in \mathbb{R}^{N \times N}$.
4  Draw each entry of weight matrix $\mathcal{W}'$ and bias vector $\mathrm{b}'$ from
   a continuous random distribution.
5  Calculate matrix $\mathbf{H}$ from $\mathbf{X}'$, $\mathcal{W}'$, and $\mathrm{b}'$ by (3).
6  Calculate parameter vector $\beta$ by $\hat{\beta} = \mathbf{H}^{\dagger}\mathbf{T}$.
7  Return $\beta$.
```

Consider a dataset with $N$ samples $(\mathbf{X}_i, t_i)$ for $i = 1, 2, \ldots, N$. We first concatenate $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N$ into a tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_K \times N}$ of order $(K+1)$, then find the $k$-mode rank $n_k$ of $\mathbf{X}$ for $1 \leq k \leq K$, and next apply Tucker decomposition to $\mathbf{X}$ such that the core tensor $\mathbf{X}'$ is of size $n_1 \times n_2 \times \ldots \times n_K \times N$ and the $(K+1)$-th factor matrix $\mathrm{B}(K+1)$ is an identity matrix. Afterwards, we extract $\mathbf{X}'$ along the $(K+1)$-th axis into $N$ subtensors $\mathbf{X}'_1, \mathbf{X}'_2, \ldots, \mathbf{X}'_N$. We next consider an SLFN with $n_1 \times n_2 \times \cdots \times n_K$ input neurons and $L$ hidden neurons. Let $\mathcal{W}' = (\mathbf{W}'_1, \mathbf{W}'_2, \ldots, \mathbf{W}'_L)$ be the total weight tensor with $\mathbf{W}'_j$ as the weight tensor of the $j$-th hidden neuron, $\mathrm{b}' = (b'_1, b'_2, \ldots, b'_L)^T$ be the bias scalar from the input layer to the $j$-th hidden neuron, and $\beta' = (\beta'_1, \beta'_2, \ldots, \beta'_L)^T$ be the weight vector between the hidden layer and the output neuron. Each entry of $\mathcal{W}'$ or $\mathrm{b}'$ is randomly drawn from a continuous random distribution, and $\mathbf{H}$ is then calculated as

$$\mathbf{H} = \begin{bmatrix} \sigma(\langle \mathbf{W}'_1, \mathbf{X}'_1 \rangle + b_1) & \ldots & \sigma(\langle \mathbf{W}'_L, \mathbf{X}'_1 \rangle + b_L) \\ \vdots & \ddots & \vdots \\ \sigma(\langle \mathbf{W}'_1, \mathbf{X}'_N \rangle + b_1) & \ldots & \sigma(\langle \mathbf{W}'_L, \mathbf{X}'_N \rangle + b_L) \end{bmatrix}. \quad (3)$$

Finally, we solve the optimization problem $\min_{\beta} ||\mathbf{H}\beta - \mathbf{T}||_2^2$, with the least square square solution $\hat{\beta} = \mathbf{H}^{\dagger}\mathbf{T}$. The above procedures are summarized in Algorithm 2.

## IV. Properties of TELM/TDELM

In this section, we establish the interpolation theorems for the TELM and TDELM. Specifically, given a dataset with $N$ distinct data samples $(\mathbf{X}_i, t_i)$, $i = 1, 2, \ldots, N$, a TELM or TDELM with $N$ hidden neurons and sigmoid activation functions has the properties as follows.

*Theorem 1 (Interpolation Capability of TELMs):* Assume that each entry of the weight tensor $\mathcal{W}$ and bias vector $\mathrm{b}$ is randomly chosen from an interval according to a continuous probability distribution. Then with probability one, $\mathbf{H}$ in (2) is invertible and $||\mathbf{H}\beta - \mathbf{T}||_2 = 0$.

*Theorem 2 (Interpolation Capability of TDELMs):* Assume that each entry of the weight tensor $\mathcal{W}'$ and bias vector $\mathrm{b}'$ is randomly chosen from an interval according to a continuous probability distribution. Then with probability one, $\mathbf{H}$ in (3) is invertible and $||\mathbf{H}\beta - \mathbf{T}||_2 = 0$.

Theorem 1 can be treated as a special case of Theorem 2 by setting $\mathbf{X}' = \mathbf{X}$ and $\mathrm{B}(k) = \mathbf{I}$ for $1 \leq k \leq K$, the proof of Theorem 2 is provided as follows.

*Proof:* Define $\boldsymbol{\rho}_j(y) = [\rho_{j,1}(y), \ldots, \rho_{j,N}(y)]^T$, where $\rho_{j,i}(y) = \sigma(\langle \mathbf{W}'_j, \mathbf{X}'_i \rangle + y)$ for $1 \leq i \leq N$. Note that $\boldsymbol{\rho}_j(b_j)$ is the $j$-th column of $\mathbf{H}$ in (3). Let $\psi_k = \langle \mathbf{W}'_j, \mathbf{X}'_k \rangle$ for $1 \leq k \leq N$. Each entry of $\mathbf{W_j}'$ is drawn from a continuous distribution over an interval and $\mathbf{X}'_i$s are distinct from each other; thus $\langle \mathbf{W}'_j, \mathbf{X}'_k \rangle \neq \langle \mathbf{W}'_j, \mathbf{X}'_l \rangle$ for $k \neq l$ with probability one. Therefore, $\psi_1, \psi_2, \ldots, \psi_N$ are distinct from each other with probability one.

Now assume that $\boldsymbol{\rho}_j(y)$ belongs to a subspace of dimension less than $N$. Then there will be a vector $\gamma = (\gamma_1, \gamma_2, \ldots, \gamma_N)^T$ that is orthogonal to this subspace. Therefore, $\langle \gamma, \boldsymbol{\rho}_j(y) \rangle = 0$ for all $y$, i.e., $\sum_{i=1}^N \gamma_i \sigma(y + \psi_i) = 0$. We will then have $S(j) \triangleq \sum_{i=1}^N \gamma_i \sigma^j(y + \psi_i) = 0$ for $j = 1, 2, \ldots, N$ due to the fact that for $1 \leq j \leq N - 1$, taking the derivative of $S(j) = 0$ on both sides over $y$ yields $j \sum_{i=1}^N \gamma_i \sigma^j(y + \psi_i)[1 - \sigma(y + \psi_i)] = 0$, and hence $S(j+1) = S(j) - \sum_{i=1}^N \gamma_i \sigma^j(y + \psi_i)[1 - \sigma(y + \psi_i)] = 0$. Let $\mathbf{G}$ be an $N \times N$ matrix whose $(i,j)$-th entry is $\sigma^i(y + \psi_j)$ for $1 \leq i, j \leq N$. Then $\mathbf{G} \cdot (\gamma_1, \gamma_2, \ldots, \gamma_N)^T = (S(1), S(2), \ldots, S(N))^T = \mathbf{0}$, and hence $\mathbf{G}$ is a singular matrix.

On the other hand,

$$\begin{aligned} \det(\mathbf{G}) &= \sigma(y + \psi_1)\sigma(y + \psi_2) \cdots \sigma(y + \psi_N) \cdot \det(\mathbf{G}') \\ &= \prod_{k=1}^N \sigma(y + \psi_k) \cdot \prod_{1 \leq i < j \leq N} [\sigma(y + \psi_j) - \sigma(y + \psi_i)] \neq 0, \end{aligned}$$

where $\mathbf{G}'$ is an $N \times N$ matrix whose $(i,j)$-th entry is $\sigma^{i-1}(y + \psi_j)$ for $1 \leq i, j \leq N$, the second equality follows from the Vandermonde determinant [14], and the last inequality follows from the fact that $\sigma(y + \psi_i)$'s are distinct from each other. Then a contradiction appears. Thus $\boldsymbol{\rho}_j(y)$ belongs to a subspace of dimension $N$, and matrix $\mathbf{H}$ is thus of full rank with probability one. ■

## V. Simulation Results

In this section, an experiment is conducted over a real-world wireless MIMO channel response dataset to compare the performance of training time and accuracy among multiple methods including the purposed TDELM. We will show that: 1) TDELM achieves comparable prediction accuracy against other methods and 2) TDELM requires lower computational cost and shorter training time than ELM and SLFN over decomposed data.

The dataset was generated via conducting experiments in a lecture hall. Specifically, a 64-element virtual uniform linear array (ULA) is used as the transmitter (Tx) antenna array, and the receiver (Rx) with a single antenna is deployed at three different locations within the auditorium of the hall. The carrier frequency is set to be 4 GHz, and the measurement bandwidth is 200 MHz that is uniformly sampled at 513 frequency points. Sixteen continuous snapshots are obtained for each sub-channel. The obtained results are stored in a tensor of size $64 \times 3 \times 8208$, and each entry represents the response from one Tx array element to one Rx array element at a given frequency point. More detailed descriptions about the dataset could be found in [15].

The channel responses are normalized with zero-mean and unit standard derivation. The window size $W$ is set to be 4. The feature tensor $\mathbf{X} \in \mathbb{R}^{64 \times 3 \times 4 \times 8208}$ is created by a sliding window method, and the $(j,k,w,l)$-th element of $\mathbf{X}$ is given by $X_{j,k,w,l} = H_{jk(w-W+1+2(l-1))}$. The sliding window method will capture the local correlation in the neighborhood of adjacent points and the tensor decomposition is conducted on tensor $\mathbf{X}$.

The dataset is divided evenly into two subsets, one as the training set and the other as the test set, both containing 4,104 samples. Two neural networks, an ELM and a TDELM, are constructed with the same hidden-layer neurons for comparison fairness. The Tucker decomposition was implemented by modifying the Tensor Toolbox library [16]. Grid search is conducted for the hyperparameters: the net size and the decomposition size. The results of mean squared error (MSE) and running time are recorded. To mitigate the fluctuation caused by the randomness, the SelectBest method [17] is adopted and training is repeated 100 times to find the best parameter set. Least mean square

TABLE I
MSE AND TIME CONSUMPTION (TC)

| | TDELM | ELM | TD+NN |
|---|---|---|---|
| MSE | 0.0280 | 0.0286 | 0.0645 |
| TC (s) | 1.3786 | 1.5896 | 397.28 |
| | NN | Mean | LMSE |
| MSE | 0.0363 | 0.0769 | 0.0377 |
| TC (s) | 689.94 | 0.0334 | 0.0805 |

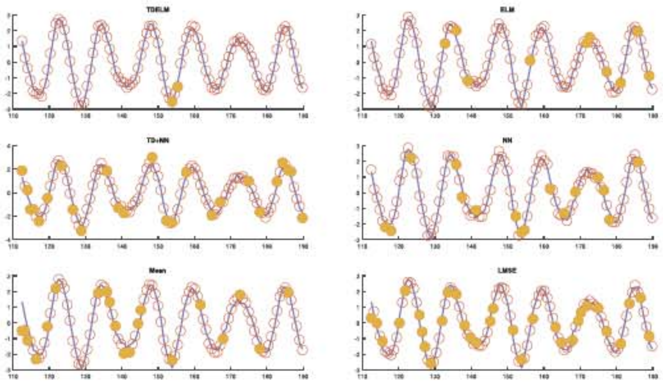

Fig. 1. Prediction results of learning machines.

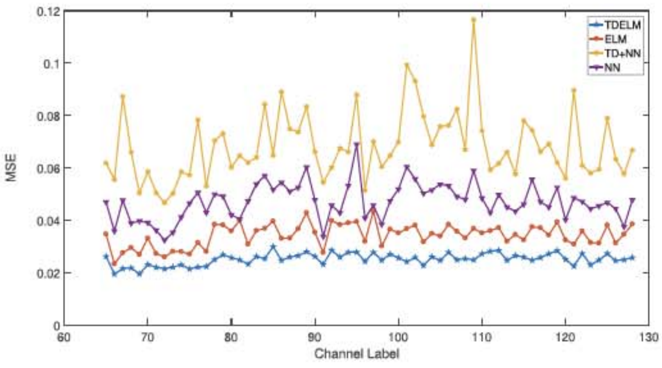

Fig. 2. Prediction results of learning machines.

filtering (LMSE), where the weighted combination of the data over a window is used as the prediction and each sub channel has its own weight parameter, and input averaging (Mean), where the mean of data over a window is used as the prediction, are also used as the basic comparison schemes.

The choices of hyperparameters are as follows. The node size in the hidden layer is 1,080, and the decomposition shrinks each tensor from $\{64, 3, 4\}$ to $\{64, 2, 2\}$, which leads to 66% less multiplications needed per inner production operation. The Tucker decomposition acquired by the Alternative Least Square algorithm[16] costs about 6 seconds for this dataset. The decomposition time does not scale with repeated training thus it is excluded from Table I. The performance in terms of MSE and time consumption (TC) are summarized in Table I and a snapshot of the interpolation results has been shown in Figs. 1 and 2.

From Table I and Fig. 2, it is shown that overall, TDELM achieved consistent gains in terms of MSE compared to all other methods, while also being the fastest on the decomposed dataset, compared with the TD+NN method. To showcase the gain better, one instance of the curve is shown in Fig. 1, where the blue lines denote the true values and yellow markers denote the interpolated values, respectively. When the prediction error for a particular point exceeds 0.3, which is roughly 10% of the maximal amplitude, the corresponding marker is filled as solid, which indicates a significant error. We can see there are much more significant errors among methods other than TDELM, which corresponds to smaller MSE shown in Fig. 2. Another interesting observation is, although TDELM has better accuracy than ELM, the TD+NN combination performs the worst. This implies that the gain might not only comes from applying tensor decomposition on the input data, but also from using the corresponding learning machine designed with such decomposition in mind.

## VI. CONCLUSION

In this paper, we proposed an extreme learning machine with tensorial inputs (TELM) and a Tucker decomposed extreme learning machine (TDELM) to handle the channel interpolation task in MIMO system. Moreover, we established a theoretical argument for the interpolation capability of TDELM. The experimental results verified that our proposed TDELM can achieve comparable performance against the traditional ELM but with reduced complexity, and outperform the other methods considerably in terms of the channel interpolation accuracy.

## REFERENCES

[1] H. C. Lee, J. C. Chiu, and K. H. Lin, "Two-dimensional interpolation-assisted channel prediction for OFDM systems," *IEEE Trans. Broadcast.*, vol. 59, no. 4, pp. 648–657, Dec. 2013.
[2] I. C. Wong, A. Forenza, R. W. Heath, and B. L. Evans, "Long range channel prediction for adaptive OFDM systems," in *Proc. IEEE Asilomar Conf. Signals Syst. Comput.*, Pacific Grove, CA, USA, Nov. 2004, pp. 732–736.
[3] Z. M. Fadlullah *et al.*, "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems," *IEEE Commun. Surv. Tut.*, vol. 19, no. 4, pp. 2432–2455, May 2017.
[4] N. Kato *et al.*, "The deep learning vision for heterogeneous network traffic control: Proposal, challenges, and future perspective," *IEEE Wireless Commun.*, vol. 24, no. 3, pp. 146–153, Jun. 2017.
[5] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Budapest, Hungary, Jul. 2004, pp. 985–990.
[6] A. H. Khanshan and H. Amindavar, "Performance evaluation of two-dimensional interpolations on OFDM channel estimation," in *Proc. 3rd IEEE/IFIP Int. Conf. Central Asia Internet*, Tashkent, Uzbekistan, 2007, pp. 1–5.
[7] D. Larsson, "Analysis of channel estimation methods for OFDMA," M.S. thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2006, pp. 12–19.
[8] N. K. Nair, and S. Asharaf, "Tensor decomposition based approach for training extreme learning machines," *Big Data Res.*, vol. 10, pp. 8–20, Dec. 2017.
[9] I. Kotsia and I. Patras, "Relative margin support tensor machines for gait and action recognition," in *Proc. ACM Int. Conf. Image Video Retrieval*, Xi'an, China, Jul. 2010, pp. 446–453.
[10] C. D. Meyer, *Matrix Analysis and Applied Linear Algebra.* Philadelphia, PA, USA: SIAM, Jun. 2001.
[11] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, Aug. 2009.
[12] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, Sep. 1966.
[13] X. Li, H. Zhou, and L. Li, "Tucker tensor regression and neuroimaging analysis," *Stat. Biosci.*, vol. 10, no. 3, pp. 520–545, Dec. 2018.
[14] S. Banerjee and A. Roy, *Linear Algebra and Matrix Analysis for Statistics.* Norwell, MA, USA: Chapman and Hall, Jun. 2014.
[15] B. Zhang *et al.*, "Empirical evaluation of indoor multi-user MIMO channels with linear and planar large antenna arrays," in *Proc. IEEE 28th Int. Symp. Pers., Indoor, Mobile Radio Commun.*, Montreal, QC, USA, Oct. 2017, pp. 1–6.
[16] B. W. Bader and T. G. Kolda, "MATLAB tensor toolbox version 2.6," 2015. [Online]. Available: http://www.sandia.gov/tgkolda/TensorToolbox/
[17] S. Dzeroski and B. Zenko, "Is combining classifiers with stacking better than selecting the best one?" *Mach. Learn.*, vol. 54, no. 3, pp. 255–273, Mar. 2004.