Signal Processing: Image Communication 73 (2019) 12–21

Contents lists available at ScienceDirect

Signal Processing: Image Communication

journal homepage: www.elsevier.com/locate/image

# Tensor rank learning in CP decomposition via convolutional neural network

Mingyi Zhou, Yipeng Liu *, Zhen Long, Longxi Chen, Ce Zhu

*Xiyuan Avenue 2006, Western High-Tech Zone, Chengdu, 611731, China*
*School of Information and Communication Engineering/Center for robotics/Center for Information in Medicine, University of Electronic Science and Technology of China (UESTC), China*

ARTICLE INFO

*Keywords:*
CANDECOMP/PARAFAC decomposition
Convolutional neural network
Deep learning
Low rank tensor approximation
Tensor rank estimation

ABSTRACT

Tensor factorization is a useful technique for capturing the high-order interactions in data analysis. One assumption of tensor decompositions is that a predefined rank should be known in advance. However, the tensor rank prediction is an NP-hard problem. The CANDECOMP/PARAFAC (CP) decomposition is a typical one. In this paper, we propose two methods based on convolutional neural network (CNN) to estimate CP tensor rank from noisy measurements. One applies CNN to the CP rank estimation directly. The other one adds a pre-decomposition for feature acquisition, which inputs rank-one components to CNN. Experimental results on synthetic and real-world datasets show the proposed methods outperforms state-of-the-art methods in terms of rank estimation accuracy.

## 1. Introduction

As a high-order generalization of matrix, tensor has a number of applications, such as signal processing [1], machine learning [2], physics [3], psychometrics and chemometrics [4]. Tensor decomposition is an effective method in analyzing high-dimensional data, e.g. Tucker decomposition [5], CANDECOMP/PARAFAC (CP) decomposition [6], and tensor singular value decomposition [7].

The CP decomposition factorizes a tensor into a sum of rank-one tensors [4]. It is a special case of Tucker decomposition when its core tensor is superdiagonal [4]. Fig. 1 illustrates the CP decomposition. For example, a tensor data $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$ can be decomposed as follows:

$$\mathcal{X} = \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \mathbf{a}_r^{(3)}, \tag{1}$$

where $R$ is the rank of the tensor $\mathcal{X}$. It is defined as the smallest number of the rank-one tensors [8].

In CP decomposition, the rank of a tensor is a predefined parameter, which means that it should be specified before the decomposition. However, rank determination is known to be an NP-hard problem [9,10]. In [11], probabilistic tensor decomposition method is used to estimate the tensor rank by modeling the complex interactions among tensor entities. In [12,13], an efficient deterministic Bayesian inference method called fully Bayesian CP factorization (FBCP) is developed using a hierarchical probabilistic model to predict the rank. It explicitly infers the underlying low CP rank tensor to capture the global information. In [14], a Bayesian method for low-rank CP decomposition of incomplete tensors, which infers the CP rank using a multiplicative Gamma process. [15] proposes an automatic relevance determination method for sparse tensor decomposition using the gradient-based sparse coding algorithm. Using Bayesian information criteria for complete tensors, a robust Tucker rank estimation algorithm is proposed in [16].

In recent years, deep neural network has been used to a series of data structure learning problems [17–26]. For example, in [17], a fast learning algorithm of sparse coding is proposed, which is useful to obtain good visual features. The speed of prediction over the network is much faster than iterations. It can be applied to a series of signal and image recovery applications [27–30].

As one of the most popular neural networks, the convolution neural network (CNN) can extract features which are hidden deeply, even when the measurements are noisy [31–33]. The tensor rank is one kind of data features, which represents the degree of interdependence between entries. Motivated by the above, the CNN is applied to solve the CP rank estimation problem.

In this paper, we propose two CNN based CP rank estimation methods from noisy measurements. One directly applies CNN to the CP rank estimation. For the other one, a pre-decomposition is conducted to obtain $\bar{R}$ rank-one components and inputs them into the 3-dimensional convolutional layers, where $\bar{R}$ is a predefined rank bound. The smallest number of the rank-one components $\hat{R}$ can be produced by the network.

* Corresponding author at: School of Information and Communication Engineering/Center for robotics/Center for Information in Medicine, University of Electronic Science and Technology of China (UESTC), China
*E-mail address:* yipengliu@uestc.edu.cn (Y. Liu).

https://doi.org/10.1016/j.image.2018.03.017
Received 14 February 2018; Received in revised form 23 March 2018; Accepted 24 March 2018
Available online 4 April 2018

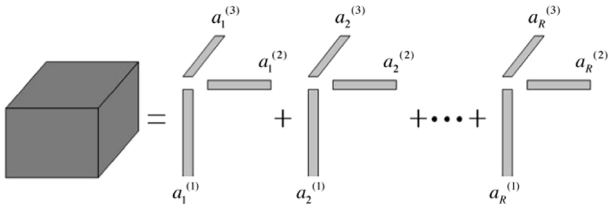

**Fig. 1.** Illustration of CP.

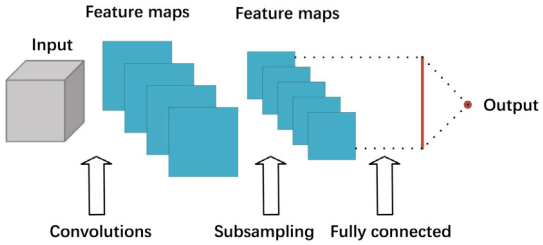


**Fig. 2.** Illustration of TRN model.

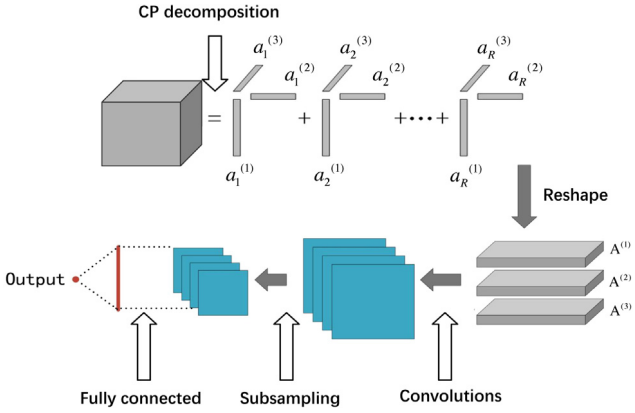


**Fig. 3.** Illustration of TRN-PD model.

Numerical experiments demonstrate that the proposed methods outperform state-of-the-art ones in terms of estimation accuracy.

The rest of this paper is organized as follows. In Section 2, the notations and preliminaries of CP decomposition are presented. In Section 3, the CNN based tensor rank estimation method is introduced. The experimental results for both synthetic and real-world datasets are in Section 4. The conclusion is drawn in Section 5.

## 2. Notations and preliminaries

In this section, we present the notations and preliminaries of CP decomposition briefly as used in [4].

A scalar, a vector, a matrix and a tensor are written as $a, \mathbf{a}, \mathbf{A}$ and $\mathcal{A}$, respectively. $R, \bar{R}$ and $\hat{R}$ denote tensor rank, predefined rank bound of pre-decomposition and predicted rank, respectively. $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$ and $\bar{\mathbf{A}}^{(n)} \in \mathbb{R}^{I_n \times \bar{R}}$ are the CP factor matrices using ranks $R$ and $\bar{R}$, respectively. The Moore Penrose pseudo-inverse of the matrix $\mathbf{U}$ is denoted by $\mathbf{U}^{\dagger}$. $\mathbf{w}$ denotes the weighting vector generated by CP decomposition. $\mathbf{V}$ is the weights of CNN model. $\|\mathcal{X}\|_{\mathrm{F}} = \sqrt{\sum_{i_1,i_2,\ldots,i_N} \mathcal{X}^2_{i_1,i_2,\ldots,i_N}}$ denotes the Frobenius norm of $N$-dimensional tensor $\mathcal{X}$.

**Definition 2.1** (*Outer Product*)**.** An N-way tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is rank-one if it can be written as the outer product of $N$ vectors:,

$$\mathcal{X} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \cdots \circ \mathbf{a}^{(N)}. \tag{2}$$

**Definition 2.2** (*Kronecker Product*)**.** The Kronecker product of $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{K \times L}$ is denoted by $\mathbf{A} \otimes \mathbf{B}$. The result is an $IK \times JL$ matrix:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{I2}\mathbf{B} & \cdots & a_{1J}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2J}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\mathbf{B} & a_{I2}\mathbf{B} & \cdots & a_{IJ}\mathbf{B} \end{bmatrix} \tag{3}$$

$$= [\mathbf{a}_1 \otimes \mathbf{b}_1 \quad \mathbf{a}_1 \otimes \mathbf{b}_2 \quad \cdots \quad \mathbf{a}_J \otimes \mathbf{b}_{L-1} \quad \mathbf{a}_J \otimes \mathbf{b}_L]. \tag{4}$$

**Definition 2.3** (*Khatri–Rao Product*)**.** The Khatri–Rao product is a matching columnwise product. The Khatri–Rao product of $\mathbf{A} \in \mathbb{R}^{I \times K}$ and $\mathbf{B} \in \mathbb{R}^{J \times K}$ is denoted by $\mathbf{A} \odot \mathbf{B}$. The result is a matrix of the size $IJ \times K$ as:

$$\mathbf{A} \odot \mathbf{B} = [\mathbf{a}_1 \otimes \mathbf{b}_1 \quad \mathbf{a}_2 \otimes \mathbf{b}_2 \quad \cdots \quad \mathbf{a}_K \otimes \mathbf{b}_K]. \tag{5}$$

**Definition 2.4** (*Hadamard Product*)**.** The Hadamard product is an elementwise matrix product. For given matrices $\mathbf{A}$ and $\mathbf{B}$, which have the same size $I \times J$, their Hadamard product denoted by $\mathbf{A} * \mathbf{B}$ is:

$$\mathbf{A} * \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \cdots & a_{1J}b_{1J} \\ a_{21}b_{11} & a_{22}b_{12} & \cdots & a_{2J}b_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}b_{I1} & a_{I2}b_{I2} & \cdots & a_{IJ}b_{IJ} \end{bmatrix} \tag{6}$$

**Definition 2.5** (*CP Decomposition*)**.** The CP decomposition factorizes a tensor into a number of rank-one tensors. Given a 3-dimensional tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, we can write it as

$$\mathcal{X} = \sum_{r=1}^{R} w_r \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \mathbf{a}_r^{(3)}, \tag{7}$$

where $R$ is a positive integer, $w_r$ is the weight for factors, $\mathbf{a}_r \in \mathbb{R}^I$, $\mathbf{a}_r^{(2)} \in \mathbb{R}^J$, and $\mathbf{a}_r^{(3)} \in \mathbb{R}^K$, $r = 1, \ldots, R$ are normalized factor vectors. We can put all the $\mathbf{a}_r^{(1)}, \mathbf{a}_r^{(2)}, \mathbf{a}_r^{(3)}$, $r = 1, \ldots, R$ into matrices $\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \mathbf{A}^{(3)}$ respectively. All the weights are put in to the vector $\mathbf{w} = [w_1, \ldots, w_R]^{\mathrm{T}}$. For an $N$-dimensional tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, its CP decomposition can be formulated as:

$$\mathcal{X} = \sum_{r=1}^{R} w_r \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \cdots \circ \mathbf{a}_r^{(N)}, \tag{8}$$

Algorithm 1 gives the details of CP decomposition solved by the alternative least squares (CP-ALS).

**Definition 2.6** (*Tensor Rank R*)**.** The rank of high-order tensor $R$ is defined as the smallest number of rank-one tensors [34].

**Algorithm 1** CP-ALS

**Input:** $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \cdots I_N}$, $R$
**initialize** $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$ for $n = 1, \ldots, N$
**repeat**
**for** $n = 1, \ldots, N$ **do**
$\mathbf{U} := \mathbf{A}^{(1)^{\mathrm{T}}}\mathbf{A}^{(1)} * \mathbf{A}^{(2)^{\mathrm{T}}}\mathbf{A}^{(2)} * \cdots * \mathbf{A}^{(N)^{\mathrm{T}}}\mathbf{A}^{(N)}$
$\mathbf{A}^{(n)} := \mathbf{X}^{(n)}(\mathbf{A}^{(N)} \odot \mathbf{A}^{(N-1)} \odot \cdots \odot \mathbf{A}^{(1)})\mathbf{U}^{\dagger}$
**end for**
**until** fitting ceases to improve or reaching maximum iteration number
**Output** $\mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}$

## 3. Tensor rank learning

A 3-dimensional tensor data $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ can be decomposed into the weighted sum of rank-one tensors. However, only the noisy measurement $\mathcal{Y} \in \mathbb{R}^{I \times J \times K}$ is available in a lot of practical applications. In low rank tensor approximation, a minimum rank of the noisy data is obtained by minimizing the data fitting as follows:

$$\min_{R} \quad \frac{1}{2}\|\mathcal{Y} - \sum_{r=1}^{R} w_r \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \mathbf{a}_r^{(3)}\|_{\mathrm{F}}^2, \tag{9}$$

which is an NP-hard problem [9,10].

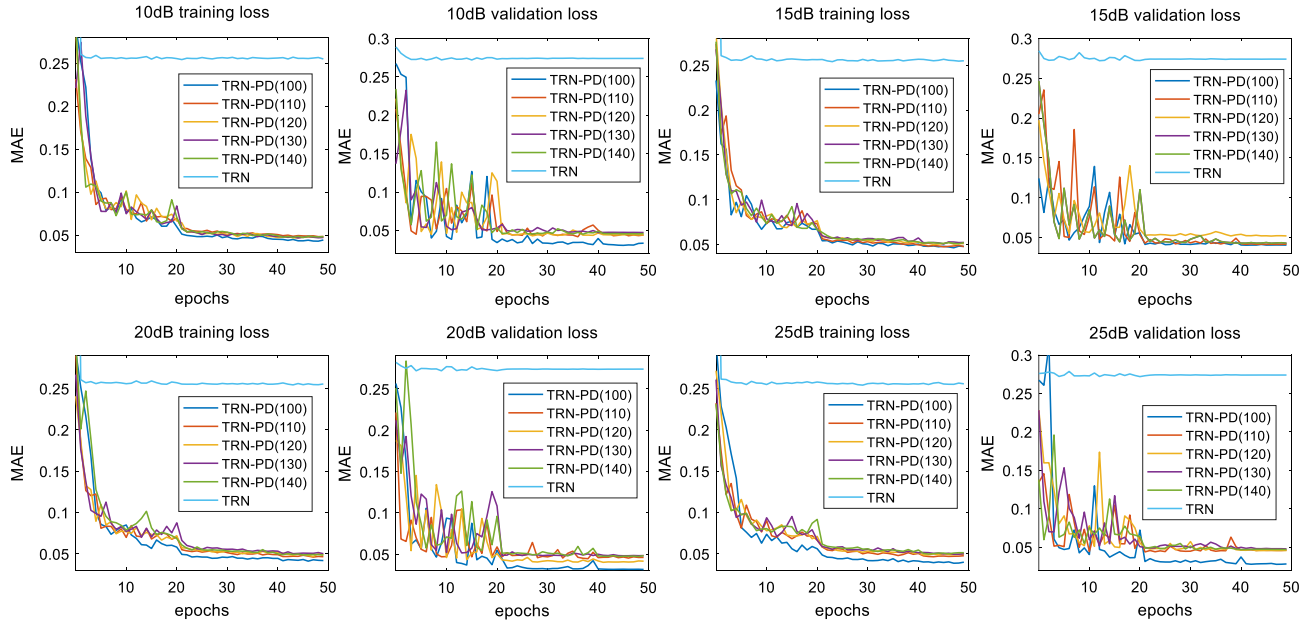


**Fig. 4.** Illustration of MAE performance comparison between TRN and TRN-PD on the $50 \times 50 \times 50$ synthetic dataset at different noise levels.

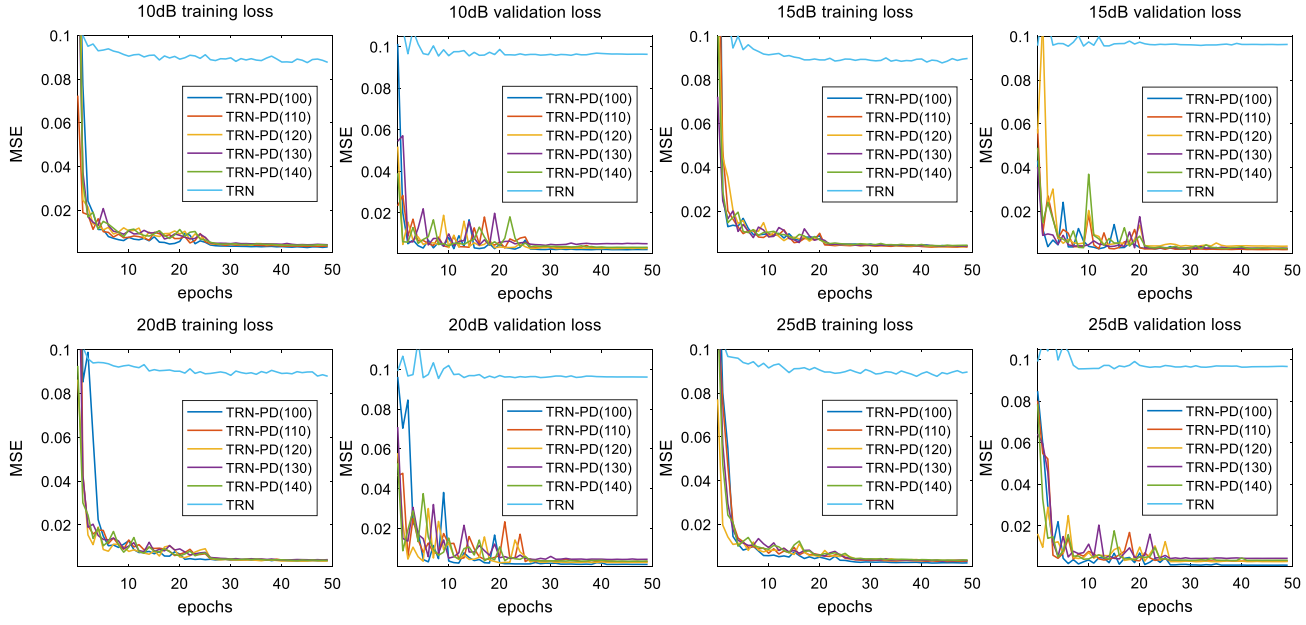


**Fig. 5.** Illustration of MSE performance comparison between TRN and TRN-PD on the $50 \times 50 \times 50$ synthetic dataset.

### 3.1. Tensor rank learning architecture

Applying deep learning for CP-rank estimation, we propose a new method using CNN to estimate the rank of tensor $\mathcal{X}$ from its noisy measurement $\mathcal{Y}$, which is called the tensor rank network (TRN). The basic idea is to build a model with a fixed depth that can be trained to estimate rank $R$. The architecture of our predictor is denoted by $g(\mathbf{V}, \mathcal{Y})$, where $\mathbf{V}$ is trainable parameters in the network. For training the neural network, the stochastic gradient is used to optimize the weights in deep network by minimizing the loss function $\mathcal{L}(\mathbf{V})$. It is defined as the error between the predicted rank from $\mathcal{Y}$ and the low rank of $\mathcal{X}$:

$$\mathcal{L}(\mathbf{V}) = \frac{1}{M}\sum_{n=1}^{M}(R_m - g(\mathbf{V}, \mathcal{Y}_m))^2, \tag{10}$$

where $M$ is the number of samples for training, and $R_m$ is the optimal rank value. The learning algorithm is stochastic gradient decent, which is formulated as follows:

$$\mathbf{V}(s+1) = \mathbf{V}(s) - \eta(s)\frac{d\mathcal{L}}{d\mathbf{V}}, \tag{11}$$

where $s$ is the number of training iterations, and $\eta(s)$ is to ensure convergence.

To learn a CNN to acquire the feature in the data, stochastic gradient decent algorithm is used to optimize the weights, and the gradients can be back-propagated through them. Fig. 2 illustrates the TRN of the form:

$$\hat{R} = g(\mathbf{V}, \mathcal{Y}), \tag{12}$$

where $\mathcal{Y} \in \mathbb{R}^{I \times J \times K}$ is the input, and $g$ is a non-linear function, as proposed by [35].

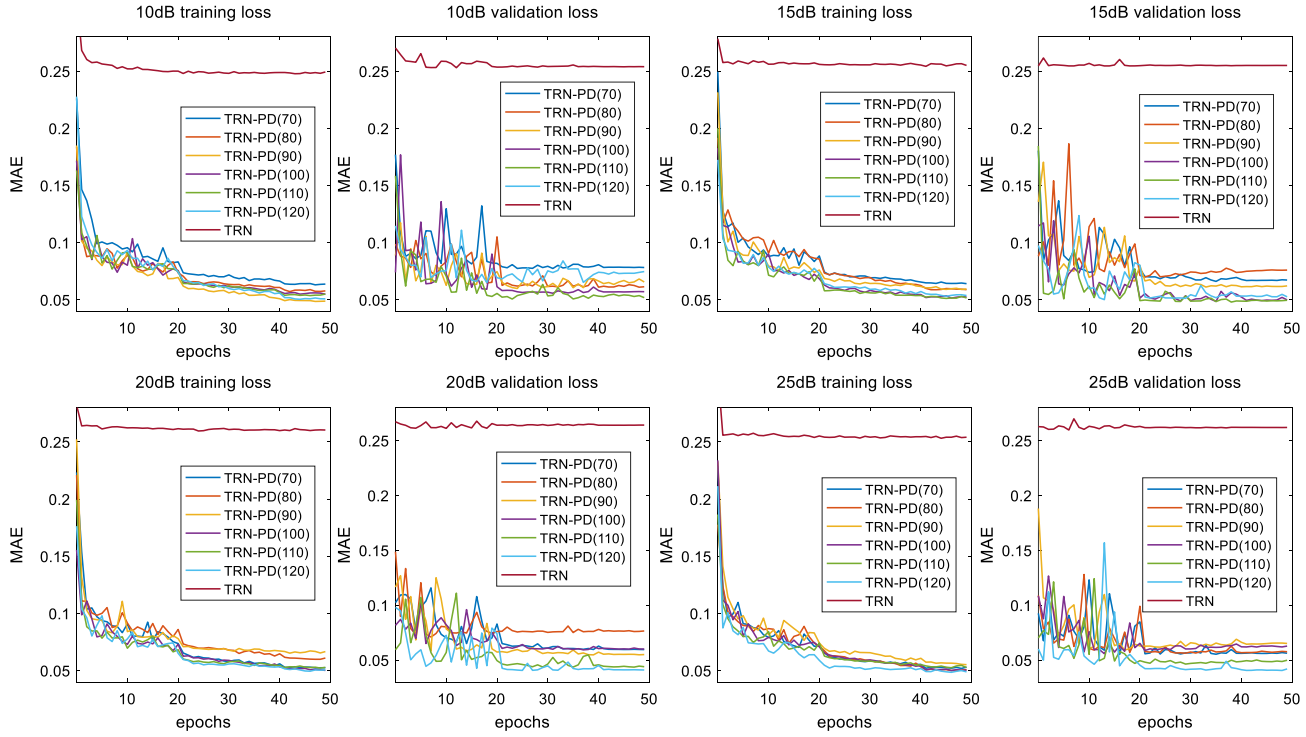


**Fig. 6.** Illustration of MAE performance comparison between TRN and TRN-PD on the $20 \times 20 \times 20$ synthetic dataset at different noise levels.

As will be demonstrated in Section 4, this architecture performs well on small-scale dataset, but it does not do the same well in some large-scale dataset experiments. This architecture has two major shortcomings. First, the 3-dimensional convolutional layers consume a lot of computational resources in large-scale data. Second, the high-order interactions are hidden more deeply in the large-scale data, which makes the loss of deep learning model hard to decrease.

### 3.2. TRN-PD architecture

To address these issues, we further transform (9) into

$$\min_{\mathbf{w}} \quad \lambda\|\mathbf{w}\|_0 + \frac{1}{2}\|\mathcal{Y} - \sum_{r=1}^{R} w_r \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \mathbf{a}_r^{(3)}\|_{\mathrm{F}}^2, \tag{13}$$

where $\mathbf{w} = [w_1, \ldots, w_r, \ldots, w_R]^{\mathrm{T}}$ is the weight vector, and $R = \|\mathbf{w}\|_0$ denotes the number of nonzero entries in $\mathbf{w}$.

To solve the rank estimation problem (13), we assume a predefined rank bound $\bar{R}$, and perform a pre-decomposition to obtain $\bar{\mathbf{w}}, \bar{\mathbf{a}}^{(n)}, n = 1, 2, 3$ as follows:

$$\min_{\bar{\mathbf{w}}, \{\bar{\mathbf{a}}^{(1)}\}, \{\bar{\mathbf{a}}^{(2)}\}, \{\bar{\mathbf{a}}^{(3)}\},} \quad \|\mathcal{Y} - \sum_{r=1}^{\bar{R}} \bar{w}_r \bar{\mathbf{a}}_r^{(1)} \circ \bar{\mathbf{a}}_r^{(2)} \circ \bar{\mathbf{a}}_r^{(3)}\|_{\mathrm{F}}^2. \tag{14}$$

This optimization problem can be solved by alternative least squares (ALS) [4]. The obtained matrices $\bar{\mathbf{A}}^{(1)}, \bar{\mathbf{A}}^{(2)}, \bar{\mathbf{A}}^{(3)}$ can be stacked into a tensor $\bar{\mathcal{Y}}$ along the 3-mode.

As Fig. 3 illustrates, we can input $\bar{\mathcal{Y}}$ to the network (12) to get the predicted rank $\hat{R}$. In fact, the network (12) is used to solve the following optimization model:

$$\begin{aligned} &\min_{\hat{R}, \hat{\mathbf{w}}, \mathbf{a}_r^{(n)}} \quad \hat{R} = \|\hat{\mathbf{w}}\|_0, \\ &\mathbf{s.t.} \quad \sum_{r=1}^{\|\hat{\mathbf{w}}\|_0} \hat{w}_r \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \mathbf{a}_r^{(3)} = \sum_{r=1}^{\bar{R}} \bar{w}_r \bar{\mathbf{a}}_r^{(1)} \circ \bar{\mathbf{a}}_r^{(2)} \circ \bar{\mathbf{a}}_r^{(3)}. \end{aligned} \tag{15}$$

We call this CP-rank estimation method tensor rank network with pre-decomposition (TRN-PD), which is summarized in Algorithm 2.

As will be demonstrated in Section 4, this method performs better than TRN on the large-scale dataset. In TRN-PD, (14) can be regarded as a step for feature extraction. Compared with the original data directly, TRN-PD uses a series of rank-one tensors based pre-decomposition to predict the smallest number of rank-one tensors in CP decomposition. The pre-decomposition can reduce the size of the input data in neural network, especially for the large-scale dataset. For example, a tensor $\mathcal{Y}$ with the size $I \times J \times K$ can be reduced to $Q \times \bar{R} \times 3$ by pre-decomposition, where $Q$ is the largest number of $I$, $J$, and $K$. In addition, as Fig. 3 illustrates, the $N$-dimensional ($N \geq 3$) tensor can be decomposed into $N$ factor matrices $\bar{\mathbf{A}}^{(1)}, \bar{\mathbf{A}}^{(2)}, \ldots, \bar{\mathbf{A}}^{(N)}$, and the obtained matrices $\bar{\mathbf{A}}^{(1)}, \bar{\mathbf{A}}^{(2)}, \ldots, \bar{\mathbf{A}}^{(N)}$ can be stacked into a 3-dimensional tensor along the 3-mode. Therefore, TRN-PD is an efficient method to estimate the rank of higher dimensional tensor.

Fig. 3 is the illustration of the proposed TRN-PD method. In pre-decomposition step, the predefined rank bound $\bar{R}$ should be given and larger than predicted rank.

**Algorithm 2** TRN-PD

**Input:** $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \cdots I_N}$
$\mathbf{V}$ is saved by backpropagation algorithm
use CP-ALS with rank $\bar{R}$ to get $\bar{\mathbf{A}}^{(1)}, \bar{\mathbf{A}}^{(2)}, \ldots, \bar{\mathbf{A}}^{(N)}$
reshape $\bar{\mathbf{A}}^{(1)}, \bar{\mathbf{A}}^{(2)}, \ldots, \bar{\mathbf{A}}^{(N)}$ to get a new tensor $\bar{\mathcal{Y}}$
use model $g(\mathbf{V}, \bar{\mathcal{Y}})$ to predict $\hat{R}$
**Output** $\hat{R}$.

## 4. Experimental results

In this section, numerical results are conducted to show the tensor rank estimation performance improvement by the proposed methods. Two state-of-the-art methods are employed for comparison:

FBCP [12]: FBCP develops an efficient deterministic Bayesian inference algorithm [12], building a hierarchical probabilistic model to predict the rank.

BRTF [13]: BRTF is a Bayesian robust tensor factorization which employs a fully Bayesian generative model for automatic CP rank estimation.

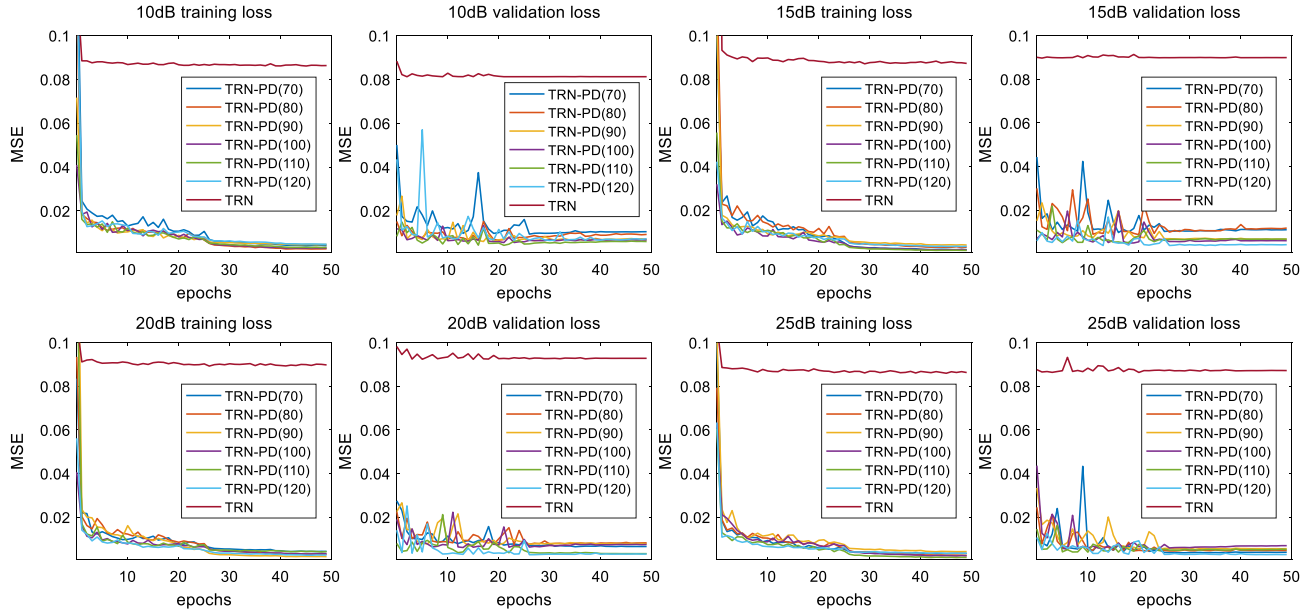


**Fig. 7.** Illustration of MSE performance comparison between TRN and TRN-PD on the $20 \times 20 \times 20$ synthetic dataset at different noise levels.

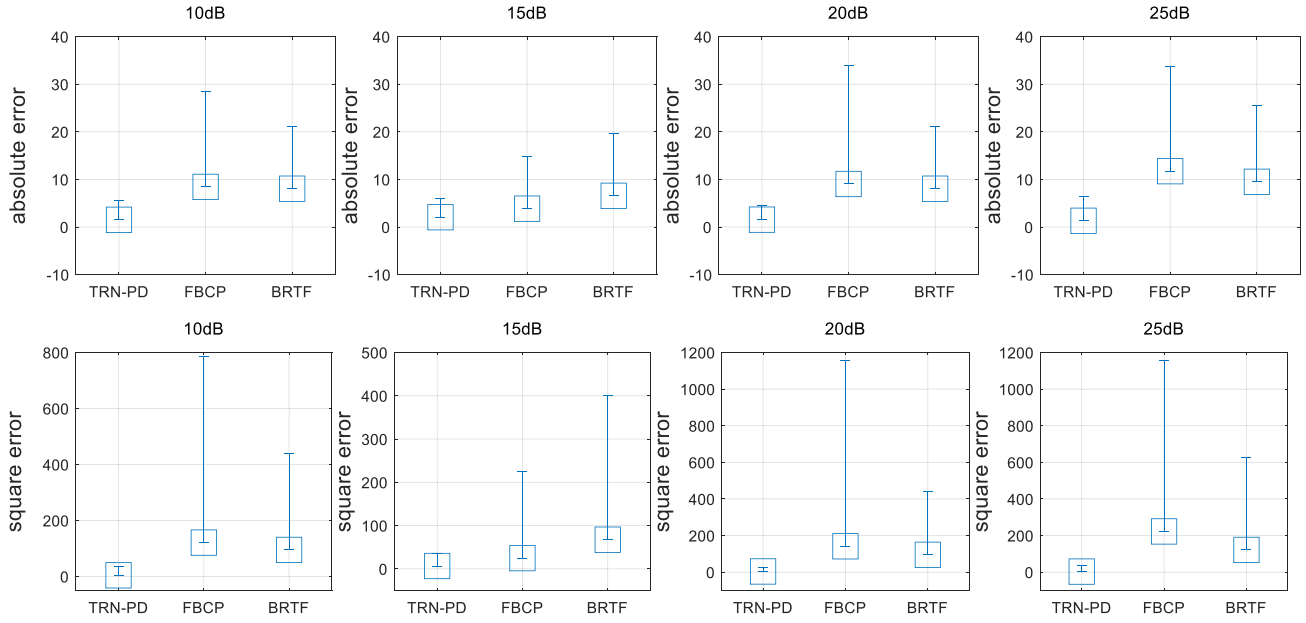


**Fig. 8.** Illustration of accuracy comparison between TRN, FBCP and BRTF on $50 \times 50 \times 50$ synthetic dataset at different noise levels.

All experiments are performed on a desktop with Intel core i7-7700k CPU and Nvidia GTX-1080Ti GPU.

### 4.1. Datasets

Firstly, we generate tensors with rank $R$. $R$ of the $50 \times 50 \times 50$ synthetic data ranges from 50 to 100, and $R$ of $20 \times 20 \times 20$ synthetic data ranges from 20 to 70. Then, each tensor is added random Gaussian noise. In this way, we get 4 datasets at different noise levels. More specifically, the values of signal-to-noise ratio (SNR) are set to 10 dB, 15 dB, 20 dB and 25 dB.

These methods are evaluated on a real-world dataset too, i.e. CMU Pose Illumination and Expression database [36]. This dataset has many human face grayscale images in different light and angle. We generate tensors that each tensor has seven different shades of grayscale images. Thus the size is $32 \times 32 \times 7$. Similar to the previous synthetic dataset, the values of SNR are set to 10 dB, 15 dB, 20 dB and 25 dB with added noise on the real-world data.

The number of training sets are 2000, 3000 and 900 on the $50 \times 50 \times 50$ synthetic dataset, $20 \times 20 \times 20$ synthetic dataset and CMU database, respectively.

### 4.2. Experimental results on synthetic datasets

In this group of experiments, all the methods are applied on synthetic datasets. Two functions are used to evaluate the estimation accuracy. One is the mean absolute error (MAE) which is calculated by $\text{MAE} = \frac{1}{M}\sum_{m=1}^{M}|R_m - \hat{R}_m|$, where $R_m$ is the real rank of testing data and $\bar{R}_m$ is

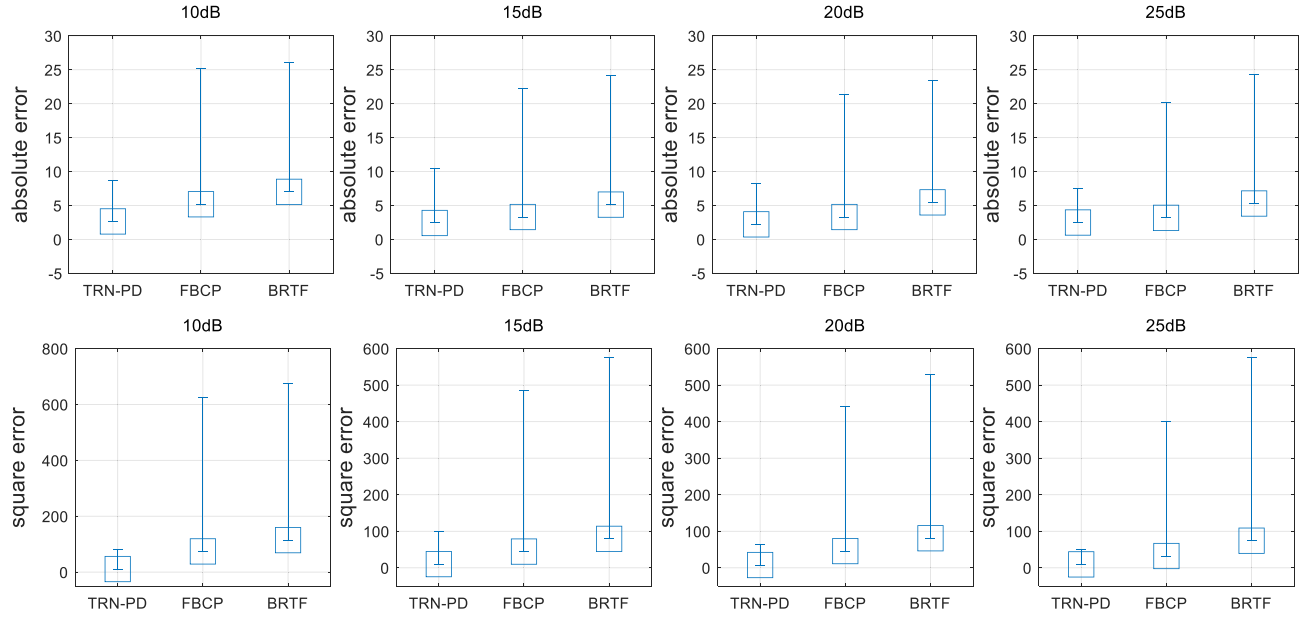


**Fig. 9.** Illustration of accuracy comparison between TRN, FBCP and BRTF on $20 \times 20 \times 20$ synthetic dataset at different noise levels.

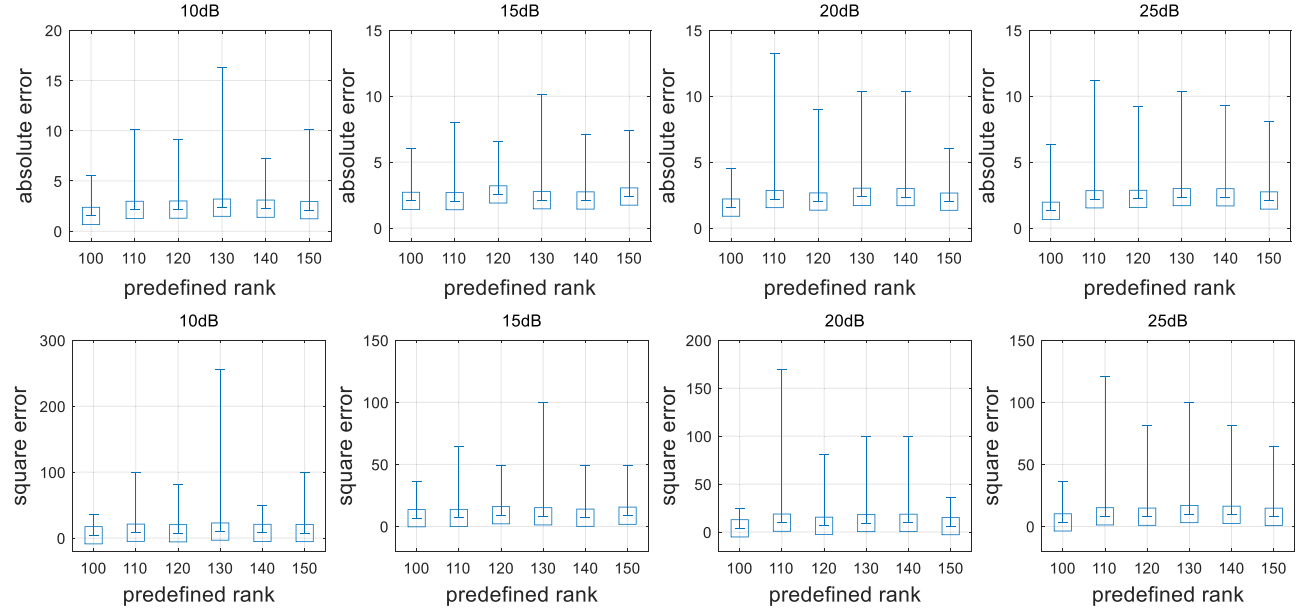


**Fig. 10.** Illustration of the rank estimation performance with different pre-decomposition parameters at different noise levels in $50 \times 50 \times 50$ synthetic dataset.

the predicted rank. The other is mean square error (MSE), which can be calculated by MSE $= \frac{1}{M}\sum_{m=1}^{M}(R_m - \hat{R}_m)^2$.

We obtain the performance of TRN and TRN-PD on synthetic datasets. We set the predefined rank bound of pre-decomposition $\bar{R}$ from 100 to 140 on $50 \times 50 \times 50$ dataset. Both MAE and MSE are used to evaluate the performance of TRN and TRN-PD. Figs. 4 and 5 are the illustrations of learning process on $50 \times 50 \times 50$ data in the first 50 iterations. TRN-PD($\bar{R}$) denotes that the TRN-PD method with the predefined rank bound of pre-decomposition $\bar{R}$. In training for the network, we perform training and validation alternatively. The figures of training loss and validation loss present the accuracy performance in training data group and validation data group respectively. They show TRN-PD works well on this dataset, but TRN can hardly converge.

We set the predefined rank bound of pre-decomposition $\bar{R}$ from 70 to 120 on the $20 \times 20 \times 20$ dataset. Figs. 6 and 7 are the illustrations of training process on the $20 \times 20 \times 20$ dataset. They show TRN method is hard to find the high-order interactions of data. But TRN-PD performs well on this dataset. It proves that the pre-decomposition is able to exact features and helps the deep network in rank prediction.

Moreover, we compare TRN-PD, FBCP and BRTF on synthetic datasets. We set $\bar{R} = 100$ on $50 \times 50 \times 50$ dataset. Fig. 8 illustrates the rank estimation accuracy performance. The similar performance comparison on the $20 \times 20 \times 20$ dataset is given in Fig. 9. The experimental results indicate that the TRN-PD method gets better results in terms of mean absolute error, mean square error, maximum absolute error and maximum square error on synthetic datasets. The TRN-PD has a larger error gap on the dataset of the size $50 \times 50 \times 50$. All these experiments

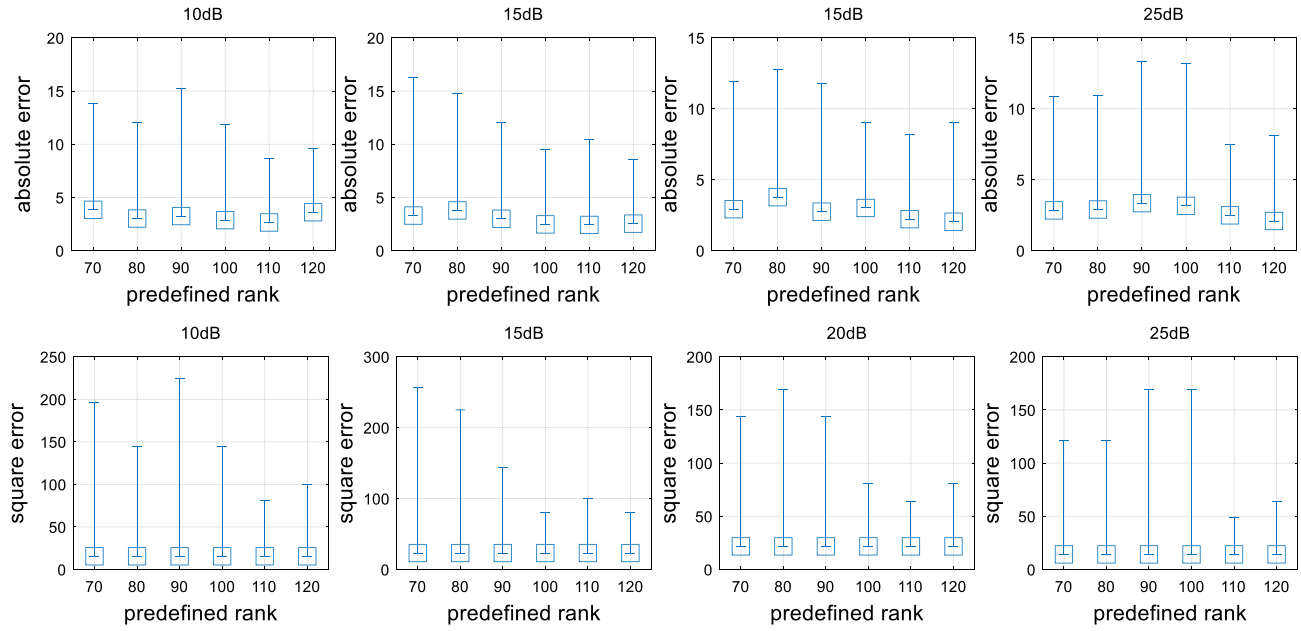


**Fig. 11.** Illustration of the rank estimation performance with different pre-decomposition parameters at different noise levels on $20 \times 20 \times 20$ synthetic dataset.

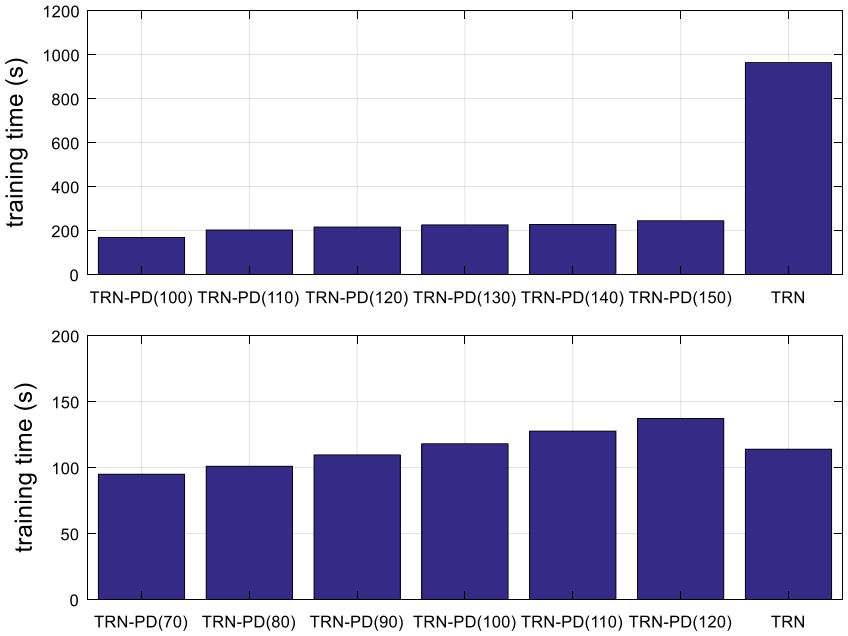


**Fig. 12.** Illustration of training time comparison between TRN and TRN-PD on synthetic dataset with 10 dB noise: $50 \times 50 \times 50$ (top), $20 \times 20 \times 20$ (bottom).

show TRN-PD has the better rank estimation accuracy than those of FBCP and BRTF.

Figs. 10 and 11 demonstrate the MAE and MSE comparisons with different predefined rank bound of pre-decomposition $\bar{R}$ in different data scale. They show the optimal $\bar{R}$ varies for different datasets.

Fig. 12 is the illustration of training times in 50 epochs on synthetic datasets. TRN-PD consumes less time than TRN on large-scale dataset. But both methods have similar training time on small-scale dataset.

### 4.3. Experimental results on real-world database

In this subsection, we show the performance improvement of the proposed method on CMU pose illumination and expression dataset. We stack facial photos of the same person with different light levels to generate 980 tensors of the size $32 \times 32 \times 7$. Gaussian noise is added to this dataset to achieve different value of SNR. Since this is a real-world dataset, none of these methods has any prior and a posteriori information.

The root mean square error (RMSE) is used for accuracy comparison for all the methods in data recovery, i.e. , RMSE $= \sqrt{(\|\mathcal{X} - \hat{\mathcal{X}}\|_F^2 / P)}$, where $\mathcal{X}$ is the original testing data without noise, $\hat{\mathcal{X}}$ is the reconstructed data, and $P$ is the number of elements in $\mathcal{X}$.

To test the performance of tensor rank prediction methods for testing data, we regard the best rank as the one which makes the recovered data have the lowest RMSE by using CP decomposition based denoising. Similar to the ones for synthetic data, the training performance is

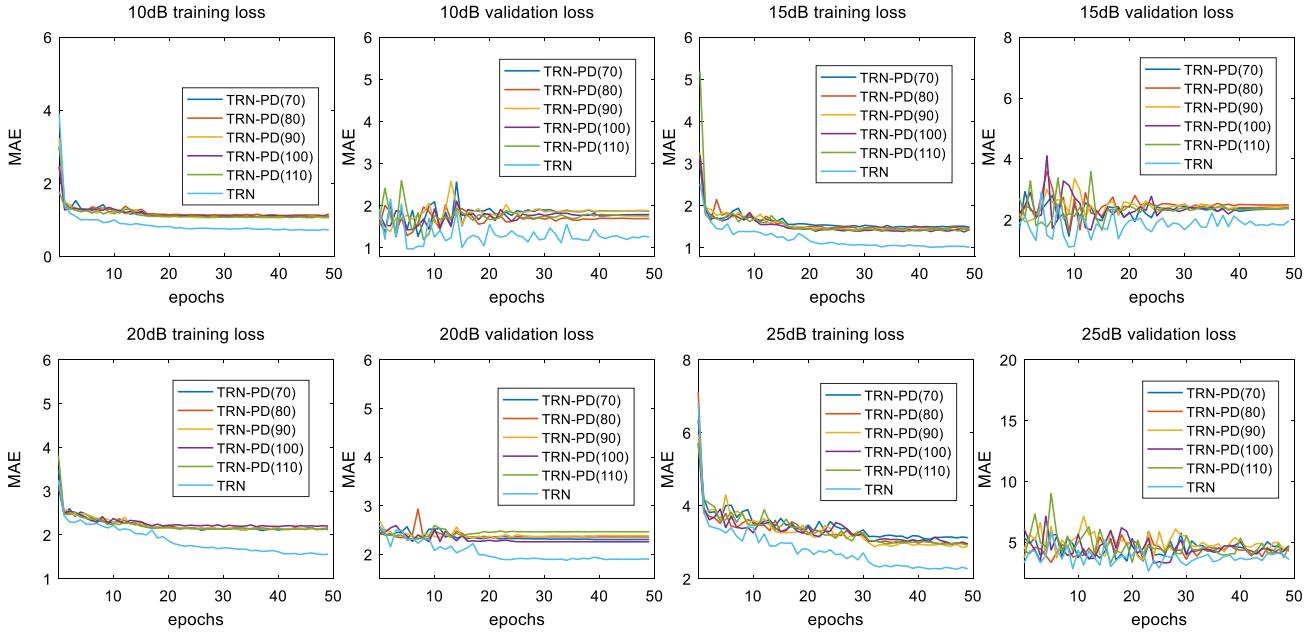


**Fig. 13.** Illustration of MAEs given by training TRN and TRN-PD on CMU dataset at different noise levels.

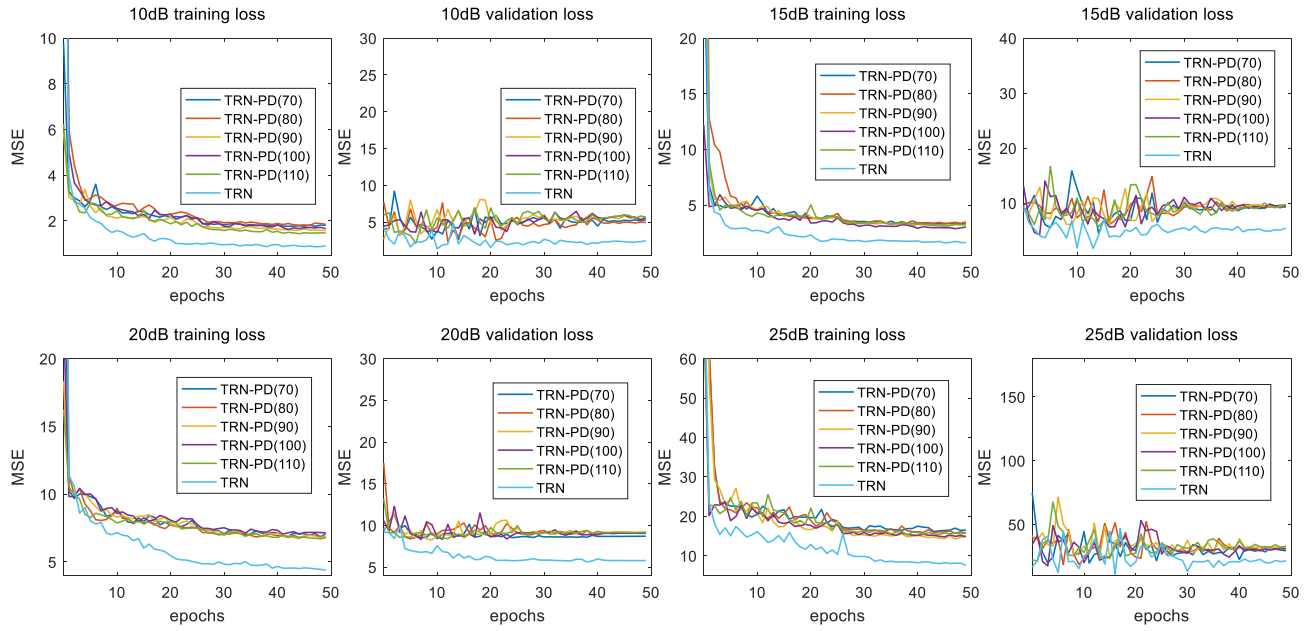


**Fig. 14.** Illustration of MSEs given by training TRN and TRN-PD on CMU dataset at different noise levels.

illustrated in Figs. 13 and 14, and it indicates that TRN obtains the lowest error in this dataset. In Fig. 15, we compare the recovered ranks and the best ranks for the real-world data, and we can see that TRN obtains the best prediction accuracy on this dataset.

Fig. 16 is the illustration of training time in 50 epochs on real-world database. The training time of TRN-PD is similar to that of TRN.

## 5. Conclusion

Considering that deep learning is able to capture the high-order feature which is hidden in depth, we propose two deep learning methods to predict the CP rank in this paper. One is the typical CNN model, and the other adds a pre-decomposition to extract features. In numerical experiments, we prove that the proposed TRN and TRN-PD achieve better estimation accuracy than those of FBCP and BRTF in both synthetic and real-world datasets. In the future, we will iteratively use the proposed TRN-PD architecture to further improve the robustness to pre-estimation rank.

## Acknowledgments

This research is supported by National Natural Science Foundation of China (NSFC, No. 61602091, No. 61571102) and the Fundamental Research Funds for the Central Universities (No. ZYGX2016J199, No. ZYGX2014Z003).

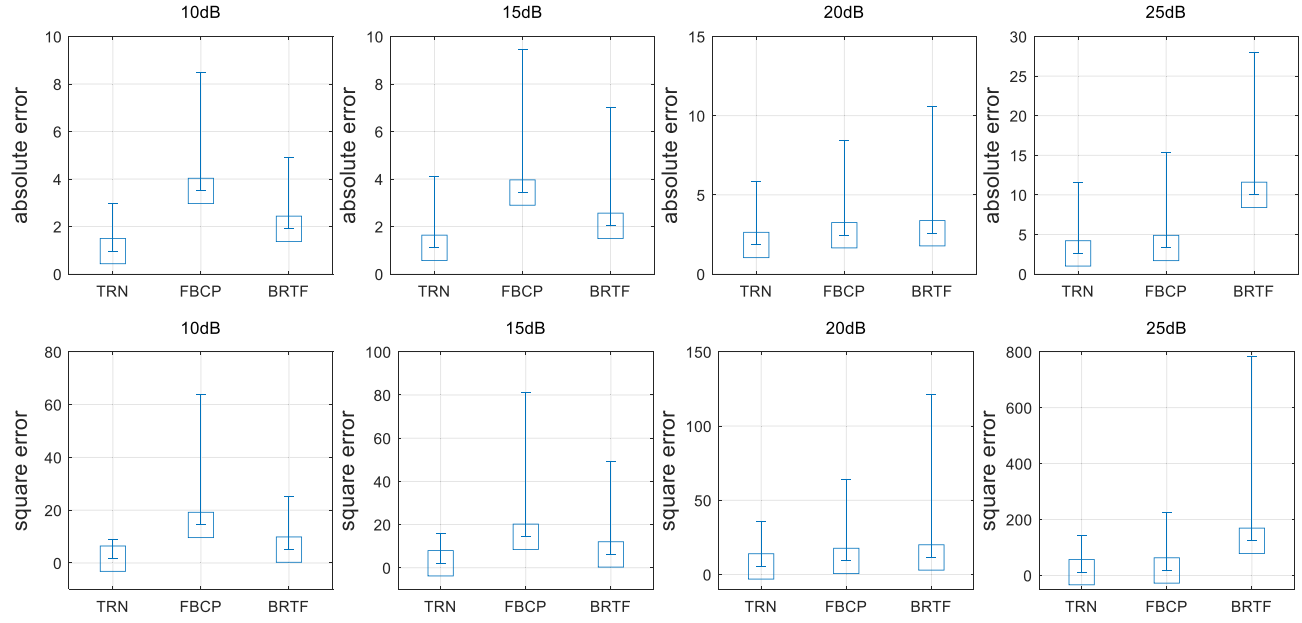


**Fig. 15.** Illustration of rank estimation accuracy of TRN, FBCP, BRTF on CMU datset at different noise levels.

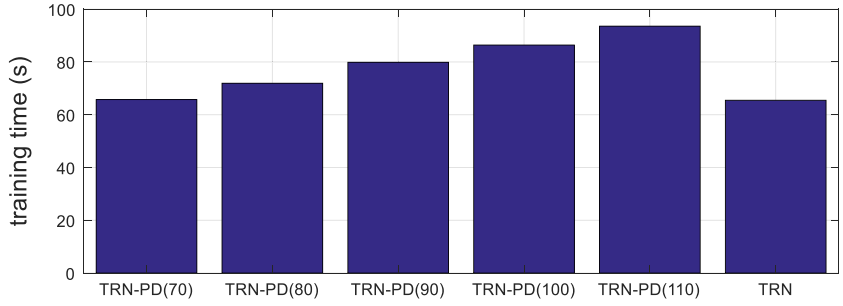


**Fig. 16.** Illustration of training time comparison between TRN and TRN-PD on real-world dataset with 10 dB noise.

## References


[1] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, H.A. Phan, Tensor decompositions for signal processing applications: From two-way to multiway component analysis, IEEE Signal Process. Mag. 32 (2) (2015) 145–163.

[2] N.D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E.E. Papalexakis, C. Faloutsos, Tensor decomposition for signal processing and machine learning, IEEE Trans. Signal Process. 65 (13) (2017) 3551–3582.

[3] U. Schollwöck, The density-matrix renormalization group in the age of matrix product states, Ann. Physics 326 (1) (2011) 96–192.

[4] T.G. Kolda, B.W. Bader, Tensor decompositions and applications, SIAM Rev. 51 (3) (2009) 455–500.

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

[6] R.A. Harshman, Foundations of the PARAFAC procedure: models and conditions for an "explanatory" multimodal factor analysis, 1970.

[7] L. Chen, Y. Liu, C. Zhu, Iterative block tensor singular value thresholding for extraction of low rank component of image data, in: The 42nd IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2017.

[8] J.B. Kruskal, Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics, Linear Algebra Appl. 18 (2) (1977) 95–138.

[9] J. Håstad, Tensor rank is NP-complete, J. Algorithms 11 (4) (1990) 644–654.

[10] C.J. Hillar, L.-H. Lim, Most tensor problems are NP-hard, J. ACM 60 (6) (2013) 45.

[11] K. Gregor, Y. LeCun, Infinite Tucker decomposition: Nonparametric Bayesian models for multiway data analysis, in: The 29th International Conference on Machine Learning, 2012, pp. 1675–1682.

[12] Q. Zhao, L. Zhang, A. Cichocki, Bayesian CP factorization of incomplete tensors with automatic rank determination, IEEE Trans. Pattern Anal. Mach. Intell. 37 (9) (2015) 1751–1763.

[13] Q. Zhao, G. Zhou, L. Zhang, A. Cichocki, S. Amari, Bayesian robust tensor factorization for incomplete multiway data, IEEE Trans. Neural Netw. Learn. Syst. 27 (4) (2016) 736–748.

[14] P. Rai, Y. Wang, S. Guo, G. Chen, D. Chen, L. Carin, Scalable Bayesian low-rank decomposition of incomplete multiway tensors, in: International Conference on Machine Learning, 2014, pp. 1800–1808.

[15] M. Mørup, L.K. Hansen, Automatic relevance determination for multi-way models, J. Chemometr. 23 (7–8) (2009) 352–363.

[16] T. Yokota, N. Lee, A. Cichocki, Robust multilinear tensor rank estimation using higher order singular value decomposition and information criteria, IEEE Trans. Signal Process. 65 (5) (2017) 1196–1206.

[17] K. Gregor, Y. LeCun, Learning fast approximations of sparse coding, in: Proceedings of the 27th International Conference on Machine Learning, ICML-10, 2010, pp. 399–406.

[18] P. Sprechmann, A.M. Bronstein, G. Sapiro, Learning efficient sparse and low rank models, IEEE Trans. Pattern Anal. Mach. Intell. 37 (9) (2015) 1821–1833.

[19] Z. Wang, Q. Ling, T. Huang, Learning deep l0 encoders, in: AAAI Conference on Artificial Intelligence, 2016, pp. 2194–2200.

[20] Z. Wang, Y. Yang, S. Chang, Q. Ling, T.S. Huang, Learning a deep $\ell_\infty$ encoder for hashing, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, AAAI Press, 2016, pp. 2174–2180.

[21] B. Xin, Y. Wang, W. Gao, D. Wipf, B. Wang, Maximal sparsity with deep networks?, in: Advances in Neural Information Processing Systems, 2016, pp. 4340–4348.

[22] N. Cohen, A. Shashua, Convolutional rectifier networks as generalized tensor decompositions, in: International Conference on Machine Learning, 2016, pp. 955–963.

[23] M. Che, A. Cichocki, Y. Wei, Neural networks for computing best rank-one approximations of tensors and its applications, Neurocomputing 267 (2017) 114–133.

[24] M. Borgerding, P. Schniter, S. Rangan, AMP-inspired deep networks for sparse linear inverse problems, IEEE Trans. Signal Process. 65 (16) (2017) 4293–4308.

[25] C. Metzler, A. Mousavi, R. Baraniuk, Learned D-AMP: Principled neural network based compressive image recovery, in: Advances in Neural Information Processing Systems, 2017, pp. 1770–1781.
[26] A. Lucas, M. Iliadis, R. Molina, A.K. Katsaggelos, Using deep neural networks for inverse problems in imaging: Beyond analytical methods, IEEE Signal Process. Mag. 35 (1) (2018) 20–36.
[27] Y. Liu, S. Wu, X. Huang, B. Chen, C. Zhu, Hybrid CS-DMRI: Periodic time-variant subsampling and omnidirectional total variation based reconstruction, IEEE Trans. Med. Imag. 36 (10) (2017) 2148–2159.
[28] S. Liang, T. Dresselaers, K. Louchami, C. Zhu, Y. Liu, U. Himmelreich, Comparison of different compressed sensing algorithms for low SNR 19F MRI applications—Imaging of transplanted pancreatic islets and cells labeled with perfluorocarbons, NMR Biomed. 30 (11) (2017).
[29] Y. Liu, M. De Vos, S. Van Huffel, Compressed sensing of multichannel EEG signals: the simultaneous cosparsity and low-rank optimization, IEEE Trans. Biomed. Eng. 62 (8) (2015) 2055–2061.
[30] Y. Liu, M. De Vos, I. Gligorijevic, V. Matic, Y. Li, S. Van Huffel, Multi-structural signal recovery for biomedical compressive sensing, IEEE Trans. Biomed. Eng. 60 (10) (2013) 2794–2805.
[31] Y. LeCun, Y. Bengio, et al., Convolutional networks for images, speech, and time series, The Handbook of Brain Theory and Neural Networks 3361 (10) (1995) 1995.
[32] S. Lawrence, C.L. Giles, A.C. Tsoi, A.D. Back, Face recognition: A convolutional neural-network approach, IEEE Trans. Neural Netw. 8 (1) (1997) 98–113.
[33] A. Krizhevsky, I. Sutskever, G. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[34] F.L. Hitchcock, The expression of a tensor or a polyadic as a sum of products, Stud. Appl. Math. 6 (1–4) (1927) 164–189.
[35] Y. LeCun, B.E. Boser, J.S. Denker, D. Henderson, R.E. Howard, W.E. Hubbard, L.D. Jackel, Handwritten digit recognition with a back-propagation network, in: Advances in Neural Information Processing Systems, 1990, pp. 396–404.
[36] T. Sim, S. Baker, M. Bsat, The CMU pose, illumination, and expression (PIE) database, in: The Fifth IEEE International Conference on Automatic Face and Gesture Recognition, IEEE, 2002, pp. 53–58.