Signal Processing 200 (2022) 108640

Contents lists available at ScienceDirect

Signal Processing

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

An interpretable bi-branch neural network for matrix completion

Xiao Peng Li[a,1], Maolin Wang[b,1], Hing Cheung So[a,2,*]

[a] *Department of Electrical Engineering, City University of Hong Kong, Hong Kong SAR, China*
[b] *School of Data Science, City University of Hong Kong, Hong Kong SAR, China*

ARTICLE INFO

Article history:
Received 29 December 2021
Revised 27 April 2022
Accepted 30 May 2022
Available online 2 June 2022

Keywords:
Low rank
Nonlinear matrix completion
Neural network
Image inpainting
Recommender system

ABSTRACT

The task of recovering a low-rank matrix given an incomplete matrix, also termed as matrix completion, arises in various applications. Methods for matrix completion can be classified into linear and nonlinear approaches. Despite the fact that the linear model provides basic theories ensuring restoring the missing entries with high probability, it has an obvious limitation that latent factors are restricted in the linear subspace. Thus, the nonlinear model has been suggested, which is mainly performed using neural networks. In this paper, a novel and interpretable neural network is developed for matrix completion. Different from existing neural networks whose structure is created by empirical design, the proposed version is devised via unfolding the matrix factorization formulation. Specifically, the two factors decomposed by matrix factorization construct the two branches of the suggested neural network, called bi-branch neural network (BiBNN). The row and column indices of each entry are considered as the input of the BiBNN, while its output is the estimated value of the entry. The training procedure aims to minimize the fitting error between all observed entries and their predicted values and then the unknown entries are estimated by inputting their coordinates into the trained network. The BiBNN is compared with state-of-the-art methods, including linear and nonlinear models, in processing synthetic data, image inpainting, and recommender system. Experimental results demonstrate that the BiBNN is superior to the existing approaches in terms of restoration accuracy.

1. Introduction

Matrix completion refers to predicting the unknown entries in an incomplete matrix using the low-rank property [1,2] and has been widely applied to various fields, such as image inpainting [3,4,55], recommender system [5,6], traffic sensing [7], system identification [8], target estimation [56] and multi-label image classification [9,10]. This is because lots of real-world data can be represented/approximated as low-rank matrices. For example, in recommender system, the user and item identity numbers are formulated as row and column indices of the matrix. Besides, since a customer does not rate all products in general, the matrix is incomplete. Moreover, the latent complete matrix is of low rank as the types of users and items are much less than the numbers of customers and products.

Over the past few years, numerous algorithms for matrix completion have been proposed, which can be classified into two categories: linear [11–21] and nonlinear models [22–27]. The linear model is the precursor and mainstay since it provides the basic theories that the missing entries of the incomplete matrix could be exactly restored with high probability under certain conditions [1,2]. In accordance with the linear model, matrix completion is formulated as a rank minimization problem subject to the constraint that the recovered entries are equal to the known elements in the observation set [2]. Since minimizing rank function is an NP-hard problem, practical methods try to handle its substitute. One efficient strategy is to convert the rank function as a constraint and then tackle the resultant problem. The representative methods involve singular value projection (SVP) [12], normalized iterative hard thresholding (NIHT) [13] and alternating projection (AP) [14]. Both SVP and NIHT are designed for noise free or Gaussian noise cases, while AP is able to deal with the data contaminated by impulsive noise. Since these approaches require computing truncated singular value decomposition (SVD), the selection of rank is a critical issue for their performance. Another scheme is to replace the rank function with the nuclear norm that has been proven being a convex envelop of the rank function [28]. Based on the nuclear norm, various algorithms have been developed, including singular value thresholding (SVT) [15], accelerated proximal gradient (APG) [16] and fixed point contin-

[*] Corresponding author.
E-mail addresses: x.p.li@my.cityu.edu.hk (X.P. Li), morin.w98@gmail.com (M. Wang), hcso@ee.cityu.edu.hk (H.C. So).
[1] Xiao Peng Li and Maolin Wang contributed equally in this work.
[2] EURASIP Member

https://doi.org/10.1016/j.sigpro.2022.108640
0165-1684/© 2022 Elsevier B.V. All rights reserved.

uation (FPC) [17]. Compared with the first strategy, the nuclear norm based methods need to perform full SVD and thus their computational complexity is higher. In addition, since the nuclear norm is equivalent to the sum of all singular values, it is slack relaxation to adopt the nuclear norm instead of the rank function, resulting in a seriously deviated solution from the ground truth. To tackle this issue, truncated nuclear norm regularization (TNNR) [18] suggests minimizing the remaining singular values after subtracting the largest ones from all singular values. Besides, Gu *et al*. [19] propose a weighted nuclear norm that assigns different weights to the whole singular values. Moreover, Schatten p-norm with $p \in (0,1]$ is developed to replace the nuclear norm [29,30]. When $p=1$, Schatten p-norm is equal to the nuclear norm, while it becomes the strict envelope of the rank function at $p \to 0$. It is worth noting that matrix completion schemes based on these improved norms still require computing SVD. Although they do not need to set an appropriate rank, the rank of the objective matrix is still affected by a user-defined regularization parameter.

One prevalent way to circumvent the time-consuming SVD is to adopt matrix factorization technique which decomposes the objective matrix into two small-size matrices and then substitute their product for the large-size matrix. Based on the idea of the matrix factorization, subspace evolution and transfer (SET) [31], low-rank matrix fitting (LMaFit) [20], alternating minimization (AltMinComplete) [21] and proximal alternating minimization (PAM) [32] are developed. Compared with SET, LMaFit, and AltMinComplete, PAM is able to converge to a second-order stationary point under some mild conditions with two randomly initialized small-size matrices. The main challenge of this strategy is to select the best rank since the performance depends on the adopted rank. To deal with this issue, rank-one matrix pursuing technique is considered, including orthogonal rank-one matrix pursuit (OR1MP) [33] and economic OR1MP (EOR1MP) [34], adaptive basis selection strategy (ABSS) [35] and ℓ_1-norm regularized rank-one matrix completion (L1MC) [36]. Herein, the objective matrix is decomposed into a sum of rank-one matrices, and the number of rank-one matrices is determined by a user-defined accuracy threshold.

Although the linear model has achieved excellent performance in many applications, it has an obvious limitation that the latent factors are restricted in the linear subspace, resulting in a small feasible region. The superiority of the nonlinear model over the linear one has been demonstrated in emotion recognition [24], image inpainting [26], collaborative filtering [37] and multi-label and multi-class classification [38]. One simple method for nonlinear matrix completion is to leverage a nonlinear kernel [23,24,39,40], which projects the data into a linear space via a nonlinear kernel and then performs matrix completion in the linear space. Although this strategy is easy to implement, the adopted kernel needs to design elaborately.

Another technique is to utilize neural networks since the activation function is able to represent the nonlinear relationship. Li and Wang propose an adaptive and implicit regularization neural network (AIR-Net) for nonlinear matrix completion [41]. Despite the good performance of the AIR-Net on image inpainting in the absence of noise, it can only handle square matrices, that is, its applicability is significantly restricted. Besides, Fan and Chow suggest an autoencoder based network where a series of autoencoders is trained sequentially using the observed entries and then all individual autoencoders are stacked together as a deep autoencoder for predicting missing entries after fine-tuning [42]. Thereafter, an improved neural network, termed as deep matrix factorization (DMF), is developed [43]. Compared with the autoencoder based network, DMF utilizes the matrix factorization scheme to design a deep neural network without the encoder procedure wherein the input and output are the low-dimensional unknown latent and partially known variables, respectively. It is well known that overfitting is a common issue for neural networks. To avoid this problem, Mercier and Uysal propose adopting Bayesian regularization and early-stopping strategy to improve the neural network performance [44]. In addition, a one-layer neural network based on nonlinear inductive matrix completion is suggested for recommender system where the predicted rating is modeled as an inner product of the two projected user and item feature vectors on the latent space [25]. Furthermore, it is verified that the neural network using the low-rank property is more effective than general neural networks [25]. Moreover, patch-based nonlinear matrix completion (PNMC) utilizes convolutional neural networks [26] via dividing the observed matrix into small-size patches. The small-patch strategy of the PNMC is shown to be able to efficiently exploit the locality among adjacent entries, and reduce the size of the deep neural network. Convolutional neural tangent kernel (CNTK) develops an infinite width neural network based on the tangent kernel and achieves good performance on image inpainting in the noiseless condition [45]. However, its memory cost is extremely high, e.g., creating a kernel for processing a 349×366 matrix requires more than 60 GiB memory. On the other hand, some works leverage graph theory to design neural networks for matrix completion based recommender system [27,46–48] where users and items are represented as graphs.

Although the above-mentioned neural network based methods attain satisfying performance, their network structures cannot be interpreted, that is, the architecture is created by empirical design. For instance, the objective matrix is decomposed into two small-size matrices in [42,43], but the recovered matrix is still full rank. Besides, AIR-Net, PNMC, and CNTK do not leverage the low-rank property and thus it is difficult to explain why they can recover the missing entries in theory.

In this paper, we devise a bi-branch neural network using fully-connected layers for nonlinear matrix completion and term it as BiBNN. The BiBNN is designed exploiting matrix factorization unfolding in which one branch denotes a factorized component. In addition, the dimensions of the last hidden layer are controlled to attain the desired rank. Moreover, the processing target of the proposed neural network is individual entry instead of the entire incomplete matrix. Specifically, one element and its coordinates are considered as the output and input of the BiBNN, respectively. As usual, the activation function between two successive hidden layers is used to introduce nonlinearity. Our main contributions are summarized as:

1. We design a fully-connected neural network via unfolding matrix factorization formulation to address matrix completion such that the BiBNN can be interpreted.
2. The computational and space complexities of the BiBNN are analyzed. In addition, we prove that the proposed neural network solving by gradient descent is convergent.
3. The proposed BiBNN exhibits better performance than several state-of-the-art methods, including linear and nonlinear models, in processing synthetic and real-world data, viz. images and recommender system.

The manuscript is outlined as follows. In Section 2, the matrix completion problem is formulated, and the matrix factorization scheme is reviewed. Besides, a similar neural network is introduced for comparing with our BiBNN. The BiBNN for matrix completion is derived in Section 3.

Empirical evaluation of the developed neural network using both synthetic and real-world data is provided in Section 4 to demonstrate its superiority over seven existing approaches in terms of recovery accuracy. Finally, Section 5 provides concluding remarks.

2. Problem formulation and related works

2.1. Matrix completion formulation

Let $\boldsymbol{X}_\Omega \in \mathbb{R}^{m\times n}$ be an observed matrix with missing entries wherein $\boldsymbol{X}_\Omega$ indicates that $\boldsymbol{X}$ projects on the binary matrix $\Omega \in \mathbb{R}^{m\times n}$, comprised of 0 and 1, which corresponds to unobserved and observed elements, respectively, resulting in

$$(\boldsymbol{X}_\Omega)_{i,j} = \begin{cases} x_{i,j}, & \text{if } \Omega_{i,j}=1 \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

where $(\boldsymbol{X}_\Omega)_{i,j}$ is the (i,j) entry of $\boldsymbol{X}_\Omega$. Conceptually, matrix completion is formulated as a rank minimization problem [2]

$$\min_{\boldsymbol{M}} \operatorname{rank}(\boldsymbol{M}) \text{ s.t. } \boldsymbol{X}_\Omega = \boldsymbol{M}_\Omega, \tag{2}$$

that is, the matrix completion aims to seek $\boldsymbol{M} \in \mathbb{R}^{m\times n}$ with the minimum rank under the condition that the elements of the restored and observed matrices in the observation set are equal. Unfortunately, (2) is an NP-hard problem since the rank function is both nonconvex and discrete. A feasible strategy is to substitute the rank function with the nuclear norm [12–14], corresponding to the following optimization problem

$$\min_{\boldsymbol{M}} \|\boldsymbol{M}\|_* \text{ s.t. } \boldsymbol{X}_\Omega = \boldsymbol{M}_\Omega, \tag{3}$$

where $\|\boldsymbol{M}\|_*$ is the nuclear norm which equals the sum of all singular values of $\boldsymbol{M}$. Since SVD is performed in each iteration to solve (3), the computational complexity is high, especially for big matrices.

Another prevailing method is to exploit the matrix factorization technique which decomposes the objective matrix $\boldsymbol{M}$ into two small-size matrices using the prior rank information, leading to

$$\min_{\boldsymbol{U},\boldsymbol{V}} \|(\boldsymbol{U}\boldsymbol{V}^T)_\Omega - \boldsymbol{X}_\Omega\|_F^2, \tag{4}$$

where $(\cdot)^T$ signifies the transpose operator, $\|\cdot\|_F$ denotes the Frobenius norm, $\boldsymbol{U}\in\mathbb{R}^{m\times r}$ and $\boldsymbol{V}\in\mathbb{R}^{n\times r}$. Herein, r is the rank of the objective matrix. For the situation of unknown rank, we provide a strategy to estimate its value, which is introduced in the next section. Since (4) avoids computing SVD, the methods to handle (4) have much lower computational complexity than those for (3). After seeking $\boldsymbol{U}$ and $\boldsymbol{V}$, the target matrix can be determined as $\boldsymbol{M}=\boldsymbol{U}\boldsymbol{V}^T$.

2.2. Previous works

It is worth mentioning that a two-stream neural network has been developed for nonlinear matrix completion, termed as neural matrix completion (NMC) [49]. Its architecture is designed based on the following formulation

$$m_{i,j} = f(\boldsymbol{r}_i^T, \boldsymbol{c}_j), \tag{5}$$

where $\boldsymbol{r}_i^T \in \mathbb{R}^m$ and $\boldsymbol{c}_j\in\mathbb{R}^n$ represent the ith row and jth column of $\boldsymbol{M}$, respectively. Specifically, $\boldsymbol{r}_i$ and $\boldsymbol{c}_j$ are the inputs of the two branches of NMC, and the outputs of two streams are $\boldsymbol{a}_i\in\mathbb{R}^r$ and $\boldsymbol{b}_j\in\mathbb{R}^r$, respectively. The neural network structure of the two branches consists of several fully-connected layers. Furthermore, the output of NMC neural network is $m_{i,j}$, calculated by

$$m_{i,j} = \frac{\boldsymbol{a}_i^T\boldsymbol{b}_j}{\|\boldsymbol{a}_i\|_2\|\boldsymbol{b}_j\|_2}, \tag{6}$$

where $\|\cdot\|_2$ denotes the ℓ_2-norm. The training data are all observed entries $x_{i,j}$ with $\Omega_{i,j}=1$, and then $x_{i,j}$ with $\Omega_{i,j}=0$ is predicted by the trained neural network. Although its performance is satisfactory, it is difficult to explain why $m_{i,j}$ can be predicted using $\boldsymbol{r}_i$ and $\boldsymbol{c}_j$ that are the ith row and jth column of the incomplete matrix, respectively.

3. Bi-Branch neural network

In this section, we design the BiBNN using the unfolding method that unrolls an iterative optimization algorithm to a neural network hierarchical architecture. Moreover, the convergence and complexity of the BiBNN are discussed.

3.1. Structure design

Prior to presenting the neural network, we introduce a set Φ involving the coordinates of the known entries in $\boldsymbol{X}_\Omega$, defined as

$$\Phi = \{(i,j)|\Omega_{i,j}=1\}. \tag{7}$$

Then, we use Φ to reformulate (4) as scalar form:

$$\min_{\boldsymbol{U},\boldsymbol{V}} \sum_{(i,j)\in\Phi} \left((\boldsymbol{U}\boldsymbol{V}^T)_{i,j} - x_{i,j}\right)^2. \tag{8}$$

In accordance with the matrix multiplication, we know that the (i,j) entry of $\boldsymbol{U}\boldsymbol{V}^T$ is equal to the product between the ith row of $\boldsymbol{U}$ and jth row of $\boldsymbol{V}$ that are signified by $\boldsymbol{u}_i^T$ and $\boldsymbol{v}_i^T$, respectively. By the representation of $m_{i,j}=\boldsymbol{u}_i^T\boldsymbol{v}_j$, (8) is re-expressed as

$$\min_{\boldsymbol{u}_i,\boldsymbol{v}_j} \sum_{(i,j)\in\Phi} \left(m_{i,j}-x_{i,j}\right)^2 \text{ s.t. } m_{i,j}=\boldsymbol{u}_i^T\boldsymbol{v}_j. \tag{9}$$

To clearly describe the network structure by making use of mathematical representation, we introduce two auxiliary parameters, namely, $\boldsymbol{\alpha}_i\in\mathbb{R}^m$ with $i\in[1,m]$ and $\boldsymbol{\beta}_j\in\mathbb{R}^n$ with $j\in[1,n]$. As basis vectors, only the ith and jth entries of $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_j$ are equal to 1, while the other elements are 0. In other words, $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_j$ indicate the location of $m_{i,j}$ in $\boldsymbol{M}$. Employing $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_j$, $m_{i,j}$ is rewritten as

$$m_{i,j} = (\boldsymbol{U}^T\boldsymbol{\alpha}_i)^T(\boldsymbol{V}^T\boldsymbol{\beta}_j). \tag{10}$$

Then, we plug (10) into (9) to obtain

$$\min_{\boldsymbol{U},\boldsymbol{V}} \sum_{(i,j)\in\Phi} \left((\boldsymbol{U}^T\boldsymbol{\alpha}_i)^T(\boldsymbol{V}^T\boldsymbol{\beta}_j)-x_{i,j}\right)^2. \tag{11}$$

We first consider unfolding (11) to construct a one-hidden-layer linear BiBNN for matrix completion and then improve it for the nonlinear model. For the linear BiBNN, $\boldsymbol{U}^T\boldsymbol{\alpha}_i$ and $\boldsymbol{V}^T\boldsymbol{\beta}_j$ are considered as two independent fully-connected layers. Herein, “independent” means that the two fully-connected layers do not share the weight matrix. Specifically, $\boldsymbol{U}^T\boldsymbol{\alpha}_i$ is considered an input $\boldsymbol{\alpha}_i$ multiplied by a weight matrix $\boldsymbol{U}$ and then the output is $\boldsymbol{u}_i=\boldsymbol{U}^T\boldsymbol{\alpha}_i$. Similarly, $\boldsymbol{V}^T\boldsymbol{\beta}_j$ is arranged as a fully-connected layer with input $\boldsymbol{\beta}_j$ and output $\boldsymbol{v}_j=\boldsymbol{V}^T\boldsymbol{\beta}_j$. Furthermore, the output of the linear BiBNN is computed as $m_{i,j}=\boldsymbol{u}_i^T\boldsymbol{v}_j$, and the corresponding fitting target is the observed $x_{i,j}$.

It is worth mentioning that the weight between the fully-connected and output layers is a constant, namely, 1, due to $m_{i,j}=\boldsymbol{u}_i^T\boldsymbol{v}_j$.

It is well known that multiple-layer neural networks generally outperform the single-layer configurations. As an illustration, we further decompose $\boldsymbol{U}$ and $\boldsymbol{V}$ for constructing a L-hidden-layer neural network as

$$\boldsymbol{U} = \boldsymbol{U}_1\cdots\boldsymbol{U}_l\cdots\boldsymbol{U}_L, \tag{12}$$

$$\boldsymbol{V} = \boldsymbol{V}_1\cdots\boldsymbol{V}_l\cdots\boldsymbol{V}_L, \tag{13}$$

where $\boldsymbol{U}_l\in\mathbb{R}^{m_l\times m_{l+1}}$ and $\boldsymbol{V}_l\in\mathbb{R}^{n_l\times n_{l+1}}$ for $l\in[1,L]$ with $m_1=m$, $n_1=n$ and $m_{L+1}=n_{L+1}=r$. Herein, m_l and n_l with $l\in[2,L]$ are the node numbers of the hidden layers. Based on this deep factorization model, we have

$$\boldsymbol{U}^T\boldsymbol{\alpha}_i = (\boldsymbol{U}_1\cdots\boldsymbol{U}_l\cdots\boldsymbol{U}_L)^T\boldsymbol{\alpha}_i = \boldsymbol{U}_L^T\cdots\boldsymbol{U}_l^T\cdots\boldsymbol{U}_1^T\boldsymbol{\alpha}_i, \tag{14}$$

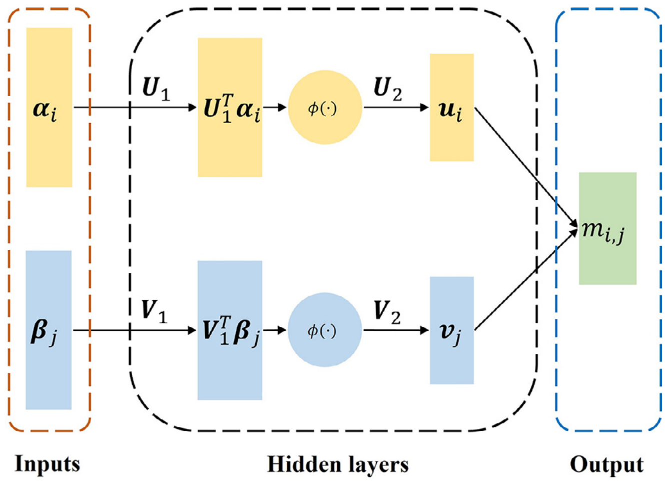


Fig. 1. Illustration of two-hidden-layer BiBNN architecture.

$$V^T\beta_j = (V_1\cdots V_l\cdots V_L)^T\beta_j = V_L^T\cdots V_l^T\cdots V_1^T\beta_j. \tag{15}$$

Then, plugging (14) and (15) into (11) results in

$$\min_{U_l,V_l,l\in[1,L]}\sum_{(i,j)\in\Phi}\left(\left(U_L^TU_{L-1}^T\cdots U_1^T\alpha_i\right)^T\left(V_L^TV_{L-1}^T\cdots V_1^T\beta_j\right)-x_{i,j}\right)^2. \tag{16}$$

However, (16) and (11) are equivalent as (12) and (13) are linear transforms. To proceed, as in typical neural networks, we insert a nonlinear function between two layers, leading to

$$\min_{U_l,V_l,l\in[1,L]}\sum_{(i,j)\in\Phi}((U_L^T\phi(U_{L-1}^T\cdots\phi(U_1^T\alpha_i)))^T(V_L^T\phi(V_{L-1}^T\cdots\phi(V_1^T\beta_j)))-x_{i,j})^2, \tag{17}$$

where $\phi(\cdot)$ is the nonlinear activation function.

Similar to the linear BiBNN, $U_L^T\phi(U_{L-1}^T\cdots\phi(U_1^T\alpha_i))$ and $V_L^T\phi(V_{L-1}^T\cdots\phi(V_1^T\beta_j))$ are constructed as two independent multiple-layer fully-connected neural networks that are considered as the two branches of the BiBNN. To be more specific, α_i is arranged as the input of the first hidden layer with the activation function in the first branch, and its output is $\phi(U_1^T\alpha_i)$. Then, the input and output of the second hidden layer are set as $\phi(U_1^T\alpha_i)$ and $\phi(U_2^T\phi(U_1^T\alpha_i))$. Accordingly, for the lth hidden layer with $l\in[2,L-1]$, its input is $\phi(U_{l-1}^T\cdots\phi(U_1^T\alpha_i))$, while its output is $\phi(U_l^T\phi(U_{l-1}^T\cdots\phi(U_1^T\alpha_i)))$. Finally, the output of the first branch is set as $u_i=U_L^T\phi(U_{L-1}^T\cdots\phi(U_1^T\alpha_i))$. For the second branch, the input of the first hidden layer is β_j, while its output is $\phi(V_1^T\beta_j)$. Without loss of generality, for the lth hidden layer with $l\in[2,L-1]$, its input and output are $\phi(V_{l-1}^T\cdots\phi(V_1^T\beta_j))$ and $\phi(V_l^T\phi(V_{l-1}^T\cdots\phi(V_1^T\beta_j)))$, respectively. The output of the second branch is arranged as $v_j=V_L^T\phi(V_{L-1}^T\cdots\phi(V_1^T\beta_j))$, and then the output of the BiBNN is $m_{i,j}=u_iv_j$.

Training the neural network constructed by (17) is to update U_l, V_l with $l\in[1,L]$ to minimize the objective function in (17) since the weight matrices of the neural network are U_l, V_l with $l\in[1,L]$. In addition, we define $\widehat{U}=\phi(\phi(U_1)\cdots U_{L-1})U_L$ and $\widehat{V}=\phi(\phi(V_1)\cdots V_{L-1})V_L$. Comparing $M=UV^T$ and $\widehat{M}=\widehat{U}\widehat{V}^T$, although both are of low rank, M is sought from the linear subspace, while $\widehat{M}$ is computed in the nonlinear space.

Fig. 1 illustrates the two-hidden-layer BiBNN architecture adopted in our experiments. We observe that it has two branches, and each branch involves two hidden layers. Matrices on the lines correspond to the weight matrices between two layers, while the formulations in the rectangular blocks denote layers' outputs. In addition, the circles indicate the activation function. In accordance with the depicted BiBNN, we provide the pseudo code in Algorithm.

Algorithm 1 BiBNN for matrix completion.

```
Require: $x_{i,j}$ with $(i,j)\in\Phi$ and $K_{\max}$.
  Initialize: Initialize $U_1^1$, $U_2^1$, $V_1^1$ and $V_2^1$ in $\mathcal{U}(-0.5,0.5)$.
  for $k=1,2,\cdots,K_{\max}$ do
      1. Update $U_2^{k+1}$ with fixing $U_1^k$, $V_1^k$ and $V_2^k$.
      2. Update $U_1^{k+1}$ with fixing $U_2^{k+1}$, $V_1^k$ and $V_2^k$.
      3. Update $V_2^{k+1}$ with fixing $U_2^{k+1}$, $U_1^{k+1}$ and $V_1^k$.
      4. Update $V_1^{k+1}$ with fixing $U_2^{k+1}$, $U_1^{k+1}$ and $V_2^{k+1}$.
      Stop if stopping criterion is met.
  end for
Ensure: $\widehat{M}=(\phi(U_1)U_2)(\phi(V_1)V_2)^T$.
```

3.2. Convergence analysis

For concise expression, the convergent property is analyzed based on a two-hidden-layer BiBNN. It is worth mentioning that the convergence analysis is applicable for BiBNN with more than two hidden layers.

According to the derivation of (17), we have

$$\begin{aligned}&\min_{U_1,U_2,V_1,V_2}\|((\phi(U_1)U_2)(\phi(V_1)V_2)^T)_\Omega-X_\Omega\|_F^2\\ :=&\min_{U_1,U_2,V_1,V_2}\|((\phi(U_1)U_2)(\phi(V_1)V_2)^T-X)\odot\Omega\|_F^2,\end{aligned} \tag{18}$$

where $\odot$ is the entry-wise product. Besides, we define the loss function of the objective function as

$$\mathcal{L}(U_1,U_2,V_1,V_2)=\|((\phi(U_1)U_2)(\phi(V_1)V_2)^T-X)\odot\Omega\|_F^2. \tag{19}$$

It is well known that neural networks are typically optimized using the gradient descent method or its variants. Hence, we analyze convergence based on the gradient descent, leading to the following procedure:

$$\begin{aligned}U_2^{k+1}&=U_2^k-\lambda\frac{\partial\mathcal{L}(U_1^k,U_2^k,V_1^k,V_2^k)}{\partial U_2}\\&=U_2^k-2\lambda\phi(U_1^k)^T(((\phi(U_1^k)U_2^k)(\phi(V_1^k)V_2^k)^T-X)\odot\Omega)\phi(V_1^k)V_2^k,\end{aligned} \tag{20}$$

$$\begin{aligned}U_1^{k+1}&=U_1^k-\lambda\frac{\partial\mathcal{L}(U_1^k,U_2^{k+1},V_1^k,V_2^k)}{\partial U_1}\\&=U_1^k-2\lambda((((\phi(U_1^k)U_2^{k+1})(\phi(V_1^k)V_2^k)^T-X)\odot\Omega)\phi(V_1^k)V_2^k(U_2^{k+1})^T)\\&\quad\odot\phi'(U_1^k),\end{aligned} \tag{21}$$

$$\begin{aligned}V_2^{k+1}&=V_2^k-\lambda\frac{\partial\mathcal{L}(U_1^{k+1},U_2^{k+1},V_1^k,V_2^k)}{\partial V_2}\\&=V_2^k-2\lambda\phi(V_1^k)^T(((\phi(U_1^{k+1})U_2^{k+1})(\phi(V_1^k)V_2^k)^T-X)\\&\quad\odot\Omega)^T\phi(U_1^{k+1})U_2^{k+1},\end{aligned} \tag{22}$$

$$\begin{aligned}V_1^{k+1}&=V_1^k-\lambda\frac{\partial\mathcal{L}(U_1^{k+1},U_2^{k+1},V_1^k,V_2^{k+1})}{\partial V_1}\\&=V_1^k-2\lambda((((\phi(U_1^{k+1})U_2^{k+1})(\phi(V_1^k)V_2^{k+1})^T-X)\\&\quad\odot\Omega)^T\phi(U_1^{k+1})U_2^{k+1}(V_2^{k+1})^T)\odot\phi'(V_1^k),\end{aligned} \tag{23}$$

where $\lambda>0$ is the learning rate parameter and $\phi'(\cdot)$ is the derivative of $\phi(\cdot)$. It is worth pointing out that the optimization problem is nonconvex w.r.t U_1, U_2, V_1 and V_2, but it is convex combined

Table 1
Comparison of different methods on synthetic data in Gaussian noise of 20dB.

Method	Proposed	PAM	MC-DMF	MC-IALM	NLMC	PMC	CNTK	CNTK$^+$
MSE	0.0509	0.0712	0.0623	0.5955	0.0794	0.0854	0.2147	0.1513

with a convex $\phi(\cdot)$ w.r.t. one with fixing remaining variables. Besides, $\phi(\cdot)$ can be nonconvex as local convexity is adequate to seek a local solution. Since (20), (21), (22) and (23) leverage the gradient descent to update $\boldsymbol{U}_2^k$, $\boldsymbol{U}_1^k$, $\boldsymbol{V}_2^k$ and $\boldsymbol{V}_1^k$, respectively, we can attain the following inequality with an adequately small learning rate

$$\begin{aligned}\mathcal{L}(\boldsymbol{U}_1^k,\boldsymbol{U}_2^k,\boldsymbol{V}_1^k,\boldsymbol{V}_2^k) &\geq \mathcal{L}(\boldsymbol{U}_1^k,\boldsymbol{U}_2^{k+1},\boldsymbol{V}_1^k,\boldsymbol{V}_2^k) \geq \mathcal{L}(\boldsymbol{U}_1^{k+1},\boldsymbol{U}_2^{k+1},\boldsymbol{V}_1^k,\boldsymbol{V}_2^k)\\ &\geq \mathcal{L}(\boldsymbol{U}_1^{k+1},\boldsymbol{U}_2^{k+1},\boldsymbol{V}_1^k,\boldsymbol{V}_2^{k+1}) \geq \mathcal{L}(\boldsymbol{U}_1^{k+1},\boldsymbol{U}_2^{k+1},\boldsymbol{V}_1^{k+1},\boldsymbol{V}_2^{k+1}),\end{aligned} \tag{24}$$

that is, $\mathcal{L}(\boldsymbol{U}_1^k,\boldsymbol{U}_2^k,\boldsymbol{V}_1^k,\boldsymbol{V}_2^k)$ updated by the gradient descent is nonincreasing. In addition, it is easy to know $\mathcal{L}(\boldsymbol{U}_1^k,\boldsymbol{U}_2^k,\boldsymbol{V}_1^k,\boldsymbol{V}_2^k) \geq 0$, indicating that the loss function has a lower bound. Thereby, the sequence of the objective values $\{\mathcal{L}(\boldsymbol{U}_1^k,\boldsymbol{U}_2^k,\boldsymbol{V}_1^k,\boldsymbol{V}_2^k)\}$ is convergent.

3.3. Complexity analysis

We first analyze the space complexity. Consider a BiBNN network with two hidden layers whose node numbers are q and r, respectively. The first branch requires storing $mq+qr$ elements, and the entries of the second branch are $nq+qr$. Thereby, the total number of entries is $(m+n+2r)q$, resulting in the space complexity of $\mathcal{O}((m+n)q)$ due to $r \ll \min(m,n)$.

Then the computational complexity is analyzed, including forward and backward propagation. For the forward propagation, the complexity of computing one input is $\mathcal{O}((m+n)(r+1)q)$. Thereby, the total computational complexity for one epoch is $\mathcal{O}((m^2n+mn^2)(r+1)qp)$ where p denotes the percentage of the observed entries. On the other hand, the complexity of the backward propagation can be calculated according to (20), (21), (22) and (23), specifically $\mathcal{O}(mqr+nqr+mnr)$ for (20), $\mathcal{O}(mqr+nqr+mn(r+q)$ for (21), $\mathcal{O}(mqr+nqr+mnr)$ for (22) and $\mathcal{O}(mqr+nqr+mn(r+q))$ for (23). As a result, the total computational complexity is $\mathcal{O}(mn(r+q))$ since $\mathcal{O}(mn)$ dominates $\mathcal{O}(mq)$ and $\mathcal{O}(nq)$.

3.4. Rank selection

It is clear that the BiBNN performance is affected by the selected rank. If the true rank of the objective matrix is unknown, we suggest leveraging the cross-validation method to search for the best rank [50]. First, Ω is divided into two subsets such that $\Omega_1+\Omega_2=\Omega$ and $\|\Omega_1\|_1/\|\Omega\|_1=0.95$. Herein, $\|\Omega\|_1$ is the ℓ_1-norm of Ω, which equals the number of observed entries in the incomplete matrix. Then, $\boldsymbol{X}_{\Omega_1}$ is adopted to train the BiBNN, while $\boldsymbol{X}_{\Omega_2}$ is used to test the neural network. Given a rank, one BiBNN can be trained based on $\boldsymbol{X}_{\Omega_1}$, and the corresponding prediction error can be computed on $\boldsymbol{X}_{\Omega_2}$. After trying different ranks, the best rank is determined by the one with the smallest test error.

3.5. Data preprocessing

Since the input data are not directly collected from the observed matrix, preprocessing is required. Given an incomplete matrix $\boldsymbol{X}_\Omega$, we extract all observed entries as well as compute the corresponding $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_j$ to attain the training dataset. Herein, one entry, associating with its $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_j$, are considered as a group of training data.

4. Experimental results

The proposed BiBNN is compared with seven representative methods, namely, PAM [32], MC-DMF [43], MC-IALM [11], NLMC [39], PMC [40], CNTK [45] and CNTK$^+$ [45]. Our neural network is programmed using the PyTorch framework [51], and implemented on a personal computer with Nvidia 2060 GPU.

4.1. Network setting

For the adopted BiBNN in all experiments, the number of hidden layers is two, and the activation function is selected as exponential linear unit (ELU). It is easy to know that the numbers of nodes in the output and last hidden layer are 1 and r, respectively. Besides, the first hidden layer contains $50+r$ nodes. The size of input layer is determined based on the dimensions of the observed matrix. Moreover, the loss function is based on the ℓ_2-norm. For the adopted neural network, according to (17), it is $\sum_{(i,j)\in\Phi}\left((\boldsymbol{U}_2^T\phi(\boldsymbol{U}_1^T\boldsymbol{\alpha}_i))^T(\boldsymbol{V}_2^T\phi(\boldsymbol{V}_1^T\boldsymbol{\beta}_j))-x_{i,j}\right)^2$, while the optimization solver is chosen as adaptive moment estimation function (Adam) with learning rate of 0.01. Moreover, all training data are utilized for one epoch, and the number of epochs is set to 2000.

4.2. Synthetic data

We first test all methods on the synthetic data with a fixed rank. The noise-free complete matrix $\boldsymbol{X}$ is generated by the product of $\boldsymbol{X}_1 \in \mathbb{R}^{150\times 15}$ and $\boldsymbol{X}_2 \in \mathbb{R}^{15\times 160}$ whose entries obey independent standard Gaussian normal distribution. Note that the dimensions of $\boldsymbol{X}$ are selected as 150×160 because the CNTK cannot process a matrix whose dimensions are larger than 200×200 on general PCs. It is clear that the true rank is 15. To approximate practical singular value distribution, the ith singular value of $\boldsymbol{X}$ is set as 2^{10-i} with $i\in[1,15]$ [34]. The binary matrix Ω consists of the same numbers of 1 and 0 where all entries are randomly distributed with missing observation ratio of 50%. In addition, the incomplete matrix is generated by

$$\boldsymbol{X}_\Omega = \boldsymbol{X}\odot\Omega+\boldsymbol{N}_\Omega \tag{25}$$

where $\boldsymbol{N}_\Omega$ contains additive white Gaussian noise whose intensity is quantified by signal-to-noise ratio (SNR)

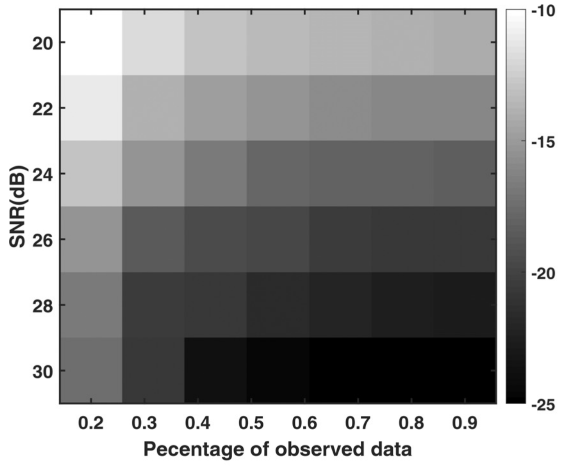
$$\mathrm{SNR}=\frac{\|\boldsymbol{X}\odot\Omega\|_F^2}{\|\boldsymbol{N}_\Omega\|_F^2}. \tag{26}$$

To evaluate the recovery performance, the mean square error (MSE) is adopted, defined as

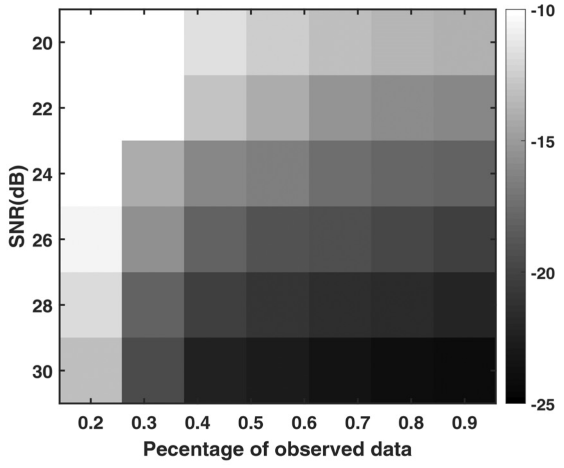
$$\mathrm{MSE}=\frac{\|\boldsymbol{M}-\boldsymbol{X}\|_F^2}{mn} \tag{27}$$

where $\boldsymbol{M}$ is the restored matrix. It is clear that a small value of MSE indicates good recovery performance.

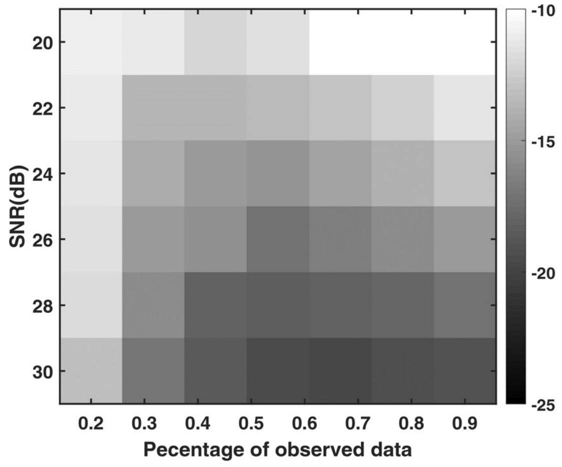
Table 1 tabulates the results on the synthetic data in Gaussian noise of 20dB by different approaches. We see that the MSE of the BiBNN is smaller than those of PAM, MC-DMF, MC-IALM, NLMC, PMC, CNTK, and CNTK$^+$. Hence, our method is superior to these existing approaches. It is worth noting that the recovered matrices by the BiBNN, PAM, and MC-IALM are of low rank, while those of the remaining schemes are of full rank. This may be the reason for the inferior performance of MC-DMF, NLMC, PMC, CNTK, and CNTK$^+$.



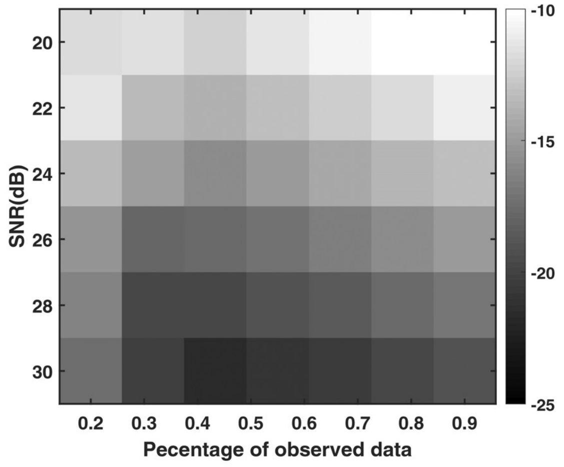
(a) Proposed



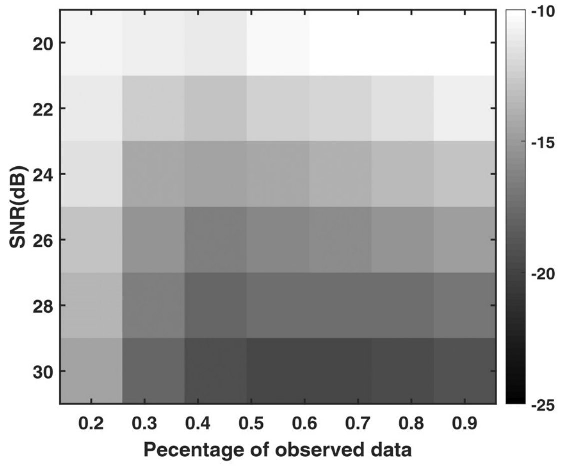
(b) PAM



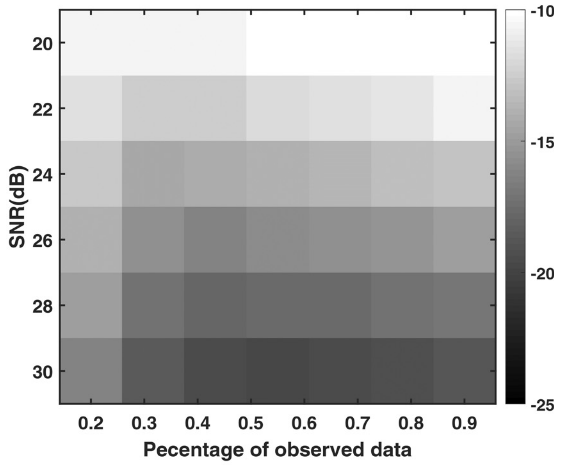
(c) MC-DMF



(d) MC-IALM



(e) NLMC



(f) PMC

Fig. 2. Phase transition of MSE versus observed percentage and SNR.

The phase transition, i.e., MSE versus the observation percentage and SNR, is depicted in Fig. 2. The performance of CNTK and CNTK$^+$ is not satisfactory, and their results will lead to a wide range in the colorbar of the figure, which is not conducive to contrast the other six algorithms. Thereby, the phase transition figures of CNTK and CNTK$^+$ are not included. We see that the proposed method and PAM are superior because they have more dark areas than MC-DMF, MC-IALM, NLMC and PMC. Comparing our approach with PAM, the former attains better performance than the latter in the low observation percentages. In addition, MC-DMF, MC-IALM, NLMC and PMC produce a phenomenon where the best performance is not in the highest observation percentage under

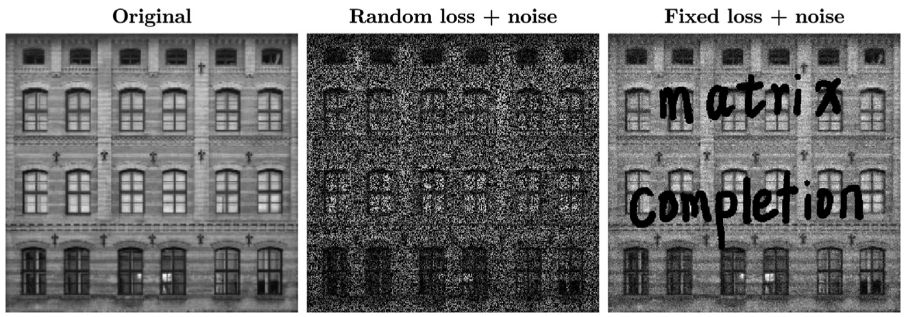


Fig. 3. Original *windows*, and two incomplete images with different masks in Gaussian noise with $\sigma^2 = 0.005$.

the same SNR. The reason may be that they cannot resist noise. A higher observation percentage implies more noise in the given data, resulting in performance degradation. In contrast, our method and PAM are able to restrain noise.

4.3. Image inpainting

Gray-scale image inpainting is one popular application of matrix completion since a gray-scale image can be straightforwardly represented as a matrix that meets the approximately low-rank property. Images, in practice, may not be fully captured because of the shadow from other objects or damage to the photosensitive device. Besides, wireless transmission may blend images with noise. In this section, we leverage matrix completion to recover incomplete images corrupted by Gaussian noise. The first used image is called *Windows* [52] whose dimensions are 349×366. The procedure to mixing noise in incomplete images is to use the built-in command of 'imnoise(image,'gaussian',μ, σ^2)' in MATLAB where μ and σ^2 denote the mean and variance of the noise, respectively. In our experiments, $\mu = 0$ is applied for all cases. Furthermore, peak SNR (PSNR) and structural similarity (SSIM) are measured to evaluate the recovery performance. These two indices can be invoked by the built-in MATLAB commands, viz. 'psnr(recovered, original)' and 'ssim(recovered, original)'. It is worth mentioning that larger values of PSNR and SSIM indicate better restoration performance.

We investigate two types of loss, namely, randomly missing and missing not at random. The former has 50% missing pixels randomly distributed in the image, while the latter is generated by the text of "matrix completion". Fig. 3 shows the original and incomplete *Windows*. The left one is the original image, while the middle and right are the observed images with random and fixed losses, respectively. Besides, the intensity of Gaussian noise is $\sigma^2 = 0.005$.

Fig. 4 depicts the recovered images by different methods, excluding CNTK and CNTK$^+$. The first and second rows show the restored images in the presence of random and fixed losses, respectively. Besides, the two evaluation indices in dB are listed below the restored images. It is seen that the BiBNN attains larger PSNRs and SSIMs than PAM, MC-DMF, MC-IALM, NLMC, and PMC, that is, our neural network outperforms these state-of-the-art methods. Note that the restored images by MC-DMF, MC-IALM, NLMC, and PMC still contain noise, leading to unsatisfactory performance. The reason for retaining noise in the results may be that the repaired images are of high-rank since the small singular values, associating with their left and right singular vectors, are noisy.

To compare with CNTK and CNTK$^+$, we need to reduce the image size because CNTK requires more than 60 GiB memory to create a kernel for processing the image with 349×366. To resize *windows*, we exploit the built-in MATLAB command of 'imresize(image,[numrows numcols])', and then attain the resized image with the dimensions of 160×160. Subsequently, the small-size *windows* is masked by the two types of masks, and then the incomplete images are contaminated by Gaussian noise with $\sigma^2 = 0.005$. The restored images are shown in Fig. 5 where the first and second rows comprise the recovered images in random and fixed masks, respectively. We see that the CNTK and CNTK$^+$ are able to restore the incomplete *Windows* in the random loss case, but their performance is still inferior to the proposed approach. With the text mask, the superiority of our network over CNTK and CNTK$^+$ is more obvious. It can be seen that the reconstructed images by the CNTK and CNTK$^+$ have blurry areas.

Moreover, the impact of the percentage of randomly missing data on PSNR is investigated, and the results are shown in Figs. 6 and 7. Fig. 6 depicts the performance of the BiBNN, MC-DMF, MC-IALM, NLMC, and PMC on the original-size *Windows*. It is seen that the developed neural network attains the highest PSNR in all percentages of missing data. When the percentage is smaller, the superiority of our BiBNN over MC-DMF, MC-IALM, NLMC, and PMC is more distinct. The difference between PAM and our network is not obvious from 10% to 60% but becomes large between 70% and 80%, especially the lowest PSNR of PAM in 80%. Besides, the PSNR of all algorithms, excluding MC-DMF and MC-IALM, decreases with the increase of the missing percentage. For the MC-DMF and MC-IALM, they attain their largest PSNR at 60% and 40%, respectively, which are consistent with the results on synthetic data. On the other hand, comparison of the proposed neural network, CNTK and CNTK$^+$ is plotted in Fig. 7, which is based on the small-size image. We see that the BiBNN outperforms the CNTK and CNTK$^+$. While the latter exhibit a similar curve to MC-DMF and MC-IALM, that is, the performance in the lowest missing percentage is not the best.

Furthermore, we compare our method with the existing algorithms on another four images, namely, *Lenna* [53], *Peppers* [53], *Sea* [18] and *Mountains* [18]. The images are illustrated in Fig. 8 where the first row includes the original images, and the incomplete images are shown in the second and third rows. Besides, Gaussian noise with $\sigma^2 = 0.005$ is added to the partially observed pictures.

The PSNR and SSIM results on these four images are tabulated in Table 2. Except the SSIM of our method on *Lenna* in fixed mask is smaller than that of PAM, the PSNR and SSIM of the developed approach are larger than those of the other methods. It is worth noting that although the BiBNN attains a smaller SSIM than PAM in fixed mask on *Lenna*, our PSNR is larger than that of PAM.

4.4. Recommender system

The second application of matrix completion is recommender system. A recommender system can be modeled as an incomplete matrix whose row and column indices represent the user and item identity numbers while the known entries are acquired ratings. The task of matrix completion is to predict the unknown ratings so as to suggest items to users. In this experiment, two datasets,

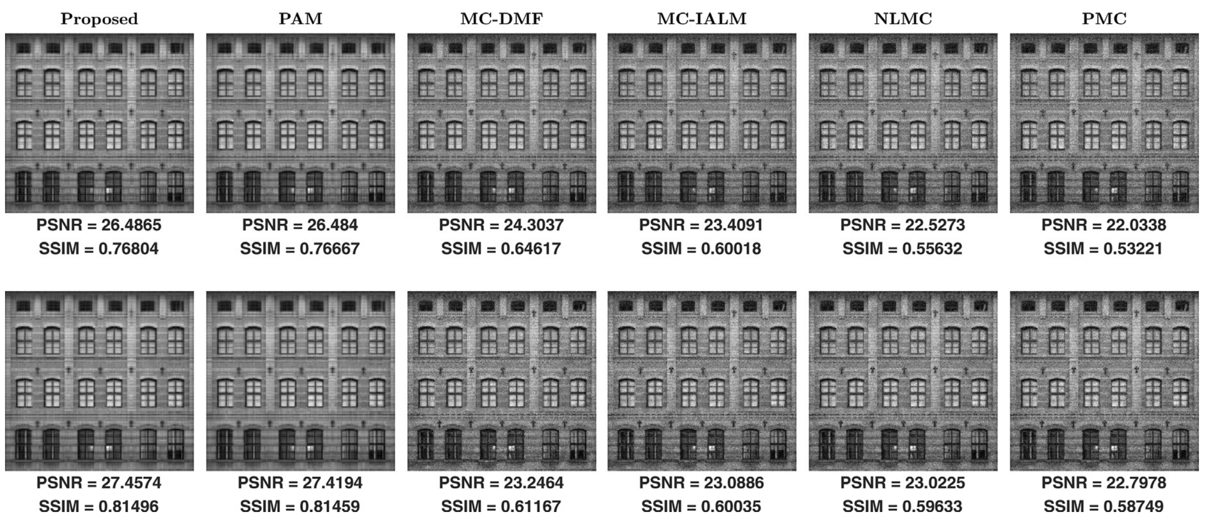


Fig. 4. Performance of different approaches in Gaussian noise with $\sigma^2 = 0.005$. The top row contains results with random loss; The bottom row contains results with fixed loss.

Table 2
Results of different methods on four images.

Image	Mask	Index	Proposed	PAM	MC-DMF	MC-IALM	NLMC	PMC
Lenna	random	PSNR	22.4340	22.3716	21.6861	21.6584	21.7937	21.2323
		SSIM	0.4606	0.4563	0.4052	0.3822	0.3808	0.3559
	fixed	PSNR	21.7603	19.5593	21.5645	21.4124	21.5564	21.2704
		SSIM	0.5095	0.5495	0.4162	0.4055	0.4057	0.4006
Peppers	random	PSNR	23.4671	23.3994	21.9162	22.1254	22.3654	21.8147
		SSIM	0.4947	0.4927	0.3902	0.3781	0.3826	0.3601
	fixed	PSNR	22.4636	21.0854	21.9735	22.0751	22.3287	21.9513
		SSIM	0.5814	0.5789	0.4050	0.3979	0.4000	0.3924
Sea	random	PSNR	26.3713	26.3079	24.3904	23.3322	22.7366	22.2032
		SSIM	0.6075	0.6035	0.4688	0.4064	0.3757	0.3520
	fixed	PSNR	26.7376	26.6685	23.1207	22.9349	22.9591	22.7584
		SSIM	0.6684	0.6672	0.4343	0.4205	0.4185	0.4099
Mountains	random	PSNR	28.7422	28.6777	25.1559	24.1487	23.2669	22.7694
		SSIM	0.6727	0.6710	0.3246	0.2716	0.2291	0.2101
	fixed	PSNR	29.1560	29.0452	23.5022	23.3547	23.2953	23.0148
		SSIM	0.7604	0.7583	0.2722	0.2454	0.2379	0.2295

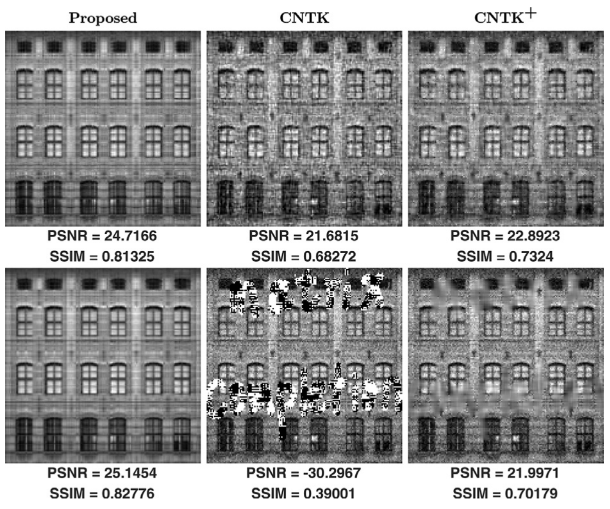


Fig. 5. Performance of proposed method, CNTK and CNTK$^+$ under two mask types in Gaussian noise with $\sigma^2 = 0.005$. The top row contains results with random loss; The bottom row contains results with fixed loss.

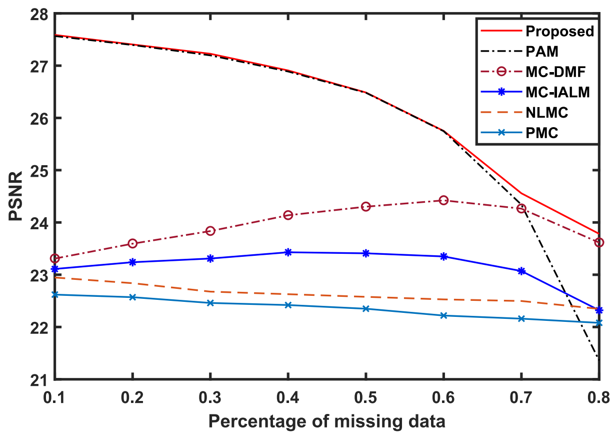


Fig. 6. PSNR versus percentage of randomly missing data in Gaussian noise with $\sigma^2 = 0.005$ by different approaches.

viz. MovieLens 100k[3] and Jester 100k[4] are utilized to evaluate the BiBNN performance. The details of these two datasets are listed in

[3] http://grouplens.org/datasets/movielens/100k/
[4] http://eigentaste.berkeley.edu/dataset/

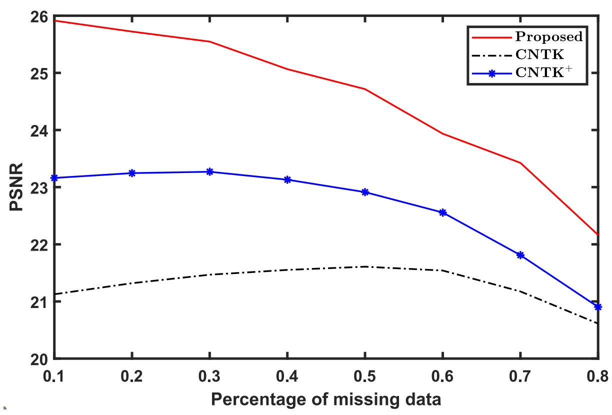


Fig. 7. PSNR versus percentage of randomly missing data in Gaussian noise with $\sigma^2 = 0.005$ by BiBNN, CNTK and CNTK$^+$.

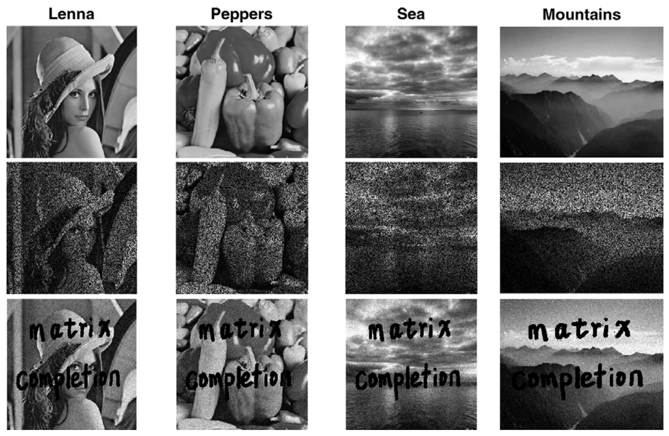


Fig. 8. Illustration of four images, including original and incomplete pictures with randomly missing and fixed masks.

Table 3
Recommendation datasets.

Dataset	# user	# item	# rating	Rating range
MovieLens 100k	943	1682	1×10^5	1 - 5
Jester 100k	159	7699	1×10^5	−10 - 10

Table 3 that includes the numbers of users, items, and ratings as well as the range of rating. The rating of the ith user to the jth item is equal to the (i, j) entry of the incomplete matrix $\boldsymbol{X}_\Omega$. Then, we remove any row or column which has less than 2 observed entries for performing cross-validation, resulting in 943×1541 for MovieLens 100k and 136×4000 for Jester 100k. Since the test dataset is not provided, we leverage the cross-validation strategy, similar to the rank selection scheme, to evaluate prediction performance. Specifically, the whole observed set Ω is divided into Ω_1 and Ω_2 such that $\Omega_1 + \Omega_2 = \Omega$ and $\|\Omega_1\|_1/\|\Omega\|_1 = 0.95$ wherein $\boldsymbol{X}_{\Omega_1}$ and $\boldsymbol{X}_{\Omega_2}$ are used to train and test, respectively. In addition, the estimation performance is evaluated by mean absolute error

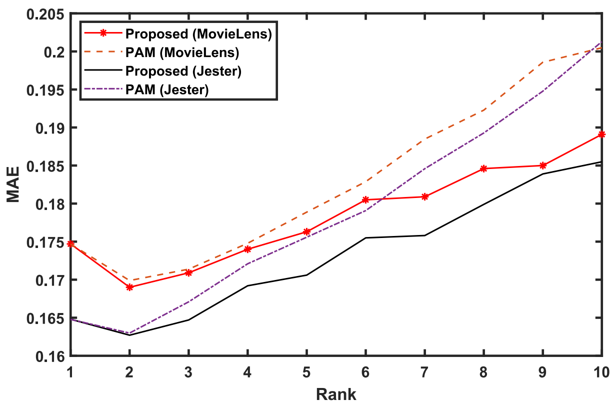


Fig. 9. MAE versus rank by proposed method and PAM on MovieLens and Jester datasets.

(MAE) [54], defined as

$$\text{MAE} = \frac{\|\boldsymbol{M}_{\Omega_2} - \boldsymbol{X}_{\Omega_2}\|_1}{(HC - LC)||\Omega_2||_1} \tag{28}$$

where HC and LC denote the highest and lowest scores in the rating range, that is, $HC - LC = 4$ for MovieLens 100k, and $HC - LC = 20$ for Jester 100k. It is clear that a small value of MAE indicates good prediction performance.

Table 4 shows the results by different methods where the adopted rank of the BiBNN and PAM is 2. Note that CNTK and CNTK$^+$ are excluded since they cannot process these two large-scale matrices. It is seen that the BiBNN attains the smallest values among six methods for both MovieLens 100k and Jester 100k.

Moreover, we investigate the performance of our BiBNN and PAM under different ranks from 1 to 10. The result is plotted in Fig. 9. Except for two similar MAEs at $r = 1$, the BiBNN has smaller values than the PAM in the range of 2 to 10. It is seen that both BiBNN and PAN achieve the smallest MAE at $r = 2$.

5. Conclusion

In this paper, we have constructed a bi-branch neural network for matrix completion. The proposed neural network is designed via unfolding the matrix factorization formulation and learned by end-to-end training using all individual observed entries. The developed BiBNN has been applied to image inpainting and recommender system, achieving better performance than seven state-of-the-art approaches in terms of prediction accuracy.

The current BiBNN requires setting an essential parameter, that is, the rank of the objective matrix. In the future, we plan to offer the neural network the ability to learn the best rank automatically.

Declaration of Competing Interest

We declare that we have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Table 4
MAE of different approaches on MovieLens 100K and Jester 100K datasets.

Dataset	Proposed	PAM	MC-DMF	MC-IALM	NLMC	PMC
MovieLens 100k	0.1690	0.1699	0.1833	0.1835	0.1788	0.1817
Jester 100k	0.1627	0.1631	0.1757	0.2215	0.1918	0.1996

CRediT authorship contribution statement

Xiao Peng Li: Conceptualization, Formal analysis, Investigation, Methodology, Validation, Writing – review & editing. **Maolin Wang:** Conceptualization, Formal analysis, Investigation, Methodology, Validation. **Hing Cheung So:** Project administration, Writing – review & editing, Resources, Methodology.

References

[1] E.J. Candès, B. Recht, Exact matrix completion via convex optimization, Found. Comput. Math. 9 (6) (2009) 717–772.
[2] E.J. Candès, Y. Plan, Matrix completion with noise, Proc. IEEE 98 (6) (2010) 925–936.
[3] H. Xue, S. Zhang, D. Cai, Depth image inpainting: improving low rank matrix completion with low gradient regularization, IEEE Trans. Image Process. 26 (9) (2017) 4311–4320.
[4] X.P. Li, Q. Liu, H.C. So, Rank-one matrix approximation with ℓ_p-norm for image inpainting, IEEE Signal Process. Lett. 27 (2020) 680–684.
[5] A. Ramlatchan, M. Yang, Q. Liu, M. Li, J. Wang, Y. Li, A survey of matrix completion methods for recommendation systems, Big Data Mining Anal., 1 (4) (2018) 308–323.
[6] A. Mongia, A. Majumdar, Matrix completion on multiple graphs: application in collaborative filtering, Signal Process. 165 (2019) 144–148.
[7] R. Du, C. Chen, B. Yang, N. Lu, X. Guan, X. Shen, Effective urban traffic monitoring by vehicular sensor networks, IEEE Trans. Veh. Technol. 64 (1) (2014) 273–286.
[8] Z. Liu, A. Hansson, L. Vandenberghe, Nuclear norm system identification with missing inputs and outputs, Syst. Control Lett. 62 (8) (2013) 605–612.
[9] Y. Luo, T. Liu, D. Tao, C. Xu, Multiview matrix completion for multilabel image classification, IEEE Trans. Image Process. 24 (8) (2015) 2355–2368.
[10] R. Cabral, F. De la Torre, J.P. Costeira, A. Bernardino, Matrix completion for weakly-supervised multi-label image classification, IEEE Trans. Pattern Anal. Mach. Intell. 37 (1) (2014) 121–135.
[11] Z. Lin, M. Chen, Y. Ma, The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices, Technical Report, 2009. Univ. Illinois at Urbana-Champaign, Champaign, IL
[12] P. Jain, R. Meka, I. Dhillon, Guaranteed rank minimization via singular value projection, in: Proceedings of the International Conference on Neural Information Processing Systems-Volume 1, 2010, pp. 937–945. British Columbia, Canada
[13] J. Tanner, K. Wei, Normalized iterative hard thresholding for matrix completion, SIAM J. Sci. Comput. 35 (5) (2013) S104–S125.
[14] X. Jiang, Z. Zhong, X. Liu, H.C. So, Robust matrix completion via alternating projection, IEEE Signal Process. Lett. 24 (5) (2017) 579–583.
[15] J.-F. Cai, E.J. Candès, Z. Shen, A singular value thresholding algorithm for matrix completion, SIAM J. Optim. 20 (4) (2010) 1956–1982.
[16] K.-C. Toh, S. Yun, An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems, Pacific J. Optim. 6 (15) (2010) 615–640.
[17] S. Ma, D. Goldfarb, L. Chen, Fixed point and Bregman iterative methods for matrix rank minimization, Math. Program. 128 (1) (2011) 321–353.
[18] Y. Hu, D. Zhang, J. Ye, X. Li, X. He, Fast and accurate matrix completion via truncated nuclear norm regularization, IEEE Trans. Pattern Anal. Mach. Intell. 35 (9) (2012) 2117–2130.
[19] S. Gu, Q. Xie, D. Meng, W. Zuo, X. Feng, L. Zhang, Weighted nuclear norm minimization and its applications to low level vision, Int. J. Comput. Vis. 121 (2) (2017) 183–208.
[20] Z. Wen, W. Yin, Y. Zhang, Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm, Math. Program. Comput. 4 (4) (2012) 333–361.
[21] P. Jain, P. Netrapalli, S. Sanghavi, Low-rank matrix completion using alternating minimization, in: Proceedings of the Annual ACM Symposium on Theory of Computing, 2013, pp. 665–674. Palo Alto, California
[22] P. Jain, I.S. Dhillon, Provable inductive matrix completion, arXiv preprint arXiv:1306.0626 (2013).
[23] X. Xu, L. He, H. Lu, A. Shimada, R.-I. Taniguchi, Non-linear matrix completion for social image tagging, IEEE Access 5 (2016) 6688–6696.
[24] X. Alameda-Pineda, E. Ricci, Y. Yan, N. Sebe, Recognizing emotions from abstract paintings using non-linear matrix completion, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 5240–5248. Las Vegas, Nevada, USA
[25] K. Zhong, Z. Song, P. Jain, I.S. Dhillon, Nonlinear inductive matrix completion based on one-layer neural networks, arXiv preprint arXiv:1805.10477 (2018).
[26] M. Yang, S. Xu, A novel patch-based nonlinear matrix completion algorithm for image analysis through convolutional neural network, Neurocomputing 389 (2020) 56–82.
[27] M. Zhang, Y. Chen, Inductive matrix completion based on graph neural networks, in: Proceedings of the International Conference on Learning Representations, 2020. Addis Ababa, Ethiopia
[28] M. Fazel, Matrix rank minimization with applications, Dept. Elect. Eng., Stanford Univ., California, USA, 2002 Ph.D. thesis.
[29] F. Nie, H. Wang, X. Cai, H. Huang, C. Ding, Robust matrix completion via joint schatten p-norm and ℓ_p-norm minimization, in: Proceeding of the IEEE International Conference on Data Mining, 2012, pp. 566–574. Brussels, Belgium Belgium
[30] F. Nie, H. Wang, H. Huang, C. Ding, Joint schatten p-norm and ℓ_p-norm robust matrix completion for missing value recovery, Knowl. Inf. Syst. 42 (3) (2015) 525–544.
[31] W. Dai, O. Milenkovic, SET: An algorithm for consistent matrix completion, in: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, 2010, pp. 3646–3649. Dallas, TX, USA
[32] Q. Li, Z. Zhu, G. Tang, Alternating minimizations converge to second-order optimal solutions, in: Proceedings of the International Conference on Machine Learning, 2019, pp. 3935–3943. California, USA
[33] Z. Wang, M.-J. Lai, Z. Lu, W. Fan, H. Davulcu, J. Ye, Rank-one matrix pursuit for matrix completion, in: Proceedings of the International Conference on Machine Learning, 2014, pp. 91–99. Beijing, China
[34] Z. Wang, M.-J. Lai, Z. Lu, W. Fan, H. Davulcu, J. Ye, Orthogonal rank-one matrix pursuit for low rank matrix completion, SIAM J. Sci. Comput. 37 (1) (2015) A488–A514.
[35] Y. Hu, C. Zhao, D. Cai, X. He, X. Li, Atom decomposition with adaptive basis selection strategy for matrix completion, ACM Trans. Multimedia Comput., Commun., Appl. 12 (3) (2016) 1–25.
[36] Q. Shi, H. Lu, Y.-M. Cheung, Rank-one matrix completion with automatic rank estimation via ℓ_1-norm regularization, IEEE Trans. Neural Netw. Learn. Syst. 29 (10) (2017) 4744–4757.
[37] S. Zhang, L. Yao, A. Sun, Y. Tay, Deep learning based recommender system: a survey and new perspectives, ACM Comput. Surv. 52 (1) (2019) 1–38.
[38] S. Si, K.-Y. Chiang, C.-J. Hsieh, N. Rao, I.S. Dhillon, Goal-directed inductive matrix completion, in: Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1165–1174. New York, USA
[39] J. Fan, T.W. Chow, Non-linear matrix completion, Pattern Recognit. 77 (2018) 378–394.
[40] J. Fan, Y. Zhang, M. Udell, Polynomial matrix completion for missing data imputation and transductive learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, 2020, pp. 3842–3849. New York, USA
[41] Z. Li, H. Wang, AIR-Net: Adaptive and implicit regularization neural network for matrix completion, arXiv preprint arXiv:2110.07557 (2021).
[42] J. Fan, T. Chow, Deep learning based matrix completion, Neurocomputing 266 (2017) 540–549.
[43] J. Fan, J. Cheng, Matrix completion by deep matrix factorization, Neural Netw. 98 (2018) 34–41.
[44] S. Mercier, I. Uysal, Noisy matrix completion on a novel neural network framework, Chemom. Intell. Lab. Syst. 177 (2018) 1–7.
[45] A. Radhakrishnan, G. Stefanakis, M. Belkin, C. Uhler, Simple, fast, and flexible framework for matrix completion with infinite width neural networks, arXiv preprint arXiv:2108.00131 (2021).
[46] F. Monti, M.M. Bronstein, X. Bresson, Geometric matrix completion with recurrent multi-graph neural networks, arXiv preprint arXiv:1704.06803 (2017).
[47] R. van den Berg, T.N. Kipf, M. Welling, Graph convolutional matrix completion, arXiv preprint arXiv:1706.02263 (2017).
[48] L.T. Nguyen, B. Shim, Low-rank matrix completion using graph neural network, in: Proceedings of the International Conference on Information and Communication Technology Convergence, 2020, pp. 17–21. Jeju, Korea (South)
[49] D.M. Nguyen, E. Tsiligianni, N. Deligiannis, Extendable neural matrix completion, in: Proceedings of the International Conference on Acoustics, Speech and Signal Processing, 2018, pp. 6328–6332. Calgary, AB, Canada
[50] C.-G. Li, R. Vidal, A structured sparse plus structured low-rank framework for subspace clustering and completion, IEEE Trans. Signal Process. 64 (24) (2016) 6557–6570.
[51] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, Pytorch: an imperative style, high-performance deep learning library, Adv. Neural Inf. Process. Syst. 32 (2019) 8026–8037.
[52] J. Liu, P. Musialski, P. Wonka, J. Ye, Tensor completion for estimating missing values in visual data, IEEE Trans. Pattern Anal. Mach. Intell. 35 (1) (2012) 208–220.
[53] Y.-H. Kuo, C.-S. Lee, C.-C. Liu, A new fuzzy edge detection method for image enhancement, in: Proceedings of the International Fuzzy Systems Conference, volume 2, 1997, pp. 1069–1074. Barcelona, Spain
[54] W.-J. Zeng, H.C. So, Outlier-robust matrix completion via ℓ_p-minimization, IEEE Trans. Signal Process. 66 (5) (2018) 1125–1140.
[55] Q. Liu, X. Li, J. Yang, Optimum co-design for image denoising between type-2 fuzzy identifier and matrix completion denoiser, IEEE. Trans. Fuzzy Syst. 30 (1) (2022) 287–292.
[56] Q. Liu, X. Li, H. Cao, Two-dimensional localization: low-rank matrix completion with random sampling in massive MIMO system, IEEE Syst. J. 15 (3) (2021) 3628–3631.