Received July 3, 2021, accepted July 13, 2021, date of publication July 20, 2021, date of current version July 27, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3098614

Demixed Sparse Principal Component Analysis Through Hybrid Structural Regularizers

YAN ZHANG AND HAOQING XU
School of Computer Science and Engineering, Southeast University, Nanjing 211189, China
School of Artificial Intelligence, Southeast University, Nanjing 211189, China

Corresponding author: Yan Zhang (zhangyan_seu@outlook.com)

This work was supported in part by the National Key Research and Development Program of China under Grant 2018AAA0100500, in part by the Fundamental Research Funds for the Central Universities, in part by the Jiangsu Provincial Key Laboratory of Network and Information Security under Grant BM2003201, and in part by the Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under Grant 93K-9.

ABSTRACT Recently, the sparse representation of multivariate data has gained great popularity in real-world applications like neural activity analysis. Many previous analyses for these data utilize sparse principal component analysis (SPCA) to obtain a sparse representation. However, ℓ_0-norm based SPCA suffers from non-differentiability and local optimum problems due to non-convex regularization. Additionally, extracting dependencies between task parameters and feature responses is essential for further analysis while SPCA usually generates components without demixing these dependencies. To address these problems, we propose a novel approach, demixed sparse principal component analysis (dSPCA), that relaxes the non-convex constraints into convex regularizers, e.g., ℓ_1-norm and nuclear norm, and demixes dependencies of feature response on various task parameters by optimizing the loss function with marginalized data. The sparse and demixed components greatly improve the interpretability of the multivariate data. We also develop a parallel proximal algorithm to accelerate the optimization for hybrid regularizers based on our method. We provide theoretical analyses for error bound and convergency. We apply our method on simulated datasets to evaluate its time cost, the ability to explain the demixed information, and the ability to recover sparsity for the reconstructed data. Finally, we successfully separate the neural activity into different task parameters like stimulus or decision, and visualize the demixed components based on the real-world dataset.

INDEX TERMS Dimensionality reduction, sparse principal component analysis, demixed principal component analysis, parallel proximal algorithms.

I. INTRODUCTION

Multivariate data is often used to describe systems with multiple dimensions and with many real-world applications like describing neural firing rate in neuroscience [1]–[7] or weather changes in meteorology [8]. Sparse representation of multivariate data achieved great success because it fully considers the inner sparsity of multivariate data itself and highlights the important features with shorter expressions. Analyzing how different task parameters are represented in multivariate data with sparsity is usually an elementary task. For example, in many neurological experiments, the neural activity is affected by multiple task parameters, i.e., the environment settings. Considering the inner structure of nervous system, only part of the neurons will response the change of a specific task parameter. As a result, the neurons could be described by sparsely linear-combined "components" in a low-dimensional space. The sparse representation provides a more explicit and interpretable way of understanding the relationship between neural activities and various task parameters.

A promising way to analyze the sparse representation of multivariate data is referred to as sparse principal component analysis (SPCA). SPCA aims to reduce the number of used variables, thus concentrating on the remaining important information and improving the interpretability of the model. SPCA finds the sparse representation of the multivariate data by setting a large number of coefficients to zero in the projection vectors of principal components (PCs), representing the most representative features of data. To be explicit, SPCA assumes that the features are sparsely correlated with each other. That is, let the $\mathbf{X} \in \mathbb{R}^{n \times p}$ denote the data matrix with

The associate editor coordinating the review of this manuscript and approving it for publication was Mehul S. Raval.

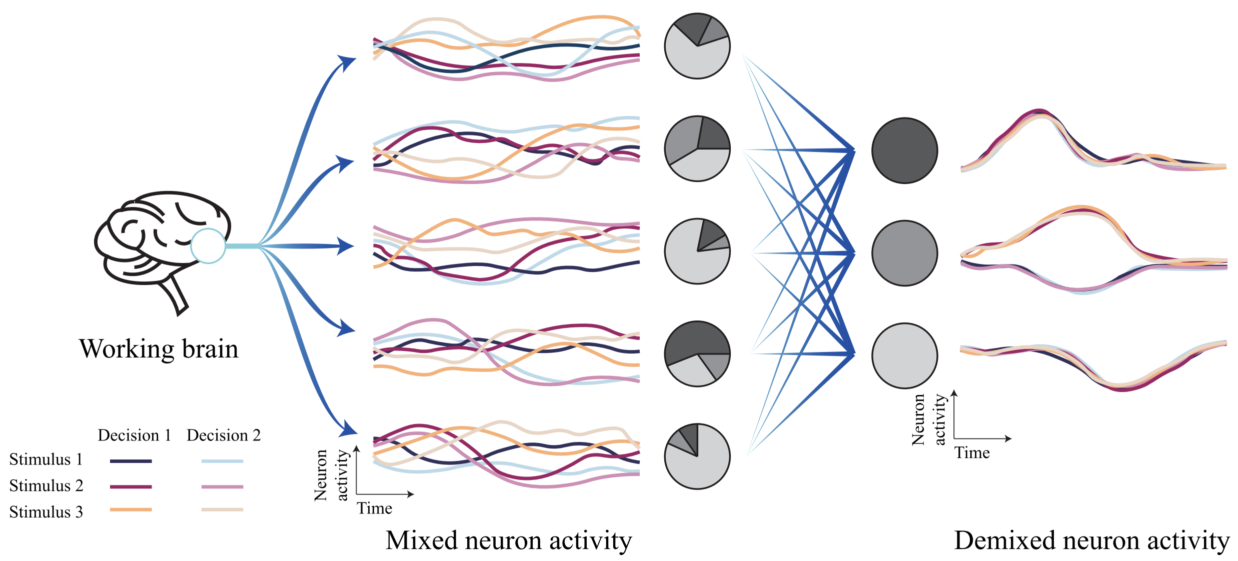

FIGURE 1. **Demixing analysis of neuron data. The brain populations of each neuron can be represented in the form of a time sequence. Thus, we use a set of circles to represent neurons with mixed information. After demixing, each circle represents one component and can interpret the neuron with only one task parameter.**

p features and n samples, the covariance matrix $\Sigma = \mathbf{X}^\top \mathbf{X} \in \mathbb{R}^{p \times p}$ have a sparse representation like $\Sigma = \widetilde{\Sigma} + \epsilon$ where $\widetilde{\Sigma}$ is a sparse matrix and ϵ is a random perturbation. $\widetilde{\Sigma}$ can be recovered from noisy data by adding sparse constraints, i.e., ℓ_0-norm, when estimating its eigenvectors of Σ.

However, SPCA encounters challenges in mainly two aspects. Firstly, SPCA is difficult to solve because it is non-differentiable and non-convex [9]. Since the ℓ_0-norm is non-differentiable, it is hard to optimize. Because the ℓ_0-norm is non-convex, the optimization procedure may obtain local optimum rather than the global optimum. The PCs they extract can be the linear combinations of multiple optimal components, therefore, leading to poor demixing performance. Secondly, the multivariate data in real-world applications may have complex dependencies on various task parameters, such as stimulus and decision in the neural data. SPCA methods can only obtain mixed information that contains little interactive information. In some research and applications, there is great interest in analyzing neuron activities with respect to a single set of task parameters. For example, in a prefrontal cortex (PFC) experiment of monkeys [1], we just want to figure out the relationship between the firing rates and external factors such as 6 different stimuli or 2 different decisions, respectively. In past clinical researches [10], solid tissue samples could be influenced by glandular epithelium and its nearby stroma. By demixing the dependencies between the feature response and the two task parameters, the heterogeneity in tumor samples can be removed, thus helping us better analyze the factors contributing to the cancer.

To improve the performance of the original SPCA based on the ℓ_0 norm when facing non-convex optimization, several methods are proposed [11]–[16]. The recent developed log-determinant rank approximation [11] applied a log-determinant function to get an non-convex rank approximation. This method contributes less to the large singular values and keeps the small singular values close to zero. Even though it is still a non-convex method, it is differentiable. Another common used way to ameliorate the non-convex optimization is to relax the non-convex constraints. In [14], LASSO based SPCA was first proposed to activate only a few coefficients in the multivariate data by using the ℓ_1-norm regularization. In [16], elastic net based SPCA was proposed to consider the grouping effect of the ℓ_1-norm and ℓ_2-norm, which is useful when the number of features p is much larger than the number of samples n. Then, structured SPCA (SSPCA) [15] was developed to consider both sparse and some previously known related structural constraints.

However, the mentioned non-convex or convex methods fail to finish the demixing task, especially for non-convex methods that always result in the local optimum. In Section VI, we compare the performance of both non-convex constraints and convex regularizers in detail. From the results, we can infer that non-convex constraints are not suitable to be combined with demixing idea. To solve the demixing problem, recently, some methods have been introduced. For example, [17], [18] used demixed principal component analysis (dPCA) to separate neural activity into stimulus-influenced, decision-influenced or other factor-influenced components. Therefore, the demixed PCs have clear meanings and greatly improve the interpretability of the model, just as Figure 1 shows. However, for now, these demixing methods rely on naïve PCA that ignores the sparsity commonly existing in multivariate data.

In this paper, we propose a demixing and sparse multivariate data analysis method called demixed sparse principal component analysis (dSPCA) based on a hybrid-regularized optimization problem. dSPCA combines the ℓ_1-norm with the nuclear norm regularizers to get sparse and low-rank estimates. Also, our model leverage the idea of demixing to decompose the multivariate data into different components. Each component explains variance mostly from only one subset of task parameters, so it shows how the particular task parameter affects the final response. Moreover, we apply four different proximal operators to deal with the optimization of two regularizations in parallel, which makes our method more effective. Our method can be seen as complementary to dPCA and SPCA in component analysis. To be specific, this paper makes the following contributions:

- **Novel demixed sparse principal component analysis method.** We propose a novel method called dSPCA to simultaneously extract the demixed sparse and low-rank PCs from data. Each demixed PC explains only one subset of the task parameters and recovers the sparse effects of the task parameters. Therefore, the sparse and low-rank representation with demixed dependencies on various task parameters explains the multivariate data in a more explicit way and we can better analyze the relationship between feature response and different task parameters.
- **Fast and scalable solution with convex and tractable regularizers.** We relax the commonly used non-convex constraints into convex regularizers. Different form the non-convex constrains resulting in messy local optimum problems, ℓ_1-norm and nuclear norm regularizers can offer an accurate and tractable approximation for different task parameters. Moreover, we adjust the formulation of the optimization problem slightly to get faster convergence like Elementary Estimators (EE) have done. Then the optimization problem can be accelerated through parallel proximal related algorithm.
- **Theoretical analysis and convergence guarantee.** We provide theoretical analyses of dSPCA with respect to time complexity and the upper error bound for its estimates. Additionally, we guarantee the convergence of the proposed algorithm. These analyses ensure the effectiveness of dSPCA.

II. BACKGROUND

A. PRINCIPAL COMPONENT ANALYSIS

We first consider naïve dimension reduction model principal component analysis (PCA). PCA is a well-known unsupervised method to analyze the representation of multivariate data. Without considering labels or parameters, it mainly uses different linear combinations of the original data to maximize the variance, thereby preserving the variability of data as much as possible.

These new variable linear combinations are called principal components (PCs). PCA often extracts PCs to maximize the explained variance. Let $\mathbf{X}$ represent the original data where $\mathbf{X} \in \mathbb{R}^{n\times p}$ (suppose $\mathbf{X}$ is already centered), n stands for the number of samples and p represents the number of features. The optimal problem can be described as follows:

$$\widehat{\mathbf{D}} = \arg\min_{\mathbf{D}} \quad \|\mathbf{X} - \mathbf{XDF}\|_F^2. \tag{1}$$

Here $\mathbf{D} \in \mathbb{R}^{p\times q}$ represents the decoding matrix, which reduces the dimension of the multivariate data. $\mathbf{F} = \mathbf{D}^\top \in \mathbb{R}^{q\times p}$ is the encoding matrix, which restores the composed data. q is the dimension of the principal components. The PCs can be given by the matrix multiplication $\mathbf{XD}$.

Although PCA can minimize the error between the projected data and the original data, it still faces some problems. One of the most common problems is that PCA just uses a linear combination of all variables, most PCs are not zero, so they contain many unimportant features and transform the data into meaningless axes. Therefore, PCA is often difficult to interpret.

B. SPARSE PRINCIPAL COMPONENT ANALYSIS

To reduce the number of variables used, one of the most common methods is to add sparse and low-rank constraints to the problem, to prevent meaningless PCs. Sparse principal component analysis (SPCA) was proposed to find the sparse representation of multivariate data. To directly obtain the sparsity factor, we can use ℓ_0-norm and rank function to control the number of non-zero elements and low-rankness. The optimization problem can be written as follows:

$$\begin{aligned}(\widehat{\mathbf{D}}, \widehat{\mathbf{F}}) = \arg\min_{\mathbf{D},\mathbf{F}} \quad & \|\mathbf{X} - \mathbf{XDF}\|_F^2\\ \text{s.t.} \quad & \operatorname{rank}(\mathbf{DF}) \le q\\ & \mathbf{FF}^\top = \mathbf{I}_{(q\times q)}\\ & \|\mathbf{DF}\|_0 \le k.\end{aligned} \tag{2}$$

TABLE 1. The notations we mainly use in this paper.

Notation	Definition
$\mathbf{X} \in \mathbb{R}^{n\times p}$	Original data with n samples and p features (suppose $\mathbf{X}$ has been centered already)
$\mathbf{D} \in \mathbb{R}^{p\times q}$	Decoder with p features and q dimension of principal components
$\mathbf{F} \in \mathbb{R}^{q\times p}$	Encoder with q dimension of principal components and p features
$\mathbf{W} \in \mathbb{R}^{p\times p}$	Dot product of $\mathbf{D}$ and $\mathbf{F}$
$\mathbf{I} \in \mathbb{R}^{q\times q}$	Identity matrix
Θ	Initial estimator
$\mathbf{U}, \mathbf{V}, \Sigma$	The columns of $\mathbf{U}$ and $\mathbf{V}$ are the left and right singular vectors, Σ has singular values and is diagonal
λ, γ	penalty parameters
Φ	Set of all task parameters
ϕ	Subset of set Φ
$\|\cdot\|_F$	The Frobenius norm for matrices or ℓ_2-norm for vectors
$\|\cdot\|_1$	The ℓ_1-norm
$\|\cdot\|_\infty$	The ℓ_∞-norm
$\|\cdot\|_*$	The nuclear norm
$\|\cdot\|_2$	The spectral norm
$\langle\cdot\rangle_\phi$	Average over the set ϕ
$\Pi_{\mathcal{M}}(\mathbf{x})$	$\min_{\mathbf{x}'\in\mathcal{M}}\|\mathbf{x}' - \mathbf{x}\|_F$, the projection function that refers to the Frobenius norm (ℓ_2-norm for vector) closest projection $\mathbf{x}' \in \mathcal{M}$ of $\mathbf{x}$

Here $\mathbf{D} \in \mathbb{R}^{p\times q}$ is the decoder matrix and $\mathbf{F} \in \mathbb{R}^{q\times p}$ is the encoder matrix. p represents the number of features, q is the dimension of the principal components, the maximum rank is $\mathbf{DF}$, and k is the number of non-zero elements. The PCs can be obtained by matrix multiplication $\mathbf{XD}$.

However, since determining the ℓ_0-norm and rank is a non-convex problem, it is highly possible to obtain local optimum. And according to [19], the problem in Eq. (2) can be viewed as a feature selection problem for ordinary least square regression. So, it is NP-hard and can result in a large computational cost.

Since non-convex problems are hard to deal with, the constraints are often relaxed as convex regularizers like ℓ_1-norm [14]. The problem of LASSO based SPCA has the following form:

$$(\widehat{\mathbf{D}}, \widehat{\mathbf{F}}) = \arg\min_{\mathbf{D},\mathbf{F}} \quad \|\mathbf{X} - \mathbf{XDF}\|_F^2 + \lambda\|\mathbf{D}\|_1 \quad \text{s.t.} \quad \mathbf{FF}^\top = \mathbf{I}_{(q\times q)}, \tag{3}$$

Here $\mathbf{D} \in \mathbb{R}^{p\times q}$ is the decoder and $\mathbf{F} \in \mathbb{R}^{q\times p}$ is the encoder of the data $\mathbf{X}$. p is the number of features and q is the dimension of the decomposed data. λ represents the penalty parameter for ℓ_1-norm regularization. The problem can be solved through standard convex optimization to get the global optimum.

Generally, in SPCA, multivariate data is projected to fewer main axes. Due to the reduction in the number of non-zero entries loaded, SPCA can be more clearly represented by using fewer PCs. Thus, it is more interpretable than original PCA.

However, in real world applications, since PCs have mixed information, the hidden dependencies between feature response and different task parameters are hard to detect. Therefore, we always hope that each PC can only have an exact meaning, which will help us better analyze the multivariate data. In the monkey PFC experiment, we want to figure out how stimuli or decisions affect the neuron populations respectively, but SPCA can do nothing when facing this problem. SPCA is still unable to solve the demixing problem and fails to interpret data dependencies over each task parameter, which is a limitation of the method.

C. DEMIXING PROBLEM AND DEMIXED PCA

In a gesture to demix the task parameter dependencies, demixed principal component analysis (dPCA) was proposed in [18]. Apart from compressing the data into a new space, [18] tells us the dependencies of the multivariate data on different task parameters and improves the interpretability of the method. In the monkey PFC experiment, the method successfully demixes the neural activity into stimulus or decision information and uses the corresponding PCs to represent them.

Decoder $\mathbf{D}$ is directly related to the generation of PCs. The decoder $\mathbf{D}$ can be obtained mainly through two steps: marginalization and loss function optimization.

Marginalization separates all task parameters by decomposing the data into different sets of averages. Suppose all the task parameters of the data are represented by the set Φ, for example $\Phi = \{a, b, c\}$. Then we can decompose $\mathbf{x}$ by

$$\begin{aligned}\mathbf{x} &= \mathbf{x}_a + \mathbf{x}_b + \mathbf{x}_c + \mathbf{x}_{ab} + \mathbf{x}_{ac} + \mathbf{x}_{bc} + \mathbf{x}_{abc} + \mathbf{x}_{\text{noise}} \\ &= \sum_\phi \mathbf{x}_\phi + \mathbf{x}_{\text{noise}}.\end{aligned} \tag{4}$$

We use $\mathbf{x}_\phi$ to represent the marginalized averages over the task parameter subset $\phi \subset \Phi$. For example, $\mathbf{x}_a$ is only dependent on task parameter a. The multiple character subscripted marginalized data, like $\mathbf{x}_{ab}$, is the simplification for a subset of all the characters, like $\mathbf{x}_{\{a,b\}}$. Then $\mathbf{X}_\phi$ can be obtained by stacking the sample vectors $\mathbf{x}_\phi$,

$$\mathbf{X} = \sum_\phi \mathbf{X}_\phi + \mathbf{X}_{\text{noise}}. \tag{5}$$

Through marginalization, in Eq. (5), data $\mathbf{X}$ can be decomposed into different sets of averages $\mathbf{X}_\phi$. $\mathbf{X}_\phi$ is the marginalized part for each subset ϕ, we can get the corresponding $\mathbf{D}_\phi$ from $\mathbf{X}_\phi$, the loss function optimization to get $\mathbf{D}_\phi$ is similar to what has been done in PCA.

In order to avoid overfitting in dPCA, a regularization function is added to the loss function. The quadratic penalty is preferred because an ℓ_2-norm has a unique solution [17]. Therefore, the problem for each subset ϕ can be re-written into the following form:

$$(\widehat{\mathbf{D}}_\phi, \widehat{\mathbf{F}}_\phi) = \arg\min_{\mathbf{D}_\phi \mathbf{F}_\phi} \quad \sum_\phi \|\mathbf{X}_\phi - \mathbf{XD}_\phi\mathbf{F}_\phi\|_F^2 + \lambda\|\mathbf{D}_\phi\mathbf{F}_\phi\|_F^2 \quad \text{s.t.} \quad \mathbf{F}_\phi\mathbf{F}_\phi^\top = \mathbf{I}_{(q\times q)}, \tag{6}$$

where $\mathbf{D}_\phi \in \mathbb{R}^{p\times q}$, $\mathbf{F}_\phi \in \mathbb{R}^{q\times p}$ denote the decoder and encoder, respectively, while λ stands for the ℓ_2-norm penalty parameter.

However, ℓ_2-norm regularization still fails to consider the sparsity of the data itself. Additionally, if we want to add ℓ_1-norm regularization or simultaneously add more than two structural constraints to dPCA, like SPCA has done [20], a huge computational cost for the optimization is added because the ℓ_1-norm is non-differentiable for all demixed subsets $\phi \subset \Phi$. Therefore, it is always difficult to achieve a simple combination of SPCA and dPCA.

D. ELEMENTARY ESTIMATORS FOR MULTIVARIATE DATA

It is critical to find out a method which can solve the ℓ_1-norm related regularization quickly. As for the linear regression problem, elementary estimators (EE) [21] provide a closed-form solution with consistent estimators and fast convergence for regularized convex problems.

Consider the linear regression model,

$$\widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \quad \sum_{i=1}^{n}(\mathbf{y}_i - \mathbf{x}_i^\top\mathbf{w})^2, \quad i = 1, \ldots, n, \tag{7}$$

the Elementary Estimator [21] with general structural regularizers was proposed for high-dimensional linear

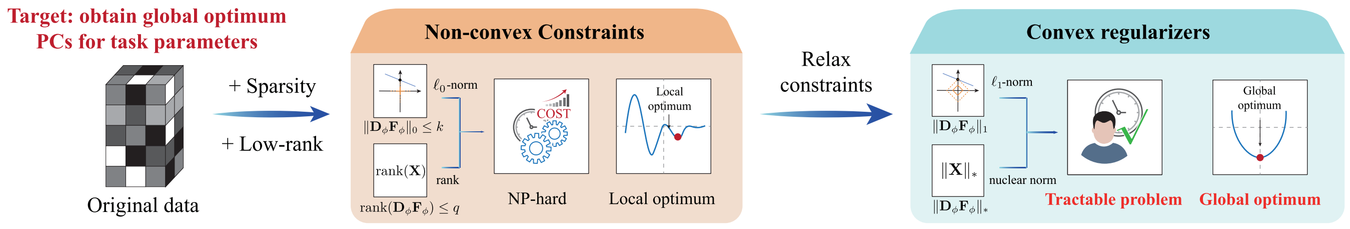

FIGURE 2. Relaxing non-convex constraints. Previous SPCA methods considered the ℓ_0 norm and the rank function to control the number of zero elements and the low-rankness. However, as non-convex constraints, the problem is NP-hard and may sometimes obtain a local optimum. This problem can lead to wrong approximation when combined with dPCA. By relaxing the non-convex constraints, in dSPCA we used ℓ_1-norm and nuclear norm as the convex regularizers of the original problem. Thus, the problem is converted to the tractable problem and we can get global optimum.

regression models. For Eq. (7), its corresponding Elementary Estimators can be written as follows:

$$\begin{aligned} \widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \quad & \mathcal{R}(\mathbf{w}) \\ \text{s.t.} \quad & \mathcal{R}^*(\mathbf{w} - \Theta) \le \lambda, \end{aligned} \tag{8}$$

where $\Theta = (\mathbf{X}^\top \mathbf{X} + \mu \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y}$ is determined from the well-known least square solution. $\mathcal{R}(\cdot)$ represents the regularization function and $\mathcal{R}^*(\mathbf{u}) = \sup_{\mathbf{w}:\mathcal{R}(\mathbf{w}) \neq 0} \frac{\mathbf{u}^\top \mathbf{w}}{\mathcal{R}(\mathbf{w})}$ is its dual form.

The estimator is available to minimize the structural complexity and is suitable for many typical regularizations, since it uses an initial estimator closer to the global optimum than a random start. For example, consider the ℓ_1-norm as an example, the estimator can be given by

$$\begin{aligned} \widehat{\mathbf{w}} = \arg\min_{\mathbf{w}} \quad & \|\mathbf{w}\|_1 \\ \text{s.t.} \quad & \|\mathbf{w} - \Theta\|_\infty \le \lambda, \end{aligned} \tag{9}$$

and the unique solution in closed form can be determined by: $\hat{\mathbf{w}} = S_\lambda(\Theta)$, where $[S_\lambda(\mathbf{w})]_i = \text{sign}(\mathbf{w}_i) \max\{|\mathbf{w}_i| - \lambda, 0\}$ represents the soft-thresholding function.

EE can provide a closed-form estimator for most single-norm constrained problems and can offer a faster convergence when compared with normal regression problem.

III. PROPOSED: DEMIXED SPARSE PRINCIPAL COMPONENT ANALYSIS THROUGH HYBRID STRUCTURAL REGULARIZERS

To get sparse representation of the multivariate data, previous SPCA methods utilize sparse constraints, but they fail to demix the dependencies between task parameters. In this paper, we propose a method called demixed principal component analysis (dSPCA), not only uses the demixing idea similar to dPCA, but also relax the non-convex constraints into convex regularizers to reach the ideal sparsity of loadings. The advantages of convex regularizers are shown in Figure 2. By applying parallel proximal related algorithm, we use fewer computational resources and accelerate the optimization. Figure 3 shows the whole process of generating the demixed PCs using convex regularizers on dSPCA.

A. RELAXING NON-CONVEX CONSTRAINTS FOR DEMIXED DATA

SPCA methods suffer from non-convex and non-differentiable constraints. In our proposed method, we relax the ℓ_0 norm and rank constraints into convex regularizers instead. We use an ℓ_1-norm as a substitute for the ℓ_0-norm, and the nuclear norm to substitute rank limit. Therefore, the convex solution offers a global optimum. To obtain the demixed information over different task parameters, we generate the demixed decoder $\mathbf{D}_\phi$ and encoder $\mathbf{F}_\phi$ after marginalization, like dPCA has done. Moreover, inspired from EE, we can revise the optimization problem for faster convergence, just as Eq. (11) shows. The process of relaxing non-convex problem is shown in Figure 2.

To separate the dependencies on different task parameters, similar to dPCA, the sparse and demixed PCs are obtained through matrix multiplication between $\mathbf{X}$ and $\mathbf{D}_\phi$. The key point of our model lies in the generation of the decoder $\mathbf{D}_\phi$. The generation of $\mathbf{D}_\phi$ includes two parts: marginalization and loss function optimization.

The marginalization of data $\mathbf{X}$ can be calculated using a recursion style like

$$\mathbf{X}_\phi = \langle \mathbf{X} \rangle_{\Phi \setminus \phi} - \sum_{\tau \subset \phi} \mathbf{X}_\tau, \tag{10}$$

where $\langle \mathbf{X} \rangle_{\Phi \setminus \phi}$ stands for averaging $\mathbf{X}$ among all parameter combinations in the task parameter subset $\Phi \setminus \phi$. By calculating the averages of a set of task parameters, marginalization allows us to separate the variance of $\mathbf{X}$ [17].

With the marginalized data and an initial estimator $\Theta = (\mathbf{X}^\top \mathbf{X} + \mu \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{X}_\phi$, from a regularized linear regression solution and EE, our proposed decoder $\mathbf{D}_\phi$ and encoder $\mathbf{F}_\phi$ are determined via the following optimizing problem:

$$\begin{aligned} (\widehat{\mathbf{D}}_\phi, \widehat{\mathbf{F}}_\phi) = \arg\min_{\mathbf{D}_\phi \mathbf{F}_\phi} & \|\mathbf{D}_\phi \mathbf{F}_\phi\|_1 + \|\mathbf{D}_\phi \mathbf{F}_\phi\|_* \\ \text{s.t.} \quad & \|\mathbf{D}_\phi \mathbf{F}_\phi - \Theta\|_\infty \le \lambda \\ & \|\mathbf{D}_\phi \mathbf{F}_\phi - \Theta\|_2 \le \gamma \\ & \mathbf{F}_\phi \mathbf{F}_\phi^\top = \mathbf{I}_{(q \times q)}. \end{aligned} \tag{11}$$

Here, $\mathbf{D}_\phi \in \mathbb{R}^{p \times q}$ and $\mathbf{F}_\phi \in \mathbb{R}^{q \times p}$ are the decoding matrix and encoding matrix, respectively. The ℓ_1-norm regularization controls the sparsity and the nuclear norm regularization controls the low-rankness. The ℓ_∞-norm and nuclear norm are their corresponding dual functions. Notice that we relax the rank constraints in SPCA to avoid problems so that the q here is just used to extract the top-q components to compare with other methods. λ and γ are tuning parameters. λ represents the penalty parameter for ℓ_1-norm. The larger the λ is,

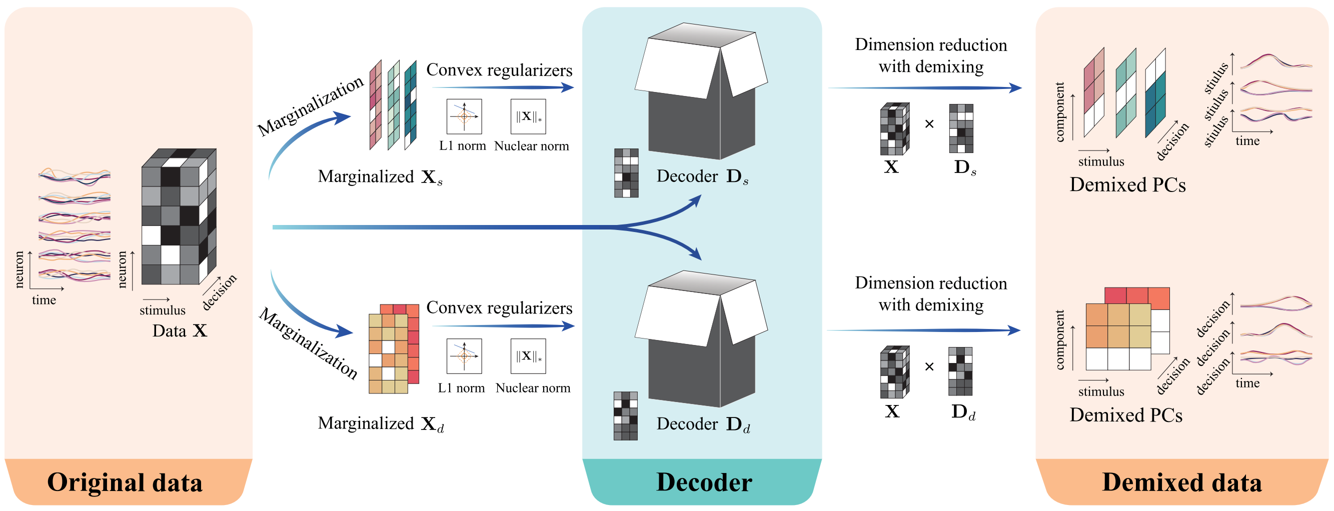

FIGURE 3. Flow diagram of proposed method. In the proposed method, a single neuron can be demixed using two main steps. For the decoder generalization, we apply marginalization to each input data to get $\mathbf{X}_\phi$. Considering the sparsity and low-rankness of data itself, regularizers are added to the problem. As is illustrated before, the ℓ_1-norm and nuclear norm are the relaxed regularizers that can help find the global optimized decoder $\mathbf{D}_\phi$. After generating the decoder from the initial data X, X can also combine with the decoder $\mathbf{D}_\phi$ to conduct dimension reduction with demixing. We apply matrix multiplication to X and $\mathbf{D}_\phi$ to get the demixed PCs.

the sparser the demixed data will be. γ represents the penalty parameter for nuclear norm. The larger the γ is, the stronger the low-rank data will be.

Different from the ℓ_2-norm dPCA, hybrid structural regularizers can deal with both sparse and low-rank structures at the same time, which allows the transformed data to have more explicit representations.

B. FAST OPTIMIZATION THROUGH PARALLEL PROXIMAL ALGORITHM

To use fewer computational resources when handling the ℓ_1-norm and nuclear norm, we use a parallel proximal algorithm related to our method instead. Inspired from the parallel proximal algorithm [22], we can optimize the problem and consider two regularizers simultaneously. Let $\mathbf{W} = \mathbf{D}_\phi \mathbf{F}_\phi$ as the reconstruction matrix for marginalization subset ϕ. We omit the subscript ϕ in $\mathbf{W}$ for notation simplification. Eq. (11) can be equivalently written as follows:

$$\widehat{\mathbf{W}} = \operatorname*{arg\,min}_{\mathbf{W}_1,\mathbf{W}_2,\mathbf{W}_3,\mathbf{W}_4} f_1(\mathbf{W}_1)+f_2(\mathbf{W}_2)+f_3(\mathbf{W}_3)+f_4(\mathbf{W}_4)$$
$$\text{s.t.} \quad \mathbf{W}_1 = \mathbf{W}_2 = \mathbf{W}_3 = \mathbf{W}_4, \tag{12}$$

where $f_1(\cdot) = \|\cdot\|_1, f_2(\cdot) = \|\cdot\|_*, f_3(\cdot) = \mathcal{I}_{\{\|\mathbf{W}-\Theta\|_\infty \le \lambda\}}(\cdot)$, $f_4(\cdot) = \mathcal{I}_{\{\|\mathbf{W}-\Theta\|_2 \le \gamma\}}(\cdot)$. $\mathcal{I}_S(\cdot)$ stands for an indicator function of set S as $\mathcal{I}_S(\mathbf{W}) = 0$ only when $\mathbf{W} \in S$, else $\mathcal{I}_S(\mathbf{W}) = \infty$. In this case, both f_1, f_2, f_3 and f_4 are convex functions, which satisfy the conditions to use parallel proximal algorithm.

The total process can be summarized in Algorithm 1. We generate the model parameters according to the initial parameter Θ. The updates for a_i, as well as $\mathbf{W}_i$, $i = 1, 2, 3, 4$, are four independent calculations, so that we can compute them with four threads in parallel. Thus, the time cost can be reduced. In the algorithm, α represents the learning rate

Algorithm 1: Parallel Proximal Based Algorithm for dSPCA

```
Data: original data $\mathbf{X}$, marginalized data $\mathbf{X}_\phi$ from
      Eq. (10), number of components $q$, the maximum
      number of iterations $T$, learning rate $\alpha \in [0, 2]$,
      ridge penalty parameter $\mu$, tuning parameters
      $\beta = \{\lambda, \lambda, \gamma, \gamma\}$.
1  Initialize the model parameters $\mathbf{W} = \mathbf{W}_1 = \mathbf{W}_2 =$
     $\mathbf{W}_3 = \mathbf{W}_4 = \Theta = (\mathbf{X}^\top\mathbf{X} + \mu\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{X}_\phi$
2  for $t = 1$ to $T$ do
3      for $i = 1, 2, 3, 4$ parallelly do
4          $a_i^t = \mathbf{prox}_{4\beta_i f_i}(\mathbf{W}_i^t)$
5      end
6      $a^t = \frac{1}{4}\sum_{i=1}^4 a_i^t$
7      for $i = 1, 2, 3, 4$ parallelly do
8          $\mathbf{W}_i^{t+1} = \mathbf{W}_i^t + \alpha(2a^t - \mathbf{W} - a_i^t)$
9      end
10     $\mathbf{W} = \mathbf{W} + \alpha(a^t - \mathbf{W})$
11 end
12 Choose top-$p$ singular values $\{\sigma_1, \ldots, \sigma_q\}$ of $\mathbf{X}_\phi\mathbf{W}$ and
     their corresponding right-singlar vectors
     $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_q\}$;
13 $\mathbf{F}_\phi = (\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_q)^\top$
14 $\mathbf{D}_\phi = \mathbf{W}\mathbf{F}_\phi^\top$
Result: Decoder $\mathbf{D}_\phi$, Encoder $\mathbf{F}_\phi$
```

of the parallel proximal algorithm for dSPCA. In Section VI, we generally set $\alpha = 1$.

The proximal operator of each f_i is derived in detail below. For ℓ_1-norm,

$$\mathbf{prox}_{4\lambda\|\mathbf{W}\|_1}(\mathbf{W}) = S_{4\lambda}(\mathbf{W}), \tag{13}$$

where $(S_{4\lambda}(\mathbf{W}))_{ij} = \text{sign}(\mathbf{W}_{ij}) \max(|\mathbf{W}_{ij}| - 4\lambda, 0)$ is the soft-thresholding function. For its corresponding dual function,

$$\mathbf{prox}_{\|\mathbf{W}-\Theta\|_\infty \le \lambda}(\mathbf{W}) = T_\lambda(\mathbf{W}; \Theta), \tag{14}$$

where $(T_\lambda(\mathbf{W}; \Theta))_{ij} = \Theta_{ij} + \text{sign}(\mathbf{W}_{ij} - \Theta_{ij}) \min(|\mathbf{W}_{ij} - \Theta_{ij}|, \lambda)$. For the nuclear-norm function, assuming matrix X has singular value decomposition (SVD) as $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^\top, \Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$ where r represents the rank of $\mathbf{X}$. We use SVD to represent the proximal mapping of the nuclear norm function:

$$\mathbf{prox}_{4\gamma\|\mathbf{W}\|_*}(\mathbf{W}) = \mathbf{U}S_{4\gamma}(\Sigma)\mathbf{V}^\top, \tag{15}$$

where $\mathbf{W} = \mathbf{U}\Sigma\mathbf{V}^\top$. For its corresponding dual form,

$$\mathbf{prox}_{\|\mathbf{W}-\Theta\|_2 \le \gamma}(\mathbf{W}) = \mathbf{U}T_\gamma(\Sigma; \mathbf{0})\mathbf{V}^\top + \Theta. \tag{16}$$

Here the $S_{4\gamma}(\cdot)$ and $T_\gamma(\cdot)$ is defined above.

Once we obtain the optimal solution $\widehat{\mathbf{W}}$ of Eq. (12), we can reconstruct the data $\mathbf{X}$ into $\mathbf{X}\widehat{\mathbf{W}}$. According to Eckart-Young-Mirsky theorem [23], its q-rank approximation is given by its top-q principal components. Assuming that $\mathbf{X}\widehat{\mathbf{W}} = \mathbf{U}\Sigma\mathbf{V}^\top$ is the singular value decomposition of the reconstructed data, the encoder and decoder can be set as $\mathbf{F}_\phi = \mathbf{V}_q^\top, \mathbf{D}_\phi = \widehat{\mathbf{W}}\mathbf{F}_\phi^\top$ where $\mathbf{V}_q$ is the q right-singular vectors with regard to the top-q singular values.

IV. THEORETICAL ANALYSIS

A. TIME COMPLEXITY

Since the marginalized step is the same with dPCA and its complexity is much lower than the following optimization procedure, we only analyze the optimization step. First, we initialize $\mathbf{W}$ once and compute $\mathbf{W}$, which only needs to consider matrix multiplication and matrix inversion. Since the computation can be easily finished by a GPU in a short time, this part can be ignored. Next, the update of $\mathbf{W}$ can be solved and the computation for the proximal operators can be sped up by Algorithm 1. From the algorithm we can easily find that the time cost is mainly composed of the computation for SVD with $O(p^3)$ time complexity in each step. If we assume that Algorithm 1 converges in T steps, then the total time complexity is bounded above by $O(Tp^3)$.

B. ERROR BOUND

In this section, we prove the upper bound of our proposed model. We state two essential assumptions:

(C1: sparse) The true reconstruction matrix $\mathbf{W}^*$ has at most k non-zero elements.

(C2: low-rank) The rank of the true reconstruct matrix $\mathbf{W}^*$ is less than r.

These two assumptions precisely define sparsity and low rank. Then the error bound is provided below.

Theorem 1: Suppose that the true reconstruction matrix $\mathbf{W}^* = \mathbf{D}^*\mathbf{F}^*$ *satisfies assumptions* ***(C1: sparse)*** *and* ***(C2: low-rank)****. Assume that the hyper-parameters* λ, γ *satisfies* $\lambda \ge \|\mathbf{W}^* - \Theta\|_1$ *and* $\gamma \ge \|\mathbf{W}^* - \Theta\|_2$*. Let* $\widehat{\mathbf{W}}$ *be the optimal solution of Eq.* (11). *Thus,*

$$\|\widehat{\mathbf{W}} - \mathbf{W}^*\|_F \le 4\sqrt{\lambda^2 k + \gamma^2 r}. \tag{17}$$

See Appendix A for the detailed proof.

C. CONVERGENCY

We provide the convergence guarantee for the iteration part of Algorithm 1 as Theorem 2.

Theorem 2: Assume the input hyper-parameters of Algorithm 1 satisfies $\alpha \in [0, 2]$, $\lambda, \gamma > 0$, *the iteration part of the algorithm generates a sequence* $\{\mathbf{W}^t\}$ *that converges to a solution of Eq.* (12).

Proof: Our convergence guarantee is based on Theorem 3 (Proved in [24] Section 3.3. See detailed theorem in Appendix B.) that provides the convergence guarantee for the parallel proximal algorithm (PPXA). Consider the problem of Eq. (12), it is obvious that

$$\lim_{\|\mathbf{W}\| \to +\infty} f_1(\mathbf{W}) + f_2(\mathbf{W}) + f_3(\mathbf{W}) + f_4(\mathbf{W}) = +\infty, \tag{18}$$

and

$$\begin{aligned} 0 &\in \text{int}\{(\mathbf{W} - \mathbf{W}_1, \ldots, \mathbf{W} - \mathbf{W}_m) \mid \mathbf{W}, \mathbf{W}_i \in \mathbb{R}^{p\times p}\} \\ &\subset \text{sri}\{(\mathbf{W} - \mathbf{W}_1, \ldots, \mathbf{W} - \mathbf{W}_m) \mid \mathbf{W}, \mathbf{W}_i \in \mathbb{R}^{p\times p}\}, \end{aligned} \tag{19}$$

where $\text{int}\{\cdot\}$ is the interior of a set and $\text{sri}\{\cdot\}$ is the strong relative interior defined in [24]. If we replace the notations in Theorem 3 with those in our algorithm, i.e., $f_i(x) \to f_i(\mathbf{W})$, $\gamma \to \beta$, $y_i^t \to \mathbf{W}_i^t$, $x_t \to \mathbf{W}^t$, and set $w_i = 1/4$, $\lambda_t = 1$, the first two assumptions of Theorem 3 are satisfied. Since we set $\lambda_t = 1$,

$$\sum_{t\in\mathbb{N}} \lambda_t(2 - \lambda_t) = \sum_{t\in\mathbb{N}} 1 = +\infty. \tag{20}$$

Thus, the three assumptions are all satisfied and then due to Theorem 3 the sequence $\{\mathbf{W}^t\}$ generated by Algorithm 1 converges to a solution point for Eq. (12). □

V. RELATED WORK

Table 2 compares our proposed method with the related studies in three different aspects. The previous methods miss at least one aspect while our proposed method considers

TABLE 2. Comparison of related studies for the analysis of multivariate data. The columns indicate the properties of methods: Sparsity refers to whether the representation of multivariate data is sparse or not; Regularizers refers to what regularizers the optimization problem has; Demixing refers to whether the representation is demixed; Parallelism refers to whether the task uses parallel optimization.

Computational Study	Sparsity	Regularizers	Demixing	Parallelism
dSPCA (Proposed)	√	ℓ_1-norm & nuclear norm	√	√
PCA [25]	×	×	×	×
Regression based Method [2]	×	×	×	×
PRF [26]	√	×	×	√
ℓ_0-SPCA [9]	√	×	×	×
ℓ_1-SPCA [20]	√	ℓ_1-norm	×	×
Elastic Net based SPCA [16]	√	ℓ_1-norm & ℓ_2-norm	×	×
dPCA [18]	×	×	√	×
ℓ_2-dPCA [17]	×	ℓ_2-norm	√	×
RPCA [13]	√	ℓ_1-norm & log-det	×	×

all of them. This illustrates that our method combines the advantages of both SPCA and dPCA.

Traditional methods have been applied when analyzing the representation of the multivariate data. A kind of classical methods for the analysis of multivariate data are statistical tests and their related methods [2], [27]–[32]. Brody *et al.* [2] modeled the firing rate decomposing the task via linear regression to find the relationship between firing rate and multiple task parameters like stimulus and decision for each neuron at each time in the PFC of the monkey experiment. Another regression-based method is [27], for each task parameter, one component can be extracted through regression coefficients weighted linear combination. Since the regression based method is generally influenced by labels and task parameters, the transformed data cannot faithfully represent the original data and lose relatively much information. In [27], the method only explained 23% of total variance. Therefore, these two supervised methods cannot help the observer to infer what the original data clearly is from its new representation. The improvement of random forest can also be a supervised method analyzing the multivariate data [26], [33]. In [26], the authors proposed parallel random forest (PRF) algorithm, which uses dimension-reduction approach during the training process and both data-parallel and task-parallel optimization to improve the performance. However, PRF only considers the classification task while ignores the relationship between the feature response and different task parameters in the original data.

When trying to obtain more information from the original data, unsupervised approaches can be more reliable. PCA behaves like a classical unsupervised method, extracting a serious of principal components (PCs) to represent multivariate data [25], [34]–[38]. The PCs aim to maximize the explained variance, and in PFC experiment mentioned above, the first PC explained 69% of the variance, which is about three times when compared with supervised linear models [17]. However, as a naïve unsupervised method, PCA fails to consider the sparsity of multivariate data and is unable to prevent the overfitting problem.

To consider the sparse constraints, [9], [15], [20], [39]–[47] applied sparse principal analysis (SPCA). In [9], the authors introduced ℓ_0-norm penalty to control sparsity. To solve the non-convex related problem caused by ℓ_0-norm, a convex relaxation to the constraints is proposed, which is solved using a greedy algorithm. Erichson *et al.* [40] used LASSO or elastic net to bring in the sparse loadings to the original data. Since many of the coefficients are brought to zero, SPCA obtains a sparse representation of ocean temperature data and successfully discriminated the band of warmer temperatures when compared with PCA, which only found spurious global correlations [40]. In [45], the authors proposed an efficient SPCA method, namely sparse PCA via regularized SVD (SPCA-rSVD), which considers the link between PCA and SVD of the input data to extract the PCs via the regularized low rank matrix approximation. Since the above SPCA methods both need to calculate the full SVDs, as an expansion of SPCA, robust principal component analysis (RPCA) was proposed to calculate partial SVDs instead of all SVDs [48], [49]. To avoid high computational cost and get a more accurate approximation of the rank, [13] proposed the Fast Factorization-based RPCA model (FFP), and applied factorization approach to further reduce the time complexity. The non-convex factorization approach transforms the original data $\mathbf{X}$ into low-rank part $\mathbf{L}$ and sparse part $\mathbf{S}$. The rank of $\mathbf{L}$ is much smaller than p and n, and makes the time complexity of FFP much faster than any other SPCA methods. Similar to FFP, [11], [12] also use non-convex constraints when analyzing the representation of multivariate data. However, in most of these mentioned methods, the mixed information cannot identify the respective information for each component.

To demix the multivariate data, [18] proposed a probabilistic graphical model and applied a fast Expectation-Maximization (EM) algorithm to finish the principal component analysis (dPCA) task. Offering a more flexible loss function, [17] used dPCA to analyze neural populations. In a PFC experiment, dPCA decomposed population activity into five parts like stimulus, decision, condition-independent, stimulus-decision interaction and noise. Each component of different parts tends to maximize the explained variance. dPCA found that 65%-90% of information was captured by condition-independent components and the variance of interaction between stimulus and decision accounts for 5%-22%. Based on the findings of dPCA applied in neuron data, [50] analyzed the neural code and its potential working mechanisms. To prevent overfitting in dPCA, [17] also implemented a quadratic penalty to the loss function. After 100 iterations of cross-validation of a PFC somatosensor task, the classification accuracy can reach around 80%. But dPCA with ℓ_2 regularization still ignored the sparsity of data representation, making this model hard to interpret in real-word messy data.

VI. EXPERIMENT

We conducted simulated and real-world experiments with dSPCA and several baseline methods to compare their behavior. The detailed experimental settings and results are provided below.

A. IMPLEMENTATION

We implement our proposed methods according to Algorithm 1. The code is available at https://github.com/SuikaXhq/dSPCA. The packages we mainly use in the implementation of dSPCA and baselines are numpy, scipy and scikit-learn. We conduct our experiments on a Linux server with 2 Intel Xeon Silver 4216 CPUs.

B. BASELINES

We compare our proposed method with the four baseline methods below:

- PCA: Principal component analysis, the traditional statistical analysis method. It aims to extract principal components from all the dimensions, via a linear combination, to explain the main variance of the data.

- SPCA [51]: An implementation of Sparse PCA. It adds a sparse regularizer to the encoder and solves the problem via an iterative approach.
- dPCA [17]: Demixed principal component analysis described in Section II-C.
- ℓ_2-dPCA: The ℓ_2 regularized dPCA described in [17]. A regularization is added to avoid overfitting.
- RPCA [13]: Robust principal component analysis. This method uses nonconvex regularizers and achieve high estimation speed.

C. METRICS

1) EXPLAINED VARIANCE

We use the cumulative explained variance and individual explained variance to estimate the performance of information description for different components. The cumulative explained variance can be regarded as the capability of preserving the information needed for the dimension reduction task. Here we use R^2 to represent the cumulative explained variance defined as

$$R^2 = \frac{\|\mathbf{X}\|_F^2 - \|\mathbf{X} - \mathbf{XDF}\|_F^2}{\|\mathbf{X}\|_F^2}. \tag{21}$$

The variance explained by each component can also be calculated using Eq. (21) replacing $\mathbf{D}, \mathbf{F}$ with each projection axis $\mathbf{d}, \mathbf{f}^\top$. To describe the marginalized information explained by the demixing components, we further decompose the R^2 to split the fraction of explained variance into additive contributions from the marginalization. That is, for each subset ϕ, the marginal explained variance R_ϕ^2 is defined as

$$R_\phi^2 = \frac{\|\mathbf{X}_\phi\|_F^2 - \|\mathbf{X}_\phi - \mathbf{XDF}\|_F^2}{\|\mathbf{X}\|_F^2}, \tag{22}$$

and $R^2 = \sum_\phi R_\phi^2$ due to the decomposition $\mathbf{X} = \sum_\phi \mathbf{X}_\phi$.

2) TIME

We record the wall-clock time of our dSPCA method and SPCA. Since the rest of the baseline methods do not add the sparsity and low-rankness constraints in their method, the estimating time is not compared in the simulation experiments.

3) SPARSITY RECOVERY

We viewed the discrimination between zero and non-zero elements as a binary classification. We suppose the zero elements in the marginal data $\mathbf{X}_\phi$ are part of the positive class and the non-zero elements are part of the negative class. The true marginal data matrix can be viewed as the ground truth and the reconstructed data matrix is the corresponding marginalization $\mathbf{XD}_\phi\mathbf{F}_\phi$, can be viewed as a predicted condition. By calculating precision and recall, we can apply an F1 score to analyze the ability of sparsity recovery for each method.

D. EXPERIMENT I: PERFORMANCE ON SYNTHETIC DATA

1) DATA PREPARATION

To ensure that the simulated data are task parameter structured, we generate the simulation data $\mathbf{X} \in \mathbb{R}^{nABT \times p}$ with p features and task parameters a, b where A, B are the corresponding task parameter numbers. To simulate the neuron data, the time parameter t is also included and T is the total number of time ticks. The final data is composed by $\mathbf{X} = \mathbf{X}_a + \mathbf{X}_b + \mathbf{X}_t + \mathbf{X}_{\text{noise}}$. The noise for each sample $\mathbf{x}_{\text{noise}}$ was generated from a normal distribution $\mathcal{N}(0, \sigma_\epsilon^2 \mathbf{I}_p)$ where σ_ϵ is set to 1 in the experiments. We generate the data n times as multiple trials for each task parameter pair and time tick.

The marginal data $\mathbf{X}_a$ is generated by the process below. I.

1) To enforce the sparsity of marginalized data $\mathbf{X}_a$, we picked out $1 - s\%$ of dimensions as support dimensions that are non-zero from the overall p features. That is, we randomly choose $p' = (1 - s\%)p$ dimensions from original space $\mathcal{P} = \mathbb{R}^p$ and set the rests to be zeros to create the support set $\mathcal{P}'_a \subset \mathbb{R}^p$. The data vectors are generated in the $\mathbb{R}^{p'}$ and then converted back into the $\mathcal{P}'_a$ with p dimensions.
2) A base vector $\mathbf{x}_a^0 \in \mathbb{R}^A$ is generated with even grid points from $[-3A/2, 3A/2]$.
3) The marginal data matrix for task parameter a in the support dimension space is defined as
$$\begin{aligned}\mathbf{X}'_a &= \begin{pmatrix} x_{a,1}^0 & x_{a,2}^0 & \cdots & x_{a,A-1}^0 & x_{a,A}^0 \\ x_{a,2}^0 & x_{a,3}^0 & \cdots & x_{a,A}^0 & x_{a,1}^0 \\ \vdots & \ddots & & & \vdots \end{pmatrix} \\ &= (\mathbf{x}_a^{\prime 1}, \mathbf{x}_a^{\prime 2}, \ldots, \mathbf{x}_a^{\prime A}) \in \mathbb{R}^{p' \times A},\end{aligned}$$
where each row is the previous row shifted by one element.
4) The marginal data vectors in the original feature space is then defined as $\widetilde{\mathbf{X}}_a = (\mathbf{x}_a^1, \mathbf{x}_a^2, \ldots, \mathbf{x}_a^A)$, where $\mathbf{x}_a^i = \Pi_{\mathcal{P}'_a}(\mathbf{x}_a^{\prime i})$ is the back projection of $\mathbf{x}_a^{\prime i}$ from the support space to original space.

Such element generation procedure ensures that the $\text{rank}(\mathbf{X}_a) = A - 1$ and $\min \|\mathbf{x}_a^i - \mathbf{x}_a^j\|^2 \geq \text{Var}[\mathbf{x}_{\text{noise}}]$. Marginal data for task parameter b is generated just the same as a.

For marginal data vectors $\widetilde{\mathbf{X}}_t = (\mathbf{x}_t^1, \mathbf{x}_t^2, \ldots, \mathbf{x}_t^T)$, the base vector $\mathbf{x}_t^0$ is defined as a set of even grid points from $[0, 10]$ and $\mathbf{X}'_t = \mathbf{c}\mathbf{x}_t^{0\top}$, where $\mathbf{c}$, is sampled from $\mathcal{N}(0, \mathbf{I}_{p'})$. The back projection procedure is the same as task parameters that $\mathbf{x}_t^i = \Pi_{\mathcal{P}'_t}(\mathbf{x}_t^{\prime i})$.

Collecting all the marginal data, the sample vector for parameter a, b and time t is defined as $\mathbf{x}_{abt} = \mathbf{x}_a^a + \mathbf{x}_b^b + \mathbf{x}_t^t + \mathbf{x}_{\text{noise}}$.

2) RESULTS

We generate the stimulated sparse datasets using the precise method described in Section VI. We set the number of trials to be $n = 10$ and the number of features to be $p \in \{100, 200, 300, 400, 500, 600\}$. For three task parameters, we vary $A \in \{6, 12, 18, 24, 30, 36\}$ and fix $B = 4$. The time tick is set to $T = 150$. Here we also use $s\% \in \{0, 20\%, 40\%, 60\%, 80\%\}$ to control the sparsity of the multivariate data. We generate 10 replicates for each setting of data and average performance and standard deviation for each method on the 10 replicates are recorded. The all data used are centralized before applying the methods. The parameter (λ, γ) of dSPCA are choosen by grid search in $(0, \|\Theta\|_\infty] \times (0, \|\Theta\|_2]$, respectively. 5×5 grid is used and we discard zeros in the grid since that setting parameter to zero makes it meaningless. We choose the parameters that minimize the reconstruction error $\|\mathbf{X}_\phi - \mathbf{X}\mathbf{D}_\phi\mathbf{F}_\phi\|_F^2$. The parameters in baseline methods are tuned via the approaches described in their own papers. The number of components, q, is set to 10 for all the methods in order to fairly compare their performance.

a: TIME COST

Figure 4 shows the time cost of SPCA and our method when varying the number of A, B and sparsity degree $s\%$. The error bars stand for the standard deviation of time cost, under logarithmic scale. We only include SPCA for comparison because other methods do not take sparsity into account. Comparing time costs between different problem settings (regularizing the sparsity or not) is unfair. Since dSPCA has a demixing part, its time cost is the average of calculating each single demixed component. Time is recorded in log-second units. From the bar chart, we can find that the time cost for SPCA is 30∼100 times higher than dSPCA. Figure 4(a) shows that SPCA is more easily influenced by sample size while dSPCA is not. Figure 4(b) shows that as the dimension of feature p grows, dSPCA is more easily influenced than SPCA. Perhaps this is because in Section IV, the complexity of our method is $O(Tp^3)$. Even though the time cost can grow as p gets larger, dSPCA is still much faster than SPCA. Figure 4(c) shows that the sparsity of the multivariate data does not influence the time cost of dSPCA, while SPCA is more easily influenced. This can result from the optimization of SPCA for sparse cases [51]. This can be a possible way to improve our method.

b: DEMIXING PERFORMANCE

Figure 5 shows the performance of individual explained variance for four methods. We do not include SPCA related methods since SPCA related methods cannot offer the decoder $\mathbf{D}$. According to Eq.(22), we need to obtain both $\mathbf{D}_\phi$ and $\mathbf{F}_\phi$. However, SPCA related algorithms only offer $\mathbf{F}_\phi$. The error bar of a single bar is the standard deviation of the total explained variance with respect to that component. Notice that such error contains not only the variance of the method but also the variance within data generation since the replicates slightly differ from each other. From the figure we can observe that PCA demix poorly as we expect. While the other three methods have similar demixing performance. Our proposed method shows almost the same demixing performance compared with dPCA methods. The RPCA method generates components that almost do not demix the task parameters.

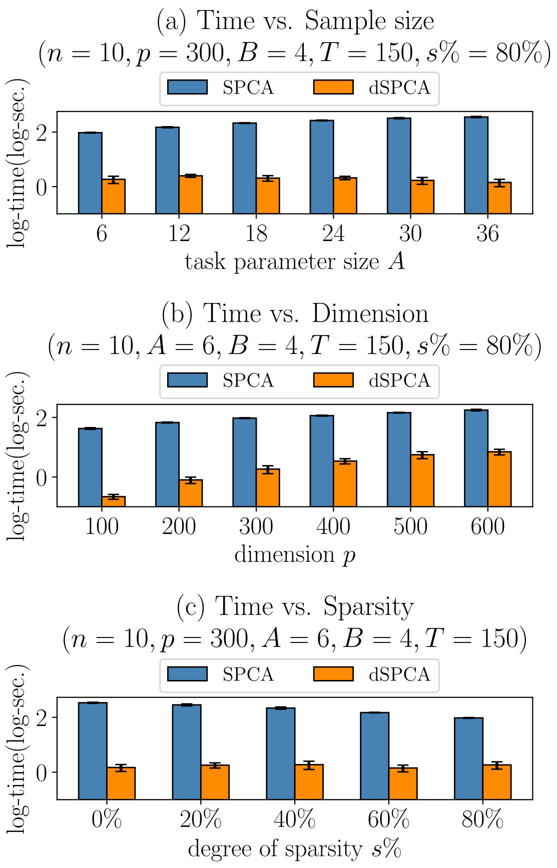

FIGURE 4. Time cost comparison for SPCA and dSPCA. Subfigure (a) shows time vs. sample size (in log-seconds) by varying the number of stimulus in {6, 12, 18, 24, 30, 36}. Subfigure (b) shows time vs. dimension (in log-seconds) by varying the number of features in {100, 200, 300, 400, 500, 600}. Subfigure (c) shows time vs. sparsity (in log-seconds) by the degree of sparsity in {0%, 20%, 40%, 60%, 80%}. The error bars show the standard deviation of time under logarithmic scale.

It means we cannot find out any useful information about the effects of the task parameters since each RPCA component is depend on all the task parameters.

c: SPARSITY RECOVERY

Table 3 shows that dSPCA performs better than dPCA and ℓ_2-dPCA when it comes to the ability to recover sparsity on both task parameter a and b. We do not include PCA or SPCA related methods since they are not applicable in the context of demixing. They cannot generate projection axes specifically for marginalized data and thus they cannot reconstruct them but reconstruct the original data. As a result, PCA and SPCA methods cannot provide useful information with respect to sparse marginalized data. Since the original data $\mathbf{X}$, i.e., the summation of sparse marginalized data, is not necessary to be sparse, it does not make sense to compare the sparsity recovery ability on the original data. Thus, we only compare the sparsity recovery performance on the marginalized data. As the sparsity $s\%$ gets lower, the F1 score for dPCA and ℓ_2-dPCA decreases, while the F1 score for dSPCA remains 1. The high F1 score of dSPCA can be explained in two aspects.

TABLE 3. F1 score of the marginalization for different methods. We recorded the F1 scores and their standard deviations among 10 replicates of data on task parameters *a* and *b* after demixing. The bold numbers represent the best results among all three method. dSPCA outperforms the other two methods in all the cases. We only discuss dPCA related methods since PCA and SPCA related methods are not applicable in the context of demixing.

Data settings						Methods					
						dPCA		ℓ_2-dPCA		dSPCA	
n	p	A	B	T	sparsity	$F1_a$±std	$F1_b$±std	$F1_a$±std	$F1_b$±std	$F1_a$±std	$F1_b$±std
10	300	6	4	150	20%	0.7572±0.0239	0.8413±0.0220	0.7579±0.0236	0.8418±0.0218	**1.0000±0.0000**	**1.0000±0.0000**
10	300	6	4	150	40%	0.7526±0.0158	0.8303±0.0147	0.7531±0.0158	0.8306±0.0146	**1.0000±0.0000**	**1.0000±0.0000**
10	300	6	4	150	60%	0.7561±0.0067	0.8462±0.0167	0.7568±0.0066	0.8467±0.0167	**1.0000±0.0000**	**1.0000±0.0000**
10	300	6	4	150	80%	0.7502±0.0084	0.8442±0.0100	0.7509±0.0083	0.8447±0.0100	**1.0000±0.0000**	**1.0000±0.0000**
10	100	6	4	150	80%	0.7440±0.0232	0.8359±0.0146	0.7454±0.0231	0.8372±0.0146	**1.0000±0.0000**	**1.0000±0.0000**
10	200	6	4	150	80%	0.7471±0.0068	0.8421±0.0112	0.7480±0.0067	0.8427±0.0112	**1.0000±0.0000**	**1.0000±0.0000**
10	400	6	4	150	80%	0.7561±0.0121	0.8398±0.0121	0.7565±0.0121	0.8402±0.0121	**1.0000±0.0000**	**1.0000±0.0000**
10	500	6	4	150	80%	0.7522±0.0072	0.8389±0.0106	0.7528±0.0073	0.8394±0.0106	**1.0000±0.0000**	**1.0000±0.0000**
10	600	6	4	150	80%	0.7540±0.0075	0.8439±0.0067	0.7543±0.0075	0.8442±0.0067	**1.0000±0.0000**	**1.0000±0.0000**
10	300	12	4	150	80%	0.7376±0.0066	0.9389±0.0075	0.7382±0.0067	0.9392±0.0075	**1.0000±0.0000**	**1.0000±0.0000**
10	300	18	4	150	80%	0.7287±0.0060	0.7040±0.0025	0.7303±0.0061	0.9708±0.0024	**1.0000±0.0000**	**1.0000±0.0000**
10	300	24	4	150	80%	0.7556±0.0046	0.9839±0.0016	0.8058±0.0046	0.9927±0.0012	**1.0000±0.0000**	**1.0000±0.0000**
10	300	30	4	150	80%	0.7947±0.0028	0.9924±0.0018	0.8446±0.0033	0.9987±0.0006	**0.9999±0.0000**	**1.0000±0.0000**
10	300	36	4	150	80%	0.8247±0.0021	0.9961±0.0011	0.8958±0.0029	0.9999±0.0002	**1.0000±0.0000**	**1.0000±0.0000**

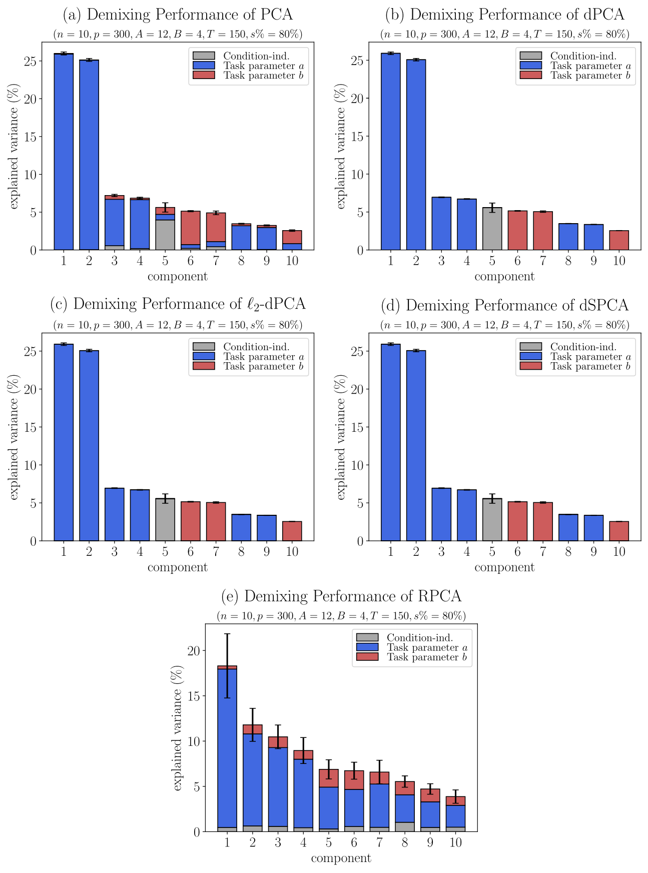

FIGURE 5. Individual explained variance. The data setting is $n = 10, p = 300, A = 12, B = 4, T = 150, s\% = 80\%$. Each bar represents the corresponding proportion of total variance. Blue bar is the stimulus-influenced variance, red bar is the decision-influenced variance, and gray bar is the condition-independent variance. The error bars show the standard deviation of the single component explained variance. SPCA methods are not included since the related algorithms fail to give the decoder $\mathbf{D}$, the individual explained variance cannot be obtained.

On one hand, the data is quite sparse that in most cases there are about 80% of the entries are zeros, so that the F1 score can be 0.8889 for all-zero results. Notice that dPCA methods usually cannot achieve 0.88. On the other hand, dSPCA regularizes the ℓ_1 norm in order to obtain sparse estimates while dPCA methods do not. Also, Figure 6 shows the comparison between F1 score and the cumulative explained variance on different task parameters. We do not include PCA and SPCA methods since they are not applicable in the context of demixing too. This suggests that no matter how the cumulative explained variance changes, the F1 score, that describes the ability to recover sparsity from the reconstructed data, is always very close to 1.0 (with 10^{-4} accuracy). While the F1 score of other methods is about 0.08∼ 0.22 lower than dSPCA. This shows that dSPCA can reconstruct data with sparsity and low-rankness precisely in various settings.

d: REGULARIZER UTILITY STUDY

Figure 7 provides a visualization for the effects of regularizers. Generally, explained variance represents the reconstruction ability of the estimate generated by the method and F1 score represents the interpretation accuracy. The explained variance is high when the two hyper-parameters are all relatively small. Since the initial estimator of dSPCA is the solution for regularized linear regression and with small hyper-parameters, the regularizers have little impact on the estimate, the reconstructed data $\mathbf{X}\mathbf{D}_\phi\mathbf{F}_\phi$ is close to $\mathbf{X}_\phi$ in Euclidean distance. That leads to the high explained variance. As the two parameters grow, the influence of the two regularizers becomes larger and explained variance firstly increases then decreases. It means that a proper hyper-parameter setting could help maintain the reconstruct ability of the estimate. When the parameters are set to zero, the sparsity recovery performance is extremely poor, indicating that the initial estimator of dSPCA cannot reflect sparse property of the data. However, with a small ℓ_1-regularization (a small λ), dSPCA can provide good enough estimate with respect to sparse interpretation.

The contributions of the two regularizers can be summarized as: (1) The ℓ_1-norm regularizer controls the degree of sparsity. With a relatively small λ, the F1 score could reach nearly 1.0 with explained variance maintained, indicating that

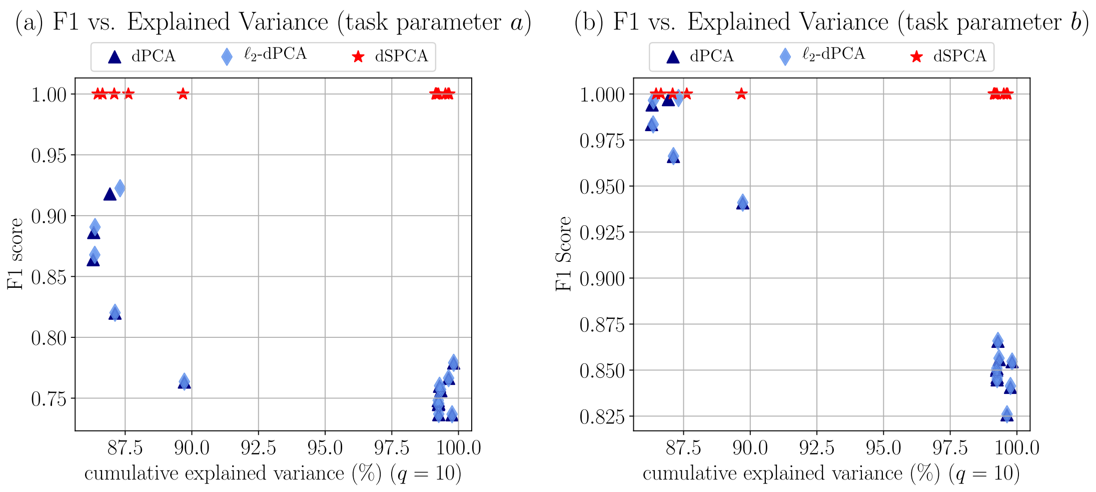

FIGURE 6. Trade-off between cumulative explained variance and F1 score. We compare both F1 score and cumulative explained variance to figure out the trade-off between them for dSPCA and the other two dPCA based methods. Here we set $q = 10$ for all situations. We only compare the performance of dPCA related methods since PCA and SPCA related methods are not applicable in the context of demixing.

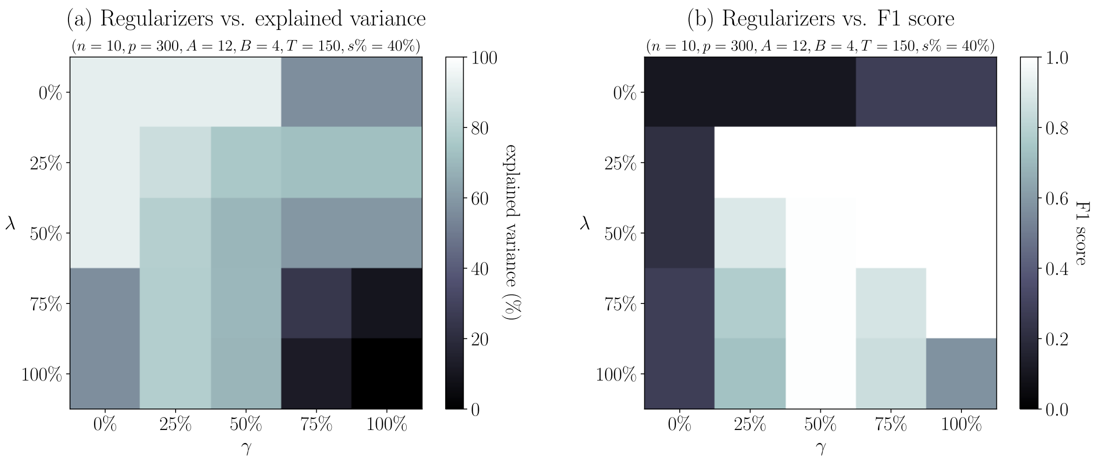

FIGURE 7. Regularizer utility comparison. The data setting is $n = 10$, $p = 300$, $A = 12$, $B = 4$, $T = 150$, $s\% = 40\%$. We report the cumulative explained variance ($q = 10$) and F1 score (task parameter a) from each setting of hyper-parameters λ and γ. 100% means the maximum value $\|\Theta\|_\infty$ and $\|\Theta\|_2$ for λ, γ, respectively. The lighter entry represents the better result. We only show the F1 score for task parameter a here in the figure since task parameter a and b behave similarly.

the ℓ_1-norm regularizer is able to recover the true sparsity of data. With a larger λ, the ℓ_1-norm regularizer involves distinct bias in the estimate, resulting in the poor performance. A relatively small penalty on ℓ_1-norm could efficiently improve the sparse recovery performance while larger penalty is not recommended. (2) The nuclear norm regularizer penalizes the rank of the result. As shown in Figure 7, the effect on explained variance of nuclear norm regularizer is slightly larger than that of ℓ_1-norm regularizer. The poor performance with large γ indicates that larger penalty on rank will cause distortion in the result. Nuclear norm regularizer has either positive or negative impacts with different settings of γ. With a proper setting of γ (50% in the figure), the nuclear norm regularizer can also benefit sparsity recovery. It is better to carefully tune the parameter γ in order to make the nuclear norm regularizer suitable for the data.

Combine the two heat maps, results show that the choice of λ, γ contains a trade-off between reconstruct ability and sparse interpretability. The two regularizers cooperate mutually for a sparse and accurate estimate. The ℓ_1-norm regularizer mainly contributes to the sparsity recovery for interpretation while the nuclear norm regularizer mainly focuses on controlling the low-rankness of the result.

E. EXPERIMENT II: PERFORMANCE ON NEURON ACTIVITY DATA

We used a large number of neural data recordings [1]–[7] to analyze the relationship between multiple task parameters and the changes of animal neural firing rates in monkey PFC with the proposed model.

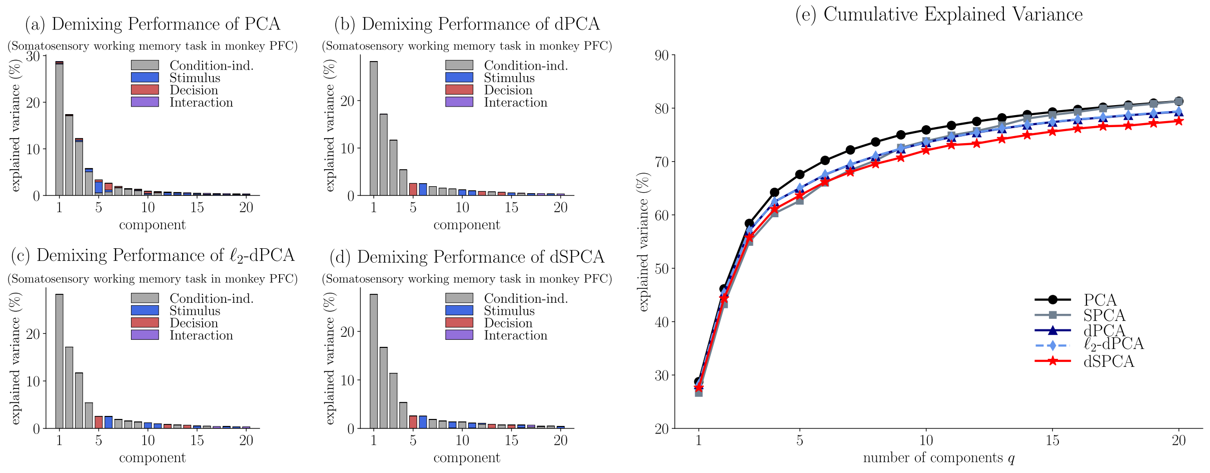

FIGURE 8. Explained variance for real-world data. We recorded the individual explained variance and cumulative explained variance by varying the number of components. Subfigure (a)-(d) shows the individual explained variance on dSPCA and other baseline methods. Each bar represents the corresponding proportion of total variance. The blue bar is the stimulus-influenced variance; the red bar is the decision-influenced variance; the gray bar is the condition-independent variance, and the purple bar is the interaction between stimuli and decisions. SPCA methods are not included since the related algorithms fail to give the decoder **D**, the individual explained variance cannot be obtained. Subfigure (e) shows the cumulative explained variance on dSPCA and other baseline methods. Here we include SPCA since it offers **XD** and **F** to obtain the cumulative explained variance.

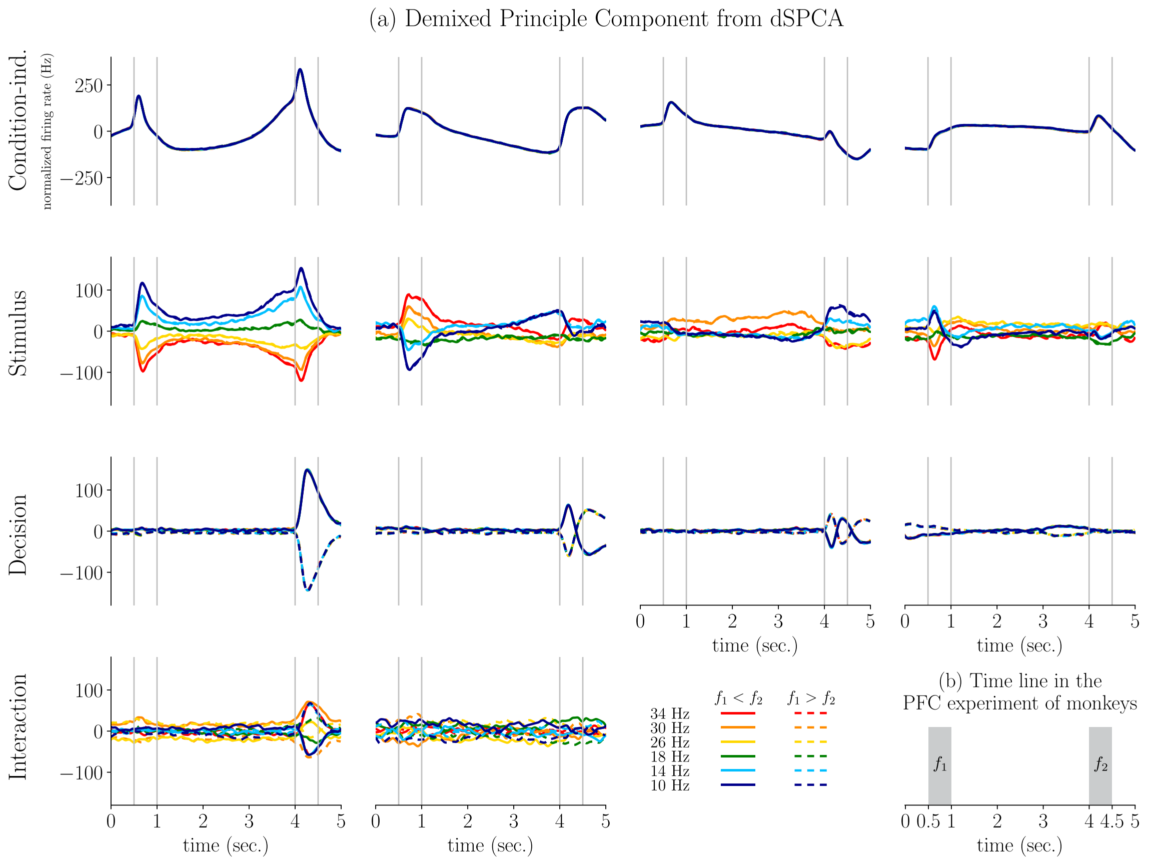

FIGURE 9. Demixed principal components for dSPCA. In Subfigure (a), the first column represents the first principal component and so does the second, third and fourth column. Each curve in the subplots represents this kind of setting. We use six colors to represent six different stimuli and two kinds of line styles to represent the two different decisions. Subfigure (b) shows the time line in the PFC experiment of monkeys.

In this dataset, the monkeys are asked to make a discrimination between two different stimuli f_1 and f_2 with a frequency of $\{10, 14, 18, 26, 30, 34\}$Hz, where f_2 was given 3 seconds after f_1. The time line is shown in Figure 9(b). Monkeys needed to decide whether $f_1 > f_2$ and press the corresponding button. Then the time-dependent firing rate of neurons, which stands for the responses of the neurons, are recorded.

We ignore the overall 6% of the error cases and only consider the correct trials. Then we use a Gaussian kernel to filter the data. The firing rates used in the experiments are sampled at 100Hz from the Gaussian kernel filtered data.

For better analysis, we ignored neurons with missing settings. Also, to improve the reliability of our datasets, we left the neurons a small number of trials $n \geq 5$ and use the averages over trials as the corresponding firing rates. After neuron selection, 793 neurons are left and for each setting, 501 firing rates are recorded.

The whole working memory task in the PFC of monkeys involves two task parameters: stimulus s (out of S) and decisions d (out of H). We use p to represent the number of neurons. For each setting, the number of recorded firing rates is T. Then the neural firing rate data can be represented as $\mathbf{X} \in \mathbb{R}^{SHT \times p}$. With the assumption of our proposed method, the each sample is composed by $\mathbf{x} = \mathbf{x}_t + \mathbf{x}_s + \mathbf{x}_d + \mathbf{x}_{st} + \mathbf{x}_{dt} + \mathbf{x}_{sd} + \mathbf{x}_{sdt}$. Since the neurons behave related closely to time parameters, we combine some of the terms to obtain a more reasonable representation in results: Conditional-independent term denotes $\mathbf{x}_t$; Stimulus term denotes $\mathbf{x}_s + \mathbf{x}_{st}$; Decision term denotes $\mathbf{x}_d + \mathbf{x}_{dt}$; Interaction term denotes $\mathbf{x}_{sd} + \mathbf{x}_{sdt}$. The names of the terms are the same in other result representations with respect to marginalizations.

Figure 8(a)-(d) shows that the demixing ability of dSPCA greatly exceeds that of PCA and is similar to other dPCA based method. SPCA related methods are not mentioned, the reason is the same as that in Figure 5. Figure 8(e) shows that the cumulative variance explained by dSPCA is very close to the cumulative variance explained by other baseline methods. Here we include SPCA since it offers **XD** and **F** to obtain the cumulative explained variance. From these two results, we can infer that dSPCA reserves the useful information through hybrid regularization as well as other PCA based methods, while it separates the dependencies of the multivariate data on different task parameters.

Table 4 illustrates that dSPCA outperforms baseline methods on sparsity of the demixed data. We do not include PCA or SPCA related methods since they are not applicable in the context of demixing, too. The results highly correspond to our expectations since dSPCA uses the ℓ_1-norm and the nuclear norm to add sparse and low-rank regularization to the optimization problem.

Figure 9 shows that both stimuli and decisions are separated clearly at the spike level. From the condition-independent dimension, the spikes with different settings behave consistently in the time series. Since the interaction combines both stimulus and decision information, it seems to be a bit messy, but spikes at the same the stimulus, with different decisions, still behave symmetrically.

TABLE 4. Sparsity for different methods. We recorded the proportion of zero elements to represent the sparsity of the reconstructed data. We only compare the dPCA related methods since PCA or SPCA related methods are not applicable in the context of demixing.

Margin	dPCA	ℓ_2-dPCA	dSPCA
Condition-ind.	53.82%	73.78%	**80.21%**
Stimulus	50.04%	69.97%	**72.47%**
Decision	49.35%	70.35%	**72.86%**
Interaction	49.81%	70.06%	**72.58%**

VII. CONCLUSION AND FUTURE WORKS

In this paper, we investigated the sparse and low-rank representation problem centered on separating the dependencies over different task parameters, which makes it easier to understand how different task parameters contribute to the explained variance. We proposed our method, dSPCA, to decompose the multivariate data with convex sparse and low-rank regularizations. Our method efficiently applies a parallel proximal algorithm to speed up the optimization for convex regularizers added to a problem. Our method outperforms other SPCA methods in stimulated and real-world experiments with respect to the sparse estimation and obtain a variance that can explain the results in a way that is no worse than baseline methods.

In our future work, we plan to reduce the time complexity of our algorithm since $O(p^3)$ complexity could have problems applying to large-scale problems. Another goal is to update **D**, **F**. Such separate update style may provide more flexibility into our method. Also, we will extend our method in the future to a generalized framework for hybrid constraints in PCA and dPCA problems in order to figure out the hidden general properties.

APPENDIX A
PROOF FOR THEOREM 1

Let $\widehat{\mathbf{W}}$ be the optimal reconstruction matrix generated from Algorithm 1 and $\mathbf{W}^*$ be the true matrix. Let $\Delta_1 = \widehat{\mathbf{W}}_{\|\cdot\|_1} - \mathbf{W}^*$, $\Delta_2 = \widehat{\mathbf{W}}_{\|\cdot\|_*} - \mathbf{W}^*$ where $\widehat{\mathbf{W}}_{\|\cdot\|_1}$ and $\widehat{\mathbf{W}}_{\|\cdot\|_*}$ are optimal solutions of minimizing ℓ_1 norm and nuclear norm correspondingly. Furthermore, denote $\mathcal{R}_1(\cdot) = \|\cdot\|_1$, $\mathcal{R}_2(\cdot) = \|\cdot\|_*$, $\beta_1 = \lambda$, $\beta_2 = \gamma$, $\mathcal{I} = \{1, 2\}$ for notation simplification. The error $\|\Delta\|_F$ where $\Delta = \Delta_1 + \Delta_2$ is bounded as:

$$\begin{aligned}
\|\Delta\|_F^2 &= \langle \Delta, \Delta \rangle_F \\
&= \sum_{\alpha \in \mathcal{I}} \langle \Delta, \Delta_\alpha \rangle_F \\
&\leq \sum_{\alpha \in \mathcal{I}} |\langle \Delta, \Delta_\alpha \rangle_F| \\
&\leq 4 \sum_{\alpha \in \mathcal{I}} \beta_\alpha \mathcal{R}_\alpha(\Pi_{\mathcal{M}_\alpha}(\Delta_\alpha)) \\
&\leq 4 \sum_{\alpha \in \mathcal{I}} \beta_\alpha \Psi_\alpha(\mathcal{M}_\alpha) \|\Pi_{\mathcal{M}_\alpha}(\Delta_\alpha)\|_F,
\end{aligned} \tag{23}$$

where $\langle \cdot, \cdot \rangle_F$ denotes the Frobenius inner product, $\mathcal{M}_\alpha$ denotes the subspace $\mathbf{W}^*$ that lies with regard to $\mathcal{R}_\alpha$ and $\Psi_\alpha(\mathcal{M}_\alpha) = \sup_{\mathbf{U} \in \bar{\mathcal{M}}_\alpha \setminus \{\mathbf{0}\}} \frac{\mathcal{R}_\alpha(\mathbf{U})}{\|\mathbf{U}\|_F}$ is the *subspace compatibility constant* defined in [52]. For an ℓ_1 norm with sparsity assumption **(C1)**, $\Psi_1(\mathcal{M}_1) = \sqrt{k}$. For nuclear norm with low-rank assumption **(C2)**, $\Psi_2(\mathcal{M}_2) = \sqrt{r}$. The second inequality is proven by [21] with Hölder's inequality. Similar to Eq. (23),

$$\|\Pi_{\mathcal{M}_\alpha}(\Delta_\alpha)\|_F^2 \le \|\Delta_\alpha\|_F^2 \le 4\beta_\alpha \Psi_\alpha(\mathcal{M}_\alpha)\|\Pi_{\mathcal{M}_\alpha}(\Delta_\alpha)\|_F. \tag{24}$$

Collect all the pieces and we obtain

$$\begin{aligned}\|\Delta\|_F &\le \sqrt{16\sum_{\alpha\in\mathcal{I}} \beta_\alpha^2 \Psi_\alpha^2(\mathcal{M}_\alpha)} \\ &= 4\sqrt{\lambda^2 k + \gamma^2 r}.\end{aligned} \tag{25}$$

APPENDIX B
PPXA CONVERGENCE GUARANTEE

The parallel proximal algorithm (PPXA) is proposed by [24] as Algorithm 2. The convergency of PPXA is guaranteed by the following theorem:

Algorithm 2: Parallel Proximal Algorithm (PPXA) [24]

```
   Data: $\gamma > 0$, $w_i \in (0,1]$, $i = 1, \dots, m$ subject to
         $\sum_{i=1}^m w_i = 1$, $y_i^0 \in \mathbb{R}^d$.
1  initialize $x_0 = \sum_{i=1}^m w_i y_i^0$
2  for $t = 1, 2, \dots$ do
3      for $i = 1$ to $m$ do
4          $a_i^t = \mathbf{prox}_{\gamma f_i / w_i}(y_i^t)$
5      end
6      $a^t = \sum_{i=1}^m w_i a_i^t$
7      $\lambda_t \in (0, 2)$
8      for $i = 1$ to $m$ do
9          $y_i^{t+1} = y_i^t + \lambda_t(2a^t - x_t - a_i^t)$
10     end
11     $x_{t+1} = x_t + \lambda_t(a^t - x_t)$
12 end
   Result: Sequence $\{x_t\}$.
```

Theorem 3: Let G be the set of solutions to

$$\min_x \sum_{i=1}^m f_i(x), \tag{26}$$

where $f_i(\cdot)$, $i = 1, 2, \dots, m$ are convex functions. Let $\{x_t\}$ be the sequence generated by Algorithm 2 under the following assumptions.

1) $\lim_{\|x\|\to+\infty} \sum_{i=1}^m f_i(x) = +\infty$.
2) $0 \in \mathrm{sri}\{(x - x_1, \dots, x - x_m) \mid x \in \mathbb{R}^d, x_i \in \mathrm{dom} f_i\}$ *where* $\mathrm{sri}\{\cdot\}$ *is the strong relative interior defined in [24].*
3) $\sum_{t\in\mathbb{N}} \lambda_t(2 - \lambda_t) = +\infty$.

Then $G \neq \varnothing$ and $\{x_t\}$ converges weakly to a point in G.

The detailed proof of Theorem 3 can be found in Section 3.3 in [24].

ACKNOWLEDGMENT
The authors would like to thank our advisor, Dr. Beilun Wang, for his guidance in writing skills, and Jiaqi Zhang, for his guidance in improving this work. Moreover, Yan Zhang thanks her best friend Xiao Tan for accompanying and encouraging her.

REFERENCES

[1] R. Romo, C. D. Brody, A. Hernández, and L. Lemus, "Neuronal correlates of parametric working memory in the prefrontal cortex," *Nature*, vol. 399, no. 6735, pp. 470–473, Jun. 1999.

[2] C. D. Brody, A. Hernández, A. Zainos, and R. Romo, "Timing and neural encoding of somatosensory parametric working memory in macaque prefrontal cortex," *Cerebral Cortex*, vol. 13, no. 11, pp. 1196–1207, Nov. 2003.

[3] X.-L. Qi, T. Meyer, T. R. Stanford, and C. Constantinidis, "Changes in prefrontal neuronal activity after learning to perform a spatial working memory task," *Cerebral Cortex*, vol. 21, no. 12, pp. 2722–2732, Dec. 2011.

[4] X.-L. Qi, T. Meyer, T. R. Stanford, and C. Constantinidis, "Neural correlates of a decision variable before learning to perform a match/non-match task," *J. Neurosci.*, vol. 32, no. 18, pp. 6161–6169, May 2012.

[5] T. Meyer, X.-L. Qi, T. R. Stanford, and C. Constantinidis, "Stimulus selectivity in dorsal and ventral prefrontal cortex after training in working memory tasks," *J. Neurosci.*, vol. 31, no. 17, pp. 6266–6276, Apr. 2011.

[6] C. E. Feierstein, M. C. Quirk, N. Uchida, D. L. Sosulski, and Z. F. Mainen, "Representation of spatial goals in rat orbitofrontal cortex," *Neuron*, vol. 51, no. 4, pp. 495–507, Aug. 2006.

[7] A. Kepecs, N. Uchida, H. A. Zariwala, and Z. F. Mainen, "Neural correlates, computation and behavioural impact of decision confidence," *Nature*, vol. 455, no. 7210, pp. 227–231, Sep. 2008.

[8] R. W. Reynolds, N. A. Rayner, T. M. Smith, D. C. Stokes, and W. Wang, "An improved *in situ* and satellite SST analysis for climate," *J. Climate*, vol. 15, no. 13, pp. 1609–1625, Jul. 2002.

[9] A. d'Aspremont, F. Bach, and L. El Ghaoui, "Optimal solutions for sparse principal component analysis," *J. Mach. Learn. Res.*, vol. 9, no. 7, pp. 1269–1294, 2008.

[10] J. Ahn, Y. Yuan, G. Parmigiani, M. B. Suraokar, L. Diao, I. I. Wistuba, and W. Wang, "DeMix: Deconvolution for mixed cancer transcriptomes using raw measured data," *Bioinformatics*, vol. 29, no. 15, pp. 1865–1871, Aug. 2013.

[11] C. Peng, Z. Kang, H. Li, and Q. Cheng, "Subspace clustering using log-determinant rank approximation," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2015, pp. 925–934.

[12] C. Peng, Z. Kang, and Q. Cheng, "Integrating feature and graph learning with low-rank representation," *Neurocomputing*, vol. 249, pp. 106–116, Aug. 2017.

[13] C. Peng, Y. Chen, Z. Kang, C. Chen, and Q. Cheng, "Robust principal component analysis: A factorization-based approach with linear complexity," *Inf. Sci.*, vol. 513, pp. 581–599, Mar. 2020.

[14] I. T. Jolliffe, N. T. Trendafilov, and M. Uddin, "A modified principal component technique based on the LASSO," *J. Comput. Graph. Statist.*, vol. 12, no. 3, pp. 531–547, 2003.

[15] R. Jenatton, G. Obozinski, and F. Bach, "Structured sparse principal component analysis," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 366–373.

[16] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *J. Roy. Statist. Soc., B, Stat. Methodol.*, vol. 67, no. 2, pp. 301–320, 2005.

[17] D. Kobak, W. Brendel, C. Constantinidis, C. E. Feierstein, A. Kepecs, Z. F. Mainen, X.-L. Qi, R. Romo, N. Uchida, and C. K. Machens, "Demixed principal component analysis of neural population data," *eLife*, vol. 5, Apr. 2016, Art. no. e10989.

[18] W. Brendel, R. Romo, and C. K. Machens, "Demixed principal component analysis," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 24, 2011, pp. 2654–2662.

[19] B. Moghaddam, Y. Weiss, and S. Avidan, "Generalized spectral bounds for sparse LDA," in *Proc. 23rd Int. Conf. Mach. Learn. (ICML)*, 2006, pp. 641–648.

[20] H. Zou, T. Hastie, and R. Tibshirani, "Sparse principal component analysis," *J. Comput. Graph. Statist.*, vol. 15, no. 2, pp. 265–286, Jun. 2006.

[21] E. Yang, A. Lozano, and P. Ravikumar, "Elementary estimators for high-dimensional linear regression," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 388–396.

[22] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*. New York, NY, USA: Springer, 2011, pp. 185–212.

[23] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, Sep. 1936.
[24] P. L. Combettes and J.-C. Pesquet, "A proximal decomposition method for solving convex variational inverse problems," *Inverse Problems*, vol. 24, no. 6, 2008, Art. no. 065014.
[25] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics Intell. Lab. Syst.*, vol. 2, nos. 1–3, pp. 37–52, 1987.
[26] J. Chen, K. Li, Z. Tang, K. Bilal, S. Yu, C. Weng, and A. Li, "A parallel random forest algorithm for big data in a spark cloud computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 919–933, Apr. 2017.
[27] V. Mante, D. Sussillo, K. V. Shenoy, and W. T. Newsome, "Context-dependent computation by recurrent dynamics in prefrontal cortex," *Nature*, vol. 503, no. 7474, pp. 78–84, 2013.
[28] D. Hwang, W. A. Schmitt, G. Stephanopoulos, and G. Stephanopoulos, "Determination of minimum sample size and discriminatory expression patterns in microarray data," *Bioinformatics*, vol. 18, no. 9, pp. 1184–1193, Sep. 2002.
[29] P. Müller, G. Parmigiani, C. Robert, and J. Rousseau, "Optimal sample size for multiple testing: The case of gene expression microarrays," *J. Amer. Stat. Assoc.*, vol. 99, no. 468, pp. 990–1001, Dec. 2004.
[30] W.-J. Lin, H.-M. Hsueh, and J. J. Chen, "Power and sample size estimation in microarray studies," *BMC Bioinf.*, vol. 11, no. 1, pp. 1–9, 2010.
[31] M.-L.-T. Lee and G. A. Whitmore, "Power and sample size for DNA microarray studies," *Statist. Med.*, vol. 21, no. 23, pp. 3543–3570, 2002.
[32] N. Yu, M.-J. Wu, J.-X. Liu, C.-H. Zheng, and Y. Xu, "Correntropy-based hypergraph regularized NMF for clustering and feature selection on multi-cancer integrated data," *IEEE Trans. Cybern.*, early access, Jun. 30, 2020, doi: 10.1109/TCYB.2020.3000799.
[33] J. Chen, K. Li, Z. Tang, K. Bilal, and K. Li, "A parallel patient treatment time prediction algorithm and its applications in hospital queuing-recommendation in a big data environment," *IEEE Access*, vol. 4, pp. 1767–1783, 2016.
[34] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 2, no. 4, pp. 433–459, 2010.
[35] J. Shlens, "A tutorial on principal component analysis," 2014, *arXiv:1404.1100*. [Online]. Available: http://arxiv.org/abs/1404.1100
[36] M. Ringnér, "What is principal component analysis?" *Nature Biotechnol.*, vol. 26, no. 3, pp. 303–304, 2008.
[37] T. Bouwmans, A. Sobral, S. Javed, S. K. Jung, and E.-H. Zahzah, "Decomposition into low-rank plus additive matrices for background/foreground separation: A review for a comparative evaluation with a large-scale dataset," *Comput. Sci. Rev.*, vol. 23, pp. 1–71, Feb. 2016.
[38] M.-J. Wu, Y.-L. Gao, J.-X. Liu, C.-H. Zheng, and J. Wang, "Integrative hypergraph regularization principal component analysis for sample clustering and co-expression genes network analysis on multi-omics data," *IEEE J. Biomed. Health Informat.*, vol. 24, no. 6, pp. 1823–1834, Jun. 2020.
[39] X.-T. Yuan and T. Zhang, "Truncated power method for sparse eigenvalue problems," *J. Mach. Learn. Res.*, vol. 14, no. 4, pp. 899–925, 2013.
[40] N. B. Erichson, P. Zheng, K. Manohar, S. L. Brunton, J. N. Kutz, and A. Y. Aravkin, "Sparse principal component analysis via variable projection," *SIAM J. Appl. Math.*, vol. 80, no. 2, pp. 977–1002, Jan. 2020.
[41] M. Journée, Y. Nesterov, P. Richtárik, and R. Sepulchre, "Generalized power method for sparse principal component analysis," *J. Mach. Learn. Res.*, vol. 11, no. 2, pp. 517–553, 2010.
[42] A. d'Aspremont, L. El Ghaoui, M. I. Jordan, and G. R. G. Lanckriet, "A direct formulation for sparse PCA using semidefinite programming," *SIAM Rev.*, vol. 49, no. 3, pp. 434–448, 2007.
[43] B. Moghaddam, Y. Weiss, and S. Avidan, "Spectral bounds for sparse PCA: Exact and greedy algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 18, 2006, p. 915.
[44] B. Moghaddam, Y. Weiss, and S. Avidan, "Fast pixel/part selection with sparse eigenvectors," in *Proc. IEEE 11th Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–8.
[45] H. Shen and J. Z. Huang, "Sparse principal component analysis via regularized low rank matrix approximation," *J. Multivariate Anal.*, vol. 99, no. 6, pp. 1015–1034, Jul. 2008.
[46] C. D. Sigg and J. M. Buhmann, "Expectation-maximization for sparse and non-negative PCA," in *Proc. 25th Int. Conf. Mach. Learn. (ICML)*, 2008, pp. 960–967.
[47] D. M. Witten, R. Tibshirani, and T. Hastie, "A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis," *Biostatistics*, vol. 10, no. 3, pp. 515–534, Apr. 2009.
[48] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?" *J. ACM*, vol. 58, no. 3, pp. 1–37, 2011.
[49] Y. Gao, M. Wu, J.-X. Liu, C.-H. Zheng, and J. Wang, "Robust principal component analysis based on hypergraph regularization for sample clustering and co-characteristic gene selection," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, early access, Mar. 10, 2021, doi: 10.1109/TCBB.2021.3065054.
[50] S. W. Keemink and C. K. Machens, "Decoding and encoding (de)mixed population responses," *Current Opinion Neurobiol.*, vol. 58, pp. 112–121, Oct. 2019.
[51] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," in *Proc. 26th Annu. Int. Conf. Mach. Learn. (ICML)*, 2009, pp. 689–696.
[52] E. Yang and P. Ravikumar, "Dirty statistical models," in *Proc. NIPS*, 2013, pp. 611–619.
[53] S. N. Negahban, P. Ravikumar, M. J. Wainwright, and B. Yu, "A unified framework for high-dimensional analysis of M-estimators with decomposable regularizers," *Stat. Sci.*, vol. 27, no. 4, pp. 538–557, 2012.
[54] S. Boyd, N. Parikh, and E. Chu, *Distributed Optimization and Statistical Learning Via the Alternating Direction Method of Multipliers*. Boston, MA, USA: Now, 2011.
[55] N. A. Steinmetz, P. Zatka-Haas, M. Carandini, and K. D. Harris, "Distributed coding of choice, action and engagement across the mouse brain," *Nature*, vol. 576, no. 7786, pp. 266–273, Dec. 2019.
[56] Z. Kang, C. Peng, and Q. Cheng, "Robust PCA via nonconvex rank approximation," in *Proc. IEEE Int. Conf. Data Mining*, Nov. 2015, pp. 211–220.

YAN ZHANG is currently pursuing the bachelor's degree with the School of Artificial Intelligence, Southeast University, China. Her research interests include dimensionality reduction and model interpretability.

HAOQING XU is currently pursuing the bachelor's degree with the School of Artificial Intelligence, Southeast University, China. His research interests include statistical analysis and large-scale machine learning.