RETHINKING LOCAL LOW RANK MATRIX DETECTION: A MULTIPLE-FILTER BASED NEURAL NETWORK FRAMEWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

The matrix local low rank representation (MLLRR) is a critical dimension reduction technique widely used in recommendation systems, text mining and computer vision. In MLLRR, how to robustly identify the row and column indices that form a distinct low rank sub-matrix is a major challenge. In this work, we first organized the general MLLRR problem into three inter-connected sub-problems based on different low rank properties, namely, LLR-1C, LLR-1, and LLR-r. Existing solutions on MLLRR all leverage problem-specific assumptions and mainly focused on the LLR-1C problem, which lacks the capacity to detect a substantial amount of true and interesting patterns generalizability and prohibits. In this work, we developed a novel multiple-filter based neural network framework, namely FLLRM, which is the first of its kind to solve all three MLLRR problems.We systematically benchmarked FLLRM with state-of-the-art methods on an extensive set of synthetic data, empowered by a robustness evaluation of parameters and theoretical discussions. Experimental results showed that FLLRM outperforms all existing methods and enables a general solution to all the three sub-problems. Experiments on real-world datasets also validated the effectiveness of FLLRM on identifying local low rank matrices corresponding to novel context specific knowledge.

1 INTRODUCTION

Matrix low rank approximation has been widely utilized for dimension reduction and matrix completion in many fields, such as image processing, collaborative filtering, text mining, and biological high throughput omics data analysis [1; 2; 3; 4; 5; 6]. Linear low rank representation approximates a matrix by a low rank matrix generated from a few number of linear bases. Conventional solutions of global low rank representations include truncated singular value decomposition (SVD) and nuclear norm based approaches, which have been widely utilized in data processing and visualization.

Although the global solutions to low rank approximation can capture the major co-variance structure, real world data are always generated from more complicated processes, under which both features and incidences may form subspace structures, where each of the subspace preserves distinctive structures "local" to the subsets of features and incidences. As illustrated in Fig 1, a matrix is generated from the sum of a series of local low rank matrices (Fig 1). One example of such "locality" property is the purchase history data, where a subset of items were purchased under a common reason in a subset of customers, while neither the items or the users sharing a common purchase reason is known [7]. Similarly, in biological single cell RNA-sequencing data, subgroups of genes were regulated by unknown signals possessed by subset of cells, which form local low rank matrices corresponding to different gene coregulation modules [8; 9; 10]. Another example is hyperspectral imaging data, in which hyperspectral pixels of certain geometric structures form distinct local low rank patterns [11; 12]. Compared with a global low rank representation, **M**atrix **L**ocal **L**ow **R**ank **R**epresentation (MLLRR) provides more interpretable patterns hidden in the data enabled by the locality and hence sparse assumptions, as well as more accurate characterization of the data structure by solving the subspace structure encoded in the data.

For an input matrix X, MLLRR aims to identify K submatrices with row and column index set denoted as $I_k \times J_k, k = 1 \dots K$, such that the submatrix $X_{I_k \times J_k}$ has a low matrix rank. Notably,

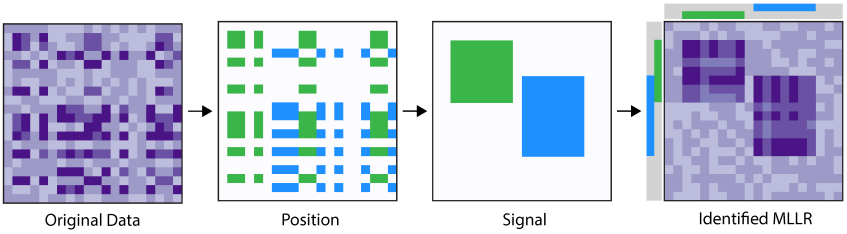

Figure 1: One example of the Matrix Local Low Rank Representation (MLLRR) Problem.

I_k and J_k can be any subset of row and column indices, hence the total number of possible $I_k \times J_k$ is 2^{N+M}, making the MLLRR problem NP-hard [13]. Existing methods for MLLRR include three types: (1) Co-clustering approach identifies submatrices with distinct mean comparing to background [14; 15]. (2) Sparse matrix decomposition based methods approach the problem by sparsifying the pattern matrices with an $\mathcal{L}_1$ penalty [16; 17; 18; 19; 20; 21]. (3) Anchor-based methods first pinpoints local regions using certain primitive similarity measure, and further conducts a low rank fitting to each anchored region [22]. In summary, existing methods circumvent the real challenges in MLLRR by introducing additional assumptions that are only applicable to special cases. There is lack of a general capability to tackle the MLLRR problem, especially for detecting submatrices with strong coherent structure but weak mean signals compared to the background noise. This calls for an urgent and unmet need of a new solution.

The challenges in the general MLLRR problem arise from two aspects: (1) the total number, row and column indices, density and rank of the local low rank matrices are always unknown, and (2) the noise distribution of different local low rank matrices and background are unnecessarily i.i.d (independent and identically distributed), which prohibits a direct application of convex optimization or probabilistic generative model. To tackle the two challenges, we first formally categorized the MLLRR problem into three sub-problems based on their different low rank properties, namely LLR-1C (sub-matrix with a spiked mean), LLR-1 (sub-matrix of rank-1), and LLR-r (sub-matrix of rank-$r, r > 1$), and argued that existing methods failed to offer a general solution to the LLR-1 and LLR-r problem. We generalized the local low rank property of a matrix that there exists a high inner product between each of its row (or column) and a vector close to its top row (or column) bases within a sub-matrix, which enables the identifiability of MLLRR. Based on this consideration, we developed a novel multiple-filter based approach, namely FLLRM (**F**ilter-based **L**ocal **L**ow **R**ank **M**atrix detection). FLLRM adopted the idea of local pattern detection in convolutional neural network, by a hierarchical data sampling, convolution with a set of predefined low rank filters, and classification and prediction of the category of local low rank matrices. We systematically benchmarked FLLRM with state-of-the-arts (SOTA) methods and evaluated its subalgorithms and parameter setting on extensively simulated data and two real-world datasets. FLLRM outperformed all SOTA methods on different MLLRR sub-problems. FLLRM is the only capability can identify the local low matrix whose mean is close to the background, and handle non-Gaussian errors. Application of FLLRM on real world data also detected context meaningful local low rank matrices.

To the best of our knowledge, FLLRM is the first method that provides a general solution to all the aforementioned three MLLRR sub-problems. The key contributions of this work include:

(1) **Generalizing the MLLRR problem**: We generalized the MLLRR problem and its mathematical formulation into three different sub-problems and discussed the progress of existing methods on each sub-problem.
(2) **A novel perspective in local low rank matrix detection**: FLLRM is the first method that utilizes multiple-filter and a neural network architecture to solve the MLLRR problem. Mathematically meaningful importance sampling, low rank filters, and convolutional computation were designed to leverage the detection power and computational feasibility.
(3) **The first general solution of MLLRR**: The FLLRM framework is the first method that provides a general solution for the unsolved LLR-1 and LLR-r problems and can handle different types of error distributions in MLLRR. FLLRM also outperforms SOTA methods on the LLR-1C problem.
(4) **Scalability**: the FLLRM framework can be applied to large scale datasets.

2 PRELIMINARIES

2.1 NOTATIONS

In this study, we denote a matrix with M rows and N columns as $X^{M\times N}$, and its entry of the ith row and jth column as X_{ij}. We denote a set of row and column indices as $I_k \subset \{1, ..., M\}$ and $J_k \subset \{1, ..., N\}$, respectively, and $X_{I_k\times J_k}$ as a submatrix with row and column indices I_k, J_k. For a matrix Z, we use $tSVD^*(Z)$ to denote its most optimal truncated SVD, namely, $tSVD^*(Z) = U\Sigma^{(r)}V^T$, in which U, V are the left and right singular matrix of Z, and $\Sigma^{(r)}$ is the diagonal matrix of Z's singular values, where all except for the top r singular values are forced to be zero. Here r is the total number of non-zero singular values, or the numerical rank of Z.

2.2 MATHEMATICAL FORMULATIONS

The MLLRR problems can be categorized into three sub-problems, namely local low rank 1 with constant mean (LLR-1C), local low rank-1 (LLR-1), and local low rank-r (LLR-r), under the notations of the optimal truncated SVD [23]. Detailed mathematical definitions of the three sub-problems are given below:

Give $X^{M\times N}$, MLLRR identifies submatrices indexed by $I_k \times J_k, k = 1, ..., K, s.t.$

LLR-1C:

$$\begin{aligned} E(X_{ij}) &= u_k, \forall (i,j) \in I_k \times J_k \\ E(X_{ij}) &= u_0, \forall (i,j) \notin I_k \times J_k \end{aligned} \tag{2.1}$$

LLR-1:

$$tSVD^*(X_{I_k\times J_k}) = U_k\Sigma_k^{(1)}V_k^T \tag{2.2}$$

LLR-r:

$$tSVD^*(X_{I_k\times J_k}) = U_k\Sigma_k^{(r)}V_k^T, r > 1 \tag{2.3}$$

Noted, LLR-1 is a special case of LLR-r when $r = 1$. A unique property of LLR-1 is that the row (or column) features in a LLR-1 submatrix are highly linearly correlated as they are generated from the same row (or column) basis. Two types of distribution for the background noise were commonly assumed in matrix low rank representation, namely (1) Gaussian and (2) a mixture of Gaussian and rare component corresponding to outlying entries. Specifically, (1) models the $i.i.d$ Gaussian error in the whole matrix or specific to each column or row, while (2) assumes certain abnormal entries in a submatrix may corrupt its low rank fitting.

2.3 RELATED WORKS

Co-clustering methods simultaneously clusters rows and columns of a two-dimensional data matrix. The general assumption is that the targeted sub-matrix has a mean structure heavily shifted away from zero, comparing to the background noise. The goal of co-clustering is to find a matrix partition defined by $\{I_k, J_k\}_{k=1}^K$ such that the following objective function is minimized: $\sum_k \sum_{i\in I_k, j\in J_k} d(x_{ij}, \mu_k)$. Here μ_k is the mean value of the k-th co-cluster; the distance measure $d(\cdot)$ could have a variety of definitions: in Bregman co-clustering [15], it is defined as the Kullback–Leibler divergence; while in the plaid model, it is defined as the Eucleadian distance [24]. Matrix decomposition based methods identify local low rank matrices by imposing $\mathcal{L}_1$ sparse penalty to the pattern matrices, such as factor matrices [20] and singular matrix [16; 18; 17; 19]. However, both the co-clustering and matrix decomposition based methods assume distinct mean differences between the pattern and background matrix. In addition, they tend to detect large submatrix that may explain better the variance of the whole matrix, while sacrificing the locality of the submatrices [16; 17]. For anchor-based methods, Lee et al. proposed the LLORMA method by using prior knowledge to select anchors of local low rank patterns [25]. As listed in table 1, K_Ω^h is the kernel function with bandwith h to smooth the projection value $P_\Omega(\cdot)$ near the anchor points Ω. However, this type of methods, highly depends on prior knowledge or LLR-1C property of local low rank patterns cannot solve the general MLLRR problem.

Table 1: Existing methods of MLLRR

Methods	Examples	Formulation	Tasks	Assumption	Ref.
Co-clustering	Bregman; Plaid	$\min \sum_k \sum_{i \in I_k, j \in J_k} d(x_{ij}, \mu_k)$	LLR-1C	Matrix partition	[15; 24]
Matrix decomposition	SSVD; PMD	$\min(\|\|X - UV^T\|\|_F^2 + \lambda_u \|\|U\|\|_1 + \lambda_v \|\|V\|\|_1)$,	LLR-1C LLR-1	Sparse patterns	[17; 20]
Anchor based methods	LLORMA WEMAREC	$\min(K_{X[\hat{I},\hat{J}]} \odot P_{X[\hat{I},\hat{J}]}(X - \hat{X}))$, s.t. $rank(X) = 1$	LLR-1	Submatrix detection	[25; 26]

2.4 Computational Challenges

The three MLLRR sub-problems summarized in Section 2.2 are NP-hard [13], which raises the following critical challenges: both row and column indices, the low rank bases, and the total number of the local low submatrices are unknown. Moreover, varied error types such as the Bernoulli outliers may diminish the low rank pattern of the whole sub-matrix. In addition, the high computational intensity of SSVD and anchor based methods prohibits the application to large scale data. Noted, co-clustering and Plaid focus on the LLR-1C problem, while SSVD and anchor-based method also utilize the LLR-1C property to seed possible local low rank matrices. Hence, there is lack of a general and effective solution of the LLR-1 and LLR-r problems in the public domain, especially when the mean of the pattern is similar to the background. The above challenges call for a more robust approach for the general MLLRR problem.

3 FLLRM Algorithm and Mathematical Basis

We develop the FLLRM (Filter based Local Low Rank Matrix detection) framework to robustly solve the three MLLRR sub-problems, inspired by the 'local connectivity' of the convolutional neural network (CNN) [27]. In imaging processing, CNN is capable of exploiting the locality of the images based on convolving sub-parts of the input image with different filters, and then creates representations of small local parts of the input, from where, the larger areas of the images are assembled. Different from images, a matrix is usually unstructured, meaning random permutations of the rows and columns of the matrix doesn't alter the local low rank patterns, and that there is a lack of an analogous proximity measure between the features as in an image.

FLLRM is designed with several key procedures in order to implement a similar convolution idea to capture the locality of an unstructured data matrix. Since there is no natural proximity measure between the features, a full permutation of either the matrix rows and columns or the filters is needed to ensure the identifiability of the local pattern, which is infeasible to large data set. We adopted the idea of random projection (RP) such that the matrix could be randomly cast onto its sub-parts, of which the locality could be better investigated using pre-defined filters [28; 29; 30]. RP works by projecting a high dimensional matrix X into lower dimensions with a given projection matrix, while only slightly distorting the distances between the original data points. The projection matrix could be highly versatile [31; 28; 32], and by carefully designing the random projection matrix, one could preserve the most desirable property of the original matrix (**Lemma** 2). In FLLRM, we implement a two-step matrix sampling approach to first slice a large matrix into medium matrix slices and randomly sample small matrices from each slice, which is equivalent to random projection with highly sparse projection matrix. The mathematical consideration of FLLRM is that a small matrix randomly sampled from a region enriched by a local low rank pattern is more likely to inherit the low rank property, which could be feasibly identified by a set of filters. Instead of training filters from scratch, we use a set of pre-defined filters that could maximally capture the possible patterns of the linear bases. We first illustrate the FLLRM algorithm and further provide mathematical discussions.

3.1 The FLLRM Algorithm

The FLLRM framework is composed of six major steps as illustrated in Fig 2. **Algorithm 1** illustrated the main algorithhm of FLLRM, while all the sub-algorithms are described in APPENDIX. The input of FLLRM is a real matrix $X^{M \times N}$, filter set $\mathcal{F}$, convergence criteria τ, and parameter set Θ. The output is a series of local low rank matrices $X_{I_k \times J_k}$. Specifically, FLLRM first slices (i) $X^{M \times N}$ into a number of $P \times P$ matrix partitions (covering every entry and the slices are not

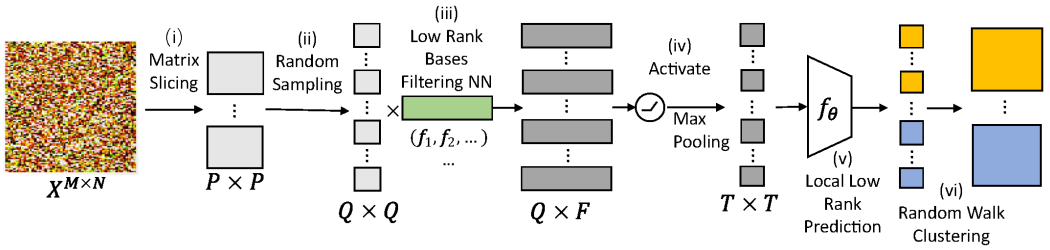

Figure 2: The framework of the FLLRM algorithm.

overlapped), denoted as $\{E_1, \ldots, E_{M_r}\} \times \{F_1, \ldots, E_{N_r}\}$, by using a randomized thresholding SVD based algorithm **Matrix_Slicing** [33]. This step reduces the input data into a number of partitions that ensures the largest local low rank pattern enrich to at least one partition (**Lemma** 1). (ii) R small $Q \times Q$ ($Q \ll P$) matrices are further randomly sampled from each $P \times P$ matrix slice by **Random_Sampling**; (iii) **Low_Rank_Bases_Filtering_NN** computes the inner product between each row and column of each sampled $Q \times Q$ matrix against a set of predefined local low rank filters and (iv) denoise the computed inner products to a $T \times T$ scoring matrix by using convolution and max-pooling functions, (v) **Local_Low_Rank_Prediction** utilizes an unsupervised variaional auto-encoder (VAE) to predict the local low rank property of each $Q \times Q$ matrix based on their scoring matrix, and (vi) **Random_Walk_Clustering** counts the frequency of each row/column index pair in the predicted local low rank $Q \times Q$ matrices and reconstructs the local low rank matrix in X by a spectral bi-clustering algorithm.

Algorithm 1: FLLRM

Inputs: A real matrix $X^{M\times N}$, low rank filter set $\mathcal{F}$, convergence criteria τ, and other parameters Θ

Outputs: The indices set $\{\mathcal{I} \times \mathcal{J}\}$, where $I_k \in \mathcal{I}, J_k \in \mathcal{J}, k = 1...K$, $X_{I_k \times J_k}$ is a low rank matrix.

FLLRM$(X, \mathcal{F}, \tau, \Theta = \{P, r_{MS}, s_{MS}, Q, r\})$:
$\mathcal{I} \leftarrow \varnothing, \mathcal{J} \leftarrow \varnothing, k \leftarrow 0$
while $!\tau$ **do**
  $\boldsymbol{L} \leftarrow \varnothing$
  $\boldsymbol{X} = X_{\{E_1,\ldots,E_{M_r}\}\times\{F_1,\ldots,E_{N_r}\}} \leftarrow$ **Matrix_Slicing**(X, P, r_{MS}, s_{MS})
  $\boldsymbol{Y} \leftarrow \varnothing$
  for *each matrix slice* X_{E_i,F_j} *in* $\boldsymbol{X}$, $E_i \times F_j \in \{E_1, \ldots, E_{M_r}\} \times \{F_1, \ldots, E_{N_r}\}$ **do**
    $X_{E_i,F_j} \leftarrow \dfrac{X_{E_i,F_j} - \textbf{mean}(X_{E_i,F_j})}{\textbf{sd}(X_{E_i,F_j})}$
    $\boldsymbol{Y}_{E_i,F_j} \leftarrow$ **Random_Sampling**(X_{E_i,F_j}, Q, r)
    $append(\boldsymbol{Y}, \boldsymbol{Y}_{E_i,F_j})$
  end
  for *each* $Q\times Q$ *matrix* $\boldsymbol{Y}_p$ *in* $\boldsymbol{Y}$ **do**
    $\boldsymbol{O}_p \leftarrow$ ***Low_Rank_Bases_Filtering_NN***$(\boldsymbol{Y}_p, \mathcal{F})$
    $append(\boldsymbol{L},$ ***Local_Low_Rank_Prediction***$(\boldsymbol{O}_p))$
  end
  LLRM $\leftarrow$ ***Random_Walk_Clustering***$(\boldsymbol{L})$
  for *each local low rank module* $LLRM_p$ *identified in* ***LLRM*** **do**
    $I_k \leftarrow$ *row indices of* $LLRM_p$, $J_k \leftarrow$ *column indices of* $LLRM_p$
    $k++$
    $\mathcal{I} \leftarrow append(\mathcal{I}, I_k), \mathcal{J} \leftarrow append(\mathcal{J}, J_k)$
    $X \leftarrow$ ***Random_Shuffle***$(X, I_k \times J_k)$
  end
end

In **Algorithm 1**, the P, r_{MS}, s_{MS}, Q, and r are parameters of **Matrix_Slicing** and **Low_Rank_Bases_Filtering_NN**, respectively; $\mathcal{I}, \mathcal{J}$ denotes set of the indices of detected local

low rank matrices; **mean** and **sd** represent mean and standard deviation; $\mathbf{Y}_{E_i,F_j}$,$\mathbf{O}_p$, and $\mathbf{L}$ represent the set of $Q \times Q$ matrices randomly sampled from X_{E_i,F_j}, the inner product matrix of each $Q \times Q$ matrix, and the set of scoring matrices of all sampled $Q \times Q$ matrices. τ is the convergence criteria, which can be set as detecting a certain number of patterns or a threshold of the top singular value divided by the nuclear norm. The sub-algorithms **Random_Sampling** conducts a simple random sampling, and **Low_Rank_Bases_Filtering_NN**, **Local_Low_Rank_Prediction** and **Random_Walk_Clustering** adopt existing convolution computation, unsupervised VAE and spectral bi-clustering algorithms. Below we discuss the mathematical considerations of **Matrix_Slicing** and filter set determination. We also provide theoretical derivations to justify the rationale of **Matrix_Slicing** and **Low_Rank_Bases_Filtering_NN** (**Lemma** 1-3) and optimization of parameters and filer set on simulated data. Details of all sub-algorithms are provided in APPENDIX.

3.2 Mathematical Discussions

To the best of our knowledge, FLLRM is the first method using multiple-filter and a neural network architecture to detect local low rank matrices. While substantial number of parameters, filters, and selection of predictors can be adjusted to fit different data and tasks, the most critical settings that determine the detection power are (1) if the matrix slicing procedure can ensure the enrichment of the largest local low rank pattern in at least one matrix slice and (2) if the filters are sensitive enough to detect true local low rank patterns. To justify (1), we derive **Lemma 1**, which suggests that if the largest LLR-1 sub-matrix is large enough, under a proper parameter setting, at least one matrix slice determined by the thresholding SVD approach described in **Matrix_Slicing** will enrich to the sub-matrix. To justify (2), we adopt the **Johnson-Lindenstrauss Theorem** (**Lemma 2**) and note that the LLR-1C and LLR-1 problem can be transformed into a LLR-1 problem of small mean difference betwee the pattern and background noise (**Lemma 3**). Noted, we also justified (1) and (2) by conducting experiments on synthetic data.

Lemma 1. For a real matrix $X^{M\times N}$, if there is at least one LLR-1 sub-matrix in X, denote the size of the largest LLR-1 sub-matrix as $M_0 \times N_0$, if $r_{MS,1} > 1/max\{\frac{M_0}{M}, \frac{N_0}{N}\}^{s_{MS,1}}$ and the other $r_{MS,i}$ are large enough, by expectation at least one matrix slice determined by the thresholding SVD approach described in **Matrix_Slicing** will enrich to the sub-matrix.

Lemma 2 (**Johnson-Lindenstrauss Theorem**, **Dasgupta&Gupta 2003**). For any $0 < \epsilon < 1$ and any integer n, let q be a positive integer such that $q \geq 4\left(\epsilon^2/2 - \epsilon^3/3\right)^{-1} \ln n$. Then for any set V of n point in $\mathbf{R}^p$, there is a map $f : \mathbf{R}^p \to \mathbf{R}^q$ such that for all $u, v \in V$

$$(1-\epsilon)\|u-v\|^2 \leq \|f(u)-f(v)\|^2 \leq (1+\epsilon)\|u-v\|^2$$

Lemma 3. For a given matrix X, if a method can detect $I_k \times J_k$, s.t. $tSVD^*(X_{I_k\times J_k}) = U_k\Sigma_k^{(1)}V_k^T$, under the constraint of **mean**($X_{I_k\times J_k}$)=**mean**(X), it can solve both LLR-1C and LLR-1 problem.

The proof of **Lemma 1,3** are available in APPENDIX and **Lemma 2** was proved in the original paper [30; 29]. To prove **Lemma 1**, we also derived the lower bound of the parameters $s_{MS,1}$ and $r_{MS,1}$ to ensure at least one matrix slice is enriched by the local low rank sub-matrix. **Lemma 2** states that a set of n points in high dimensional Euclidean space can be mapped into an $O\left(\log n/\epsilon^2\right)$ -dimensional Euclidean space such that the distance between any two points changes by only a factor of $(1 \pm \epsilon)$. **Lemma 3** suggests that for any method can detect the LLR-1 matrix with the same mean as the background noise can generally solve all LLR-1C and LLR-1 problem. Specifically, **Lemma 1** suggests the largest local low rank matrix will be enriched to at least one $P \times P$ matrix slice and **Lemma 2** suggests its local low rank property will be further inherited in the $Q \times Q$ matrices randomly sampled from the matrix slice, which could be identified by filters in the low dimension of $\mathbf{R}^Q$. Noted, a larger number of filters could achieve a higher maximal inner product with any vector in $\mathbf{R}^Q$. However, too many filters will decrease the specificity and increase the computational consumption. In addition, controlling the size of filter sets enable a larger Q, which also increase the specificity. **Lemma 3** suggests that only the filters with zero mean need to be considered, since when **mean**($X_{I_k\times J_k}$)=**mean**(X), all the low rank bases of **mean**($Y_{I_k\times J_k}$) in Y=X-**mean**(X) are with a zero mean. **Lemma 2 and 3** largely reduce the number of filters need to be considered and suggest the computational feasibility of FLLRM. Selection of optimal filter set is given in APPENDIX.

Scalability. The computational complexity of **Matrix_Slicing** is $O(max\{M,N\}^3)$. The **Random_Sampling** has a total complexity of $O(MNr)$, where r is the number of submatrices to

be sampled on each $P \times P$ patch. The **Low_Rank_Bases_Filtering_NN** has a total complexity $O(MNrF)$, where F is total number of filters. The complexity of other sub algorithms are much less than **Matrix_Slicing**. Hence the total complexity of FLLRM is $O(max\{M, N\}^3)$. Detailed scalability analysis is provided in APPENDIX.

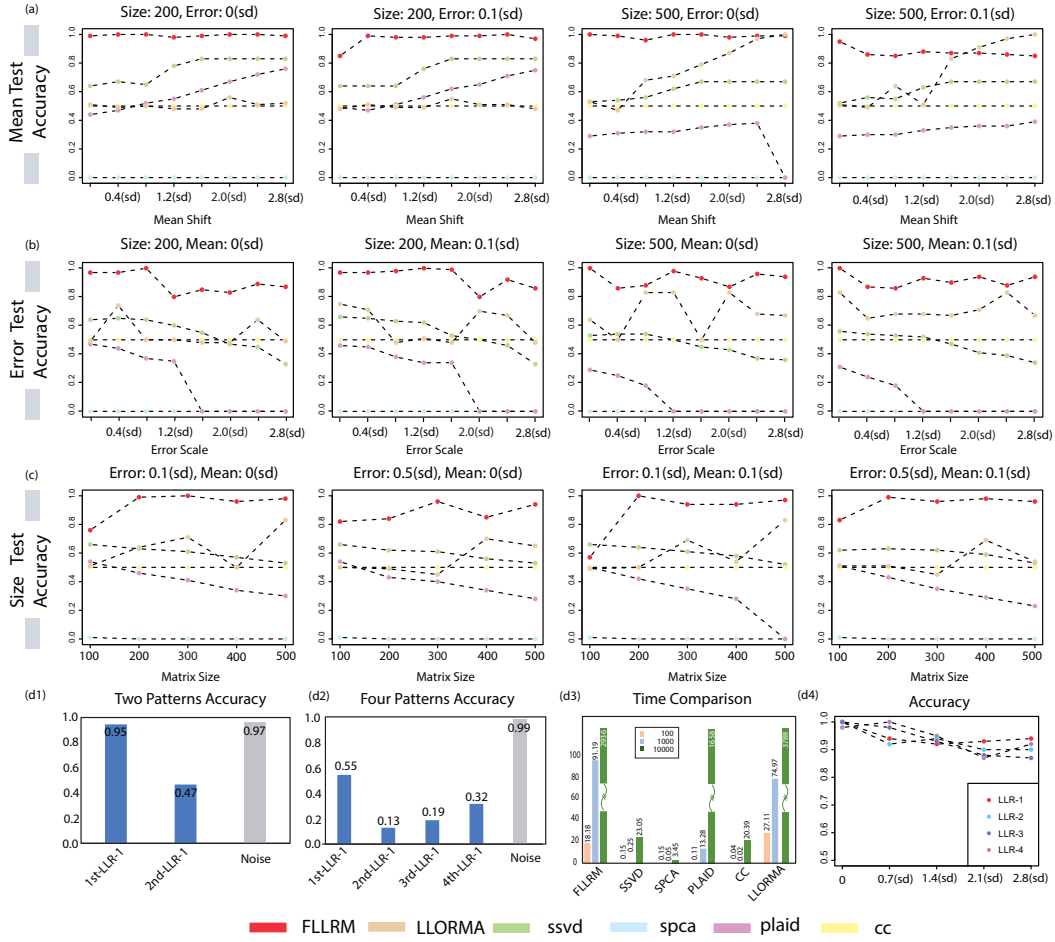

Figure 3: Benchmark of FLLRM on Synthetic Data.

4 Experiments on Synthetic Data

We evaluated the performance of FLLRM and its sub algorithms on synthetic datasets with a comprehensive setting regarding four different aspects: (i) the overall performance of FLLRM on the LLR-1 and LLR-r problem and comparison with SOTA methods; (ii) running time of FLLRM and comparison with SOTA methods; (iii) the effectiveness of the **Matrix_Slicing** algorithm; and (iv) the effectiveness and optimization of filter length and filter sets.

We simulated data with local low rank sub-matrix by the general form: $X = X_{pattern} + X_{noise}$. For (i,iii,iv), X is fixed as a 1000×1000 matrix and $X^{n\times n}_{pattern} = UV^T$. Three different sizes of X were utilized in evaluating the scalability. The pattern is denoted as $X_{I_l J_l}$, where I_l and J_l are the index set of the non-zero entries in $U_{,l}$ and $V_{,l}$, respectively. Each non-zero entry in $U_{,l}$ and $V_{,l}$ is generated from the uniform distribution $U(0, 1)$. The size, mean and error of $X_{pattern}$ were perturbed. Denoting sd as the standard deviation of the background error, we simulated series of data with one rank-1 local low rank matrix: (1) varied pattern mean (0-3$\times$sd) with fixed size (200×200, 500×500) and error (0 and 0.1 times of sd); (2) varied error (0-3$\times$sd) with fixed size (200×200, 500×500) and mean (0 and 0.1 times of sd); (3) varied pattern size (100-500) with fixed mean (0

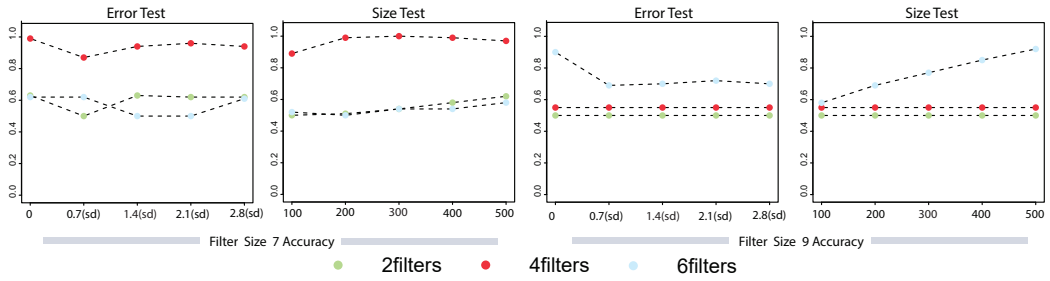

Figure 4: Filter Parameters Experiment.

and 0.1 times of sd) and error (0 and 0.1 times of sd). In total, we achieved 284 different simulation scenarios, each is with 5 replications. See details of experiments in APPENDIX.

4.1 Benchmark of FLLRM on Synthetic Data

We benchmark FLLRM with five SOTA methods, namely Bregman co-clustering (CC) and Plaid, two sparse matrix decomposition methods (SSVD and SPCA), and one anchor based method LLORMA. Specifically, an optimized parameter set P=50, Q=7, τ={the top 10 patterns}, $\mathcal{F}$={full permutation of {-1,-1,0,0,0,1,1}, {-1,-1,-1,0,1,1,1}, {-2,-1,0,0,0,1,2}, {-2,-1,-1,0,1,1,2}} were utilzed for FLLRM. We evaluated the methods based on the accuracy of detecting the true pattern and background noise. Details of parameter optimization of FLLRM, experimental setting of other methods and evaluation metric are provided in 4.2 and APPENDIX.

We first evaluated the performance FLLRM on the LLR-1 problem under the three aforementioned simulation settings. Fig 3a-c illustrated the accuracy (y-axis) of FLLRM (red) and other methods under different simulation parameters. Overall, FLLRM achieved higher than 0.85 accuracy under most settings, which is consistently higher than baseline methods. SPCA failed to identify a pattern while CC recognize the whole matrix as one pattern when analyzing a dense matrix, hence these two methods were excluded in further analysis. The experiment of perturbed mean demonstrated that FLLRM is highly robust to the difference in the mean of the pattern and background noise (Fig 3a), particularly, it is the only method that can identify the pattern when its mean is close to the background mean, i.e., the LLR-1 problem without the LLR-1C property. In contrast, LLORMA, SSVD and Plaid cannot detect the pattern when the mean difference with respect to the background noise is small and their prediction accuracy increases when the mean difference becomes more distinct. All methods showed a decreased prediction accuracy with respect to the increase of the pattern error. FLLRM and LLORMA are more robust to higher pattern errors comparing to SSVD and Plaid. The size test suggested that FLLRM can accurately identify the pattern when its size of the pattern is larger than 150×150 in a 1000×1000 matrix. On the other hand, the prediction accuracy of FLLRM and LLORMA increases while the the accuracy of SSVD and Plaid decreases when the pattern size increases. An explanation is that the larger pattern is easier to be identified by the **Matrix_Slicing** in FLLRM or the anchoring procedure in LLORMA while the sparse assumption of SSVD and Plaid limits the pattern size. We also evaluated the performance of FLLRM on simulated datasets with 2 and 4 patterns, namely, $K = 2, 4$, on which FLLRM achieved a satisfactory accuracy (Fig 3d1-2). It is noteworthy that FLLRM can iteratively identify and remove the top local low rank pattern from the input matrix, which enables a comprehensive detection of multiple patterns. We evaluated the time consumption of the methods on dense matrices of three sizes, $10^2 \times 10^2$, $10^3 \times 10^3$, and $10^4 \times 10^4$ (Fig 3d3). The running time of FLLRM, Plaid and LLORMA are at a similar level, where FLLRM is faster than LLORMA and slower than Plaid. Here we focus on the LLR-1 problem in a dense matrix. Both LLORMA and Plaid are more efficient on the LLR-1C problem in a sparse matrix. We also evaluated FLLRM on the LLR-r problem with r=2,3,4 (Fig 3d4) and demonstrated that FLLRM has a high robustness in detecting LLR-r patterns.

4.2 Robustness Analysis of Sub Algorithms and Parameter Optimization

We also evaluated the effectiveness of the **Matrix_Slicing**, **Low_Rank_Bases_Filtering_NN** and **Local_Low_Rank_Prediction** algorithms and optimized the filter length and filter sets. We observed

the **Matrix_Slicing** achieved at least one matrix slice enriched to the true pattern under all test settings, suggesting the effectiveness of the sub-algorithm (see details in APPENDIX). Lemma 2, 3 and the mathematical discussion suggested possible optimal filter settings. We further evaluated the prediction accuracy achieved by different filter lengths and sets. To leverage the prediction specificity and computational feasibility, we evaluated six sets of filters of length 7 and 9 on the synthetic data with varied values of pattern mean, error and size (see details in APPENDIX). Fig 4 illustrated the prediction accuracy (y-axis) of the pattern under different setting and parameters. We identified the 7-digit filter set composed by the full permutation of {-1,-1,0,0,0,1,1}, {-1,-1,-1,0,1,1,1}, 1,2}, and {-2,-1,-1,0,1,1,2}, totaling 2,450 filters, achieved more than 0.8 (averaged 0.96) prediction accuracy and consistently outperform other filter settings in all experiments, hence was selected as the optimal filter set.

5 Experimental Results on Real-world Datasets

Single cell RNA-sequencing (scRNA-seq) is a new type of high throughput molecular profiling commonly utilized in biological and biomedical fields [34]. More than 5,000 scRNA-seq data have been generated since 2016. A typical scRNA-seq data is a sparse matrix that measures expression level of $\sim$10,000 genes (rows) in $\sim$5,000 individual cells (columns). Recent system biology analyses suggested that the LLR-1C, LLR-1 and LLR-r matrices in scRNA-seq directly correspond to sub population of cells with distinct functional variation [35; 9]. We applied FLLRM and other SOTA methods on two real-world scRNA-seq data, GSE72056 (melanoma) and GSE103322 (head and neck cancer), the two high quality datasets that are most commonly utilized in testing pattern detection methods on scRNA-seq data [36; 37]. We utilized standardized data normalization protocol and selected the 4000 top expressed genes and the 2000 cells of the top total expression level to build our testing data. Four evaluation were utilized to evaluate the performance of each method, namely (1) the Low Rankness, evaluated by the averaged Pearson correlation between the rows and the first row base, and (2) the Size, of each identified local low rank matrix, (3) the total Coverage Rate of all the identified local patterns, and (4) the Running Time. Detailed background information, processing approach and evaluation metrics of the scRNA-seq data are provided in APPENDIX.

We evaluated the top six local low rank matrices identified by each method. The low rankness of all the patterns identified by FLLRM are consistently higher than the ones detected by other methods (Fig 5a). Specifically, to ensure a fair comparison of the patterns detected by different methods, we extract the 200 rows and 100 columns of the largest absolute value of the first row and column bases of each pattern detected by LLORMA and SSVD, i.e. the rows and columns with strongest low rankness, to ensure the sub-matrices formed by them are consistently smaller than the patterns detected by FLLRM (Fig 5b). We observed the low rankness of these top significant low rank patterns detected by LLORMA and SSVD are also consistently lower than the ones detected by FLLRM (Fig 5a). All the local patterns detected by SPCA only have three columns. Plaid and CC only detected one local pattern, whose low rankness is much lower than the ones detected by FLLRM, LLORMA and SSVD. The pattern detected by CC is much larger than then pattern detected by other methods. Noted, the patterns identified by FLLRM do not only have the highest low rankness, but also achieved the highest total coverage over the input matrix. The FLLRM is with longer running time than SSVD and SPCA but is faster than LLORMA and CC on the two real-world datasets. We also examined the biological meaning of the low rank patterns detected by FLLRM, LLORMA and CC by testing the enrichment of the gene features of each pattern against known biological pathways. Noted, three local low rank matrices correspond to distinct biological functions in cell metabolism, cell proliferation, and antigen presentation are only seen in the patterns detected by FLLRM. With the above evidence, we conclude FLLRM outperform baseline methods in detecting LLR-1 and LLR-r modules on the selected real-world data.

6 Conclusion

We developed FLLRM, the first method that utilizes multiple-filter and a neural network architecture to solve the MLLRR problem. Our experiments demonstrated FLLRM outperforms all baseline methods on the LLR-1 and LLR-r MLLRR problem, on both synthetic and real-world datasets. To the best of our knowledge, FLLRM is the only method that provides a general solution to these two MLLRR tasks.

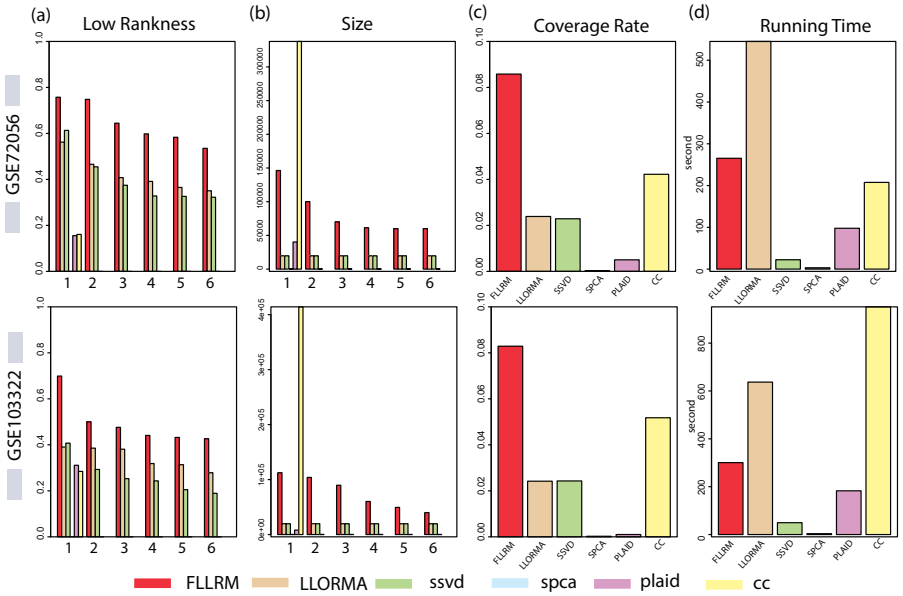

Figure 5: Experiment on real-world data.

REFERENCES

[1] Changlin Wan, Wennan Chang, Tong Zhao, Yong Zang, Sha Cao, and Chi Zhang. Denoising individual bias for a fairer binary submatrix detection. *arXiv preprint arXiv:2007.15816*, 2020.

[2] Farial Shahnaz, Michael W Berry, V Paul Pauca, and Robert J Plemmons. Document clustering using nonnegative matrix factorization. *Information Processing & Management*, 42(2):373–386, 2006.

[3] Volodymyr Kysenko, Karl Rupp, Oleksandr Marchenko, Siegfried Selberherr, and Anatoly Anisimov. Gpu-accelerated non-negative matrix factorization for text mining. In *International Conference on Application of Natural Language to Information Systems*, pages 158–163. Springer, 2012.

[4] Dongsheng Li, Chao Chen, Wei Liu, Tun Lu, Ning Gu, and Stephen Chu. Mixture-rank matrix approximation for collaborative filtering. In *Advances in Neural Information Processing Systems*, pages 477–485, 2017.

[5] Hui Ji, Sibin Huang, Zuowei Shen, and Yuhong Xu. Robust video restoration by joint sparse and low rank matrix approximation. *SIAM Journal on Imaging Sciences*, 4(4):1122–1142, 2011.

[6] Yongyong Chen, Xiaolin Xiao, and Yicong Zhou. Low-rank quaternion approximation for color image processing. *IEEE Transactions on Image Processing*, 29:1426–1439, 2019.

[7] Yao Cheng, Liang Yin, and Yong Yu. Lorslim: low rank sparse linear methods for top-n recommendations. In *2014 IEEE International Conference on Data Mining*, pages 90–99. IEEE, 2014.

[8] Chun-Qiu Xia, Ke Han, Yong Qi, Yang Zhang, and Dong-Jun Yu. A self-training subspace clustering algorithm under low-rank representation for cancer classification on gene expression data. *IEEE/ACM transactions on computational biology and bioinformatics*, 15(4):1315–1324, 2017.

[9] Changlin Wan, Wennan Chang, Yu Zhang, Fenil Shah, Xiaoyu Lu, Yong Zang, Anru Zhang, Sha Cao, Melissa L Fishel, Qin Ma, et al. Ltmg: a novel statistical modeling of transcriptional expression states in single-cell rna-seq data. *Nucleic acids research*, 47(18):e111–e111, 2019.

[10] Wennan Chang, Changlin Wan, Yong Zang, Chi Zhang, and Sha Cao. Supervised clustering of high dimensional data using regularized mixture modeling. *arXiv preprint arXiv:2007.09720*, 2020.

[11] Wei He, Hongyan Zhang, Liangpei Zhang, and Huanfeng Shen. Total-variation-regularized low-rank matrix factorization for hyperspectral image restoration. *IEEE transactions on geoscience and remote sensing*, 54(1):178–188, 2015.

[12] Xuelong Li, Guosheng Cui, and Yongsheng Dong. Graph regularized non-negative low-rank matrix factorization for image clustering. *IEEE transactions on cybernetics*, 47(11):3840–3853, 2016.

[13] Larry J Stockmeyer. *The set basis problem is NP-complete*. IBM Thomas J. Watson Research Division, 1975.

[14] Changlin Wan, Wennan Chang, Tong Zhao, Mengya Li, Sha Cao, and Chi Zhang. Fast and efficient boolean matrix factorization by geometric segmentation. *arXiv*, pages arXiv–1909, 2019.

[15] Arindam Banerjee, Inderjit Dhillon, Joydeep Ghosh, Srujana Merugu, and Dharmendra S Modha. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. *Journal of Machine Learning Research*, 8(Aug):1919–1986, 2007.

[16] Dan Yang, Zongming Ma, and Andreas Buja. A sparse singular value decomposition method for high-dimensional data. *Journal of Computational and Graphical Statistics*, 23(4):923–942, 2014.

[17] Haipeng Shen and Jianhua Z Huang. Sparse principal component analysis via regularized low rank matrix approximation. *Journal of multivariate analysis*, 99(6):1015–1034, 2008.

[18] Mihee Lee, Haipeng Shen, Jianhua Z Huang, and JS Marron. Biclustering via sparse singular value decomposition. *Biometrics*, 66(4):1087–1095, 2010.

[19] Martin Sill, Sebastian Kaiser, Axel Benner, and Annette Kopp-Schneider. Robust biclustering by sparse singular value decomposition incorporating stability selection. *Bioinformatics*, 27(15):2089–2097, 2011.

[20] Daniela M Witten, Robert Tibshirani, and Trevor Hastie. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10(3):515–534, 2009.

[21] Sepp Hochreiter, Ulrich Bodenhofer, Martin Heusel, Andreas Mayr, Andreas Mitterecker, Adetayo Kasim, Tatsiana Khamiakova, Suzy Van Sanden, Dan Lin, Willem Talloen, et al. Fabia: factor analysis for bicluster acquisition. *Bioinformatics*, 26(12):1520–1527, 2010.

[22] Joonseok Lee, Seungyeon Kim, Guy Lebanon, Yoram Singer, and Samy Bengio. Llorma: Local low-rank matrix approximation. *The Journal of Machine Learning Research*, 17(1):442–465, 2016.

[23] Sergey Solovyev and Sébastien Tordeux. An efficient truncated svd of large matrices based on the low-rank approximation for inverse geophysical problems. *Sibirskie Elektronnye Matematičeskie Izvestiâ*, 12:592–609, 2015.

[24] Laura Lazzeroni and Art Owen. Plaid models for gene expression data. *Statistica sinica*, pages 61–86, 2002.

[25] Joonseok Lee, Seungyeon Kim, Guy Lebanon, and Yoram Singer. Local low-rank matrix approximation. In *International conference on machine learning*, pages 82–90, 2013.

[26] Chao Chen, Dongsheng Li, Yingying Zhao, Qin Lv, and Li Shang. Wemarec: Accurate and scalable recommendation through weighted and ensemble matrix approximation. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 303–312, 2015.

[27] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.

[28] Ella Bingham and Heikki Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 245–250, 2001.

[29] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003.

[30] Sanjoy Dasgupta. Experiments with random projection. *arXiv preprint arXiv:1301.3849*, 2013.

[31] Dimitris Achlioptas. Database-friendly random projections. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 274–281, 2001.

[32] Ping Li, Trevor J Hastie, and Kenneth W Church. Very sparse random projections. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 287–296, 2006.

[33] Dan Yang, Zongming Ma, and Andreas Buja. A sparse singular value decomposition method for high-dimensional data. *Journal of Computational and Graphical Statistics*, 23(4):923–942, 2014.

[34] Anoop P Patel, Itay Tirosh, John J Trombetta, Alex K Shalek, Shawn M Gillespie, Hiroaki Wakimoto, Daniel P Cahill, Brian V Nahed, William T Curry, Robert L Martuza, et al. Single-cell rna-seq highlights intratumoral heterogeneity in primary glioblastoma. *Science*, 344(6190):1396–1401, 2014.

[35] Wenpin Hou, Zhicheng Ji, Hongkai Ji, and Stephanie C Hicks. A systematic evaluation of single-cell rna-sequencing imputation methods. *bioRxiv*, 2020.

[36] Itay Tirosh, Benjamin Izar, Sanjay M Prakadan, Marc H Wadsworth, Daniel Treacy, John J Trombetta, Asaf Rotem, Christopher Rodman, Christine Lian, George Murphy, et al. Dissecting the multicellular ecosystem of metastatic melanoma by single-cell rna-seq. *Science*, 352(6282):189–196, 2016.

[37] Sidharth V Puram, Itay Tirosh, Anuraag S Parikh, Anoop P Patel, Keren Yizhak, Shawn Gillespie, Christopher Rodman, Christina L Luo, Edmund A Mroz, Kevin S Emerick, et al. Single-cell transcriptomic analysis of primary and metastatic tumor ecosystems in head and neck cancer. *Cell*, 171(7):1611–1624, 2017.

APPENDIX

RETHINKING LOCAL LOW RANK MATRIX DETECTION: A MULTIPLE-FILTER BASED NEURAL NETWORK FRAMEWORK

Anonymous authors
Paper under double-blind review

1 SUB-ALGORITHMS OF FLLRM

In this section, we illustrate the details of each sub algorithm in **FLLRM**. Specifically, the **Algorithm 1-3** are of the **Matrix_Slicing** approach, which partitions the input matrix into non-overlapping $P \times P$ matrices. Our mathematical discussion in the main text suggested that if the largest local low rank sub-matrix is large enough, the matrix slices achieved by **Matrix_Slicing** could ensure the largest local low rank sub-matrix enrich to at least on matrix slice. The **Algorithm 4** randomly samples a large number of small $Q \times Q$ matrices from each matrix , which ensure each entry in the original matrix can be sampled multiple times. The **Algorithm 5-6** utilizes a set of predefined local low rank filters to evaluate the strength of local low rank property of each sampled $Q \times Q$ matrices and predicts their potential hitting to a true local low rank matrix. The **Algorithm 7** identifies and reconstructs the local low rank matrices based on the prediction of local low rank hitting make on the $Q \times Q$ matrices.

1.1 MATRIX SLICING BY RANDOMIZED THRESHOLD SVD

The **Matrix_Slicing** function conducts a matrix partition based on the idea of randomized threshold SVD, among which the top local low rank matrix is enriched to at least one of the small matrix partitions. The input of this algorithm is a real matrix $X^{M\times N}$, the output is the index set of matrix partition $\{E_1, \ldots, E_{M_r}\} \times \{F_1, \ldots, E_{N_r}\}$, where each X_{E_i,F_j} is a $P \times P$ matrix. Specifically, **Matrix_Slicing** first randomly samples $s_{MS,1}$ rows $r_{MS,1}$ times, computes the column base of its top singular value, identifies the top $s_{MS,2}$ columns that are best fitted by the top column base, computes the goodness of fitting as the averaged R-square value of the $s_{MS,2}$ columns, then select the $r_{MS,2}$ sets with the top averaged R-square values among the $r_{MS,1}$ sampled sets (**Determine_Slicing**). Based on this approach, $r_{MS,2}$ matrices of $s_{MS,1}$ rows and $s_{MS,2}$ columns, each of which is with a larger first singular value comparing to the un-selected ones. **Matrix_Slicing** further iteratively conducts this row- or column-wise threshold SVD approach to generate one $P \times P$ matrix, and its row and column indices were set as $\{E_1\} \times \{F_1\}$. The same approach will be conducted on $X_{\{1\ldots M\}\setminus\{E_1\},\{1\ldots N\}\setminus\{F_1\}}$ to compute $\{E_2\} \times \{F_2\}$, and further iteratively compute $\{E_1, \ldots, E_{M_r}\} \times \{F_1, \ldots, E_{N_r}\}$. In this study, we empirically set $P = 50$, thresholding size s_{MS}=$\{5, 10, 25, 50, 50\}$, number of thresholding r_{MS}=$\{1000, 100, 10, 1, 1\}$. Noted, the **Determine_Slicing** algorithm generates matrix slices of a square region. We introduced a **Determine_Slicing_oneside** algorithm to slice the rest part of the matrix. Our mathematical analysis suggested that if the largest local low rank matrix is large enough, it will enrich to at least one matrix slice generated by the **Matrix_Slicing** function.

The input of **Algorithm 1 Matrix_Slicing** is a real matrix $X^{M\times N}$, slicing size and parameters P, r_{MS}, s_{MS} and the output is the matrix slices. Two sub algorithms (detailed below) **Determine_Slicing** and **Determine_Slicing_oneside** were utilized to compute matrix slices by using iterative truncated SVD. $X_{\{E_1,\ldots,E_{M_r}\}\times\{F_1,\ldots,F_{N_r}\}}$ denotes the final matrix slice, in which E_i and F_j are the set of row and column indices of each slice.

Algorithm 2 Determine_Slicing computes the index of the first slice of an input matrix by a randomized thresholding SVD approach. The input of **Determine_Slicing** is a real matrix $X^{M\times N}$,

Algorithm 1: Matrix_Slicing

Inputs: A real matrix $X^{M\times N}$, slicing size P, r_{MS}, s_{MS}.
Outputs: The slices $\mathbf{X} = X_{\{E_1,\ldots,E_{M_r}\}\times\{F_1,\ldots,F_{N_r}\}}$.
$R^{set} \leftarrow$ NULL , $C^{set} \leftarrow$ NULL
$R^C \leftarrow$ {1,...,M}, $C^C \leftarrow$ {1,...,N}
$i \leftarrow 0$
while $R^C != NULL$ & $C^C != NULL$ **do**
  $i++$
  $E_i, F_i \leftarrow$ **Determine_Slicing**$(X_{R^C\times C^C}, P, r_{MS}, s_{MS})$
  $append(R^{set}, E_i), append(C^{set}, F_i)$
  $R^C \leftarrow R^C \setminus E_i$, $C^C \leftarrow C^C \setminus F_i$
end
if $R^C != NULL$ **then**
  $E_{i+1}, ..., E_{M_r} \leftarrow$ **Determine_Slicing_oneside**$(\mathrm{X}_{R^C\times\{1,...,N\}}, C^{set})$
  $append(R^{set}, E_{i+1}, ..., E_{M_r})$
if $C^C != NULL$ **then**
  $F_{i+1}, ..., F_{N_r} \leftarrow$ **Determine_Slicing_oneside**$(\mathrm{X}_{\{1,...,M\}\times C^C}, R^{set})$
  $append(C^{set}, F_{i+1}, ..., F_{N_r})$
$\mathbf{X} = X_{\{E_1,\ldots,E_{M_r}\}\times\{F_1,\ldots,F_{N_r}\}}$

Algorithm 2: Determine_Slicing

Inputs: A real matrix $X^{M\times N}$, slicing size $P, r_{MS}, s_{MS}, r_{MS,|r_{MS}|} = 1, s_{MS,|s_{MS}|} = P$.
Outputs: The row indices E and column indices F of the first slice.
$row^{set} \leftarrow NULL$, $column^{set} \leftarrow NULL$
$i \leftarrow 1$
$row^{set} \leftarrow$ randomly sampling $s_{MS,i}$ rows $r_{MS,i}$ times
while $s_{MS,i} > 1$ **do**
  $rank^{score} \leftarrow NULL$
  $i++$
  $column^{set0} \leftarrow NULL$
  for *each* row_p^{set} *in* row^{set} **do**
    $\mathrm{U}\Sigma V^T \leftarrow$ **SVD**$(\mathrm{X}_{row_p^{set},.})$
    $column_p^{set0} \leftarrow \{j | X_{row_p^{set},j}\ is\ of\ the\ top\ s_{MS,i}\ largest$ **cor**$(X_{row_p^{set},j}, U_{,1})\}$
    $rank_p^{score} \leftarrow$ **mean**(**cor**$(X_{row_p^{set},column_p^{set0}}, U_{,1})$)
  end
  $column^{set} \leftarrow \{column_p^{set0} | rank_p^{score}\ is\ the\ top\ r_{MS,i}\ largest\ rank^{score}\}$
  $rank^{score} \leftarrow NULL$
  $i++$
  $row^{set0} \leftarrow NULL$
  for *each* $column_p^{set}$ *in* $column^{set}$ **do**
    $\mathrm{U}\Sigma V^T \leftarrow$ **SVD**$(\mathrm{X}_{.,column_p^{set}})$
    $row_p^{set0} \leftarrow \{j | X_{j,column_p^{set}}\ is\ of\ the\ top\ s_{MS,i}\ largest$ **cor**$(X_{j,column_p^{set}}, V_{,1})\}$
    $rank_p^{score} \leftarrow$ **mean**(**cor**$(X_{row_p^{set0},column_p^{set}}, V_{,1})$)
  end
  $row^{set} \leftarrow \{row_p^{set0} | rank_p^{score}\ is\ the\ top\ r_{MS,i}\ largest\ rank^{score}\}$
end
$E \leftarrow row^{set}, F \leftarrow column^{set}$

slicing size and parameters P, r_{MS}, s_{MS} and the output is the row indices E and column indices F of the first slice. **Determine_Slicing** is initiated by randomly generates a series of row index set. The algorithm first computes the columns with the top Pearson correlation to the first column base of the submatrix of each row index set, select the subset of index set with top mean of the Pearson correlations, then conduct this approach on the row side and iterative the column/row-wise computation. Through the computation, the number of selected row and column indices in each set increase to P while the number of selected sets reduce to 1, which generate the index set of one slice. The algorithm repeats this approach to generate all index sets of the matrix slicing. In the algorithm, **SVD**, **mean** and **cor** represents computing of singular value decomposition, mean and Pearson correlation coefficients.

Algorithm 3: Determine_Slicing_oneside

Inputs: A real matrix $X^{M\times N}$, row-side slices $row^{set} = \{E_1, \ldots, E_{M_r}\}$ or column-side slices $column^{set} = \{F_1, \ldots, F_{N_r}\}$, and slicing size P.
Outputs: Column-side slice C^{set} or row-side slice R^{set}.

if *Input is row-side* **then**
- $C^{set} \leftarrow NULL$
- $C^C \leftarrow \{1, ..., N\}$
- **while** $C^C \neq NULL$ **do**
  - $X \leftarrow X_{.,C^C}$
  - $rank^{score} \leftarrow NULL$
  - **for** *each* row_p^{set} *in* row^{set} **do**
    - $\mathrm{U}\Sigma V^T \leftarrow$ **SVD**$(\mathrm{X}_{row_p^{set},.})$
    - $column_p^{set0} \leftarrow \{j | X_{row_p^{set},j}\ is\ of\ the\ top\ P\ largest$ **cor**$(X_{row_p^{set},j}, U_{,1})\}$
    - $rank_p^{score} \leftarrow$ **mean**(**cor**$(X_{row_p^{set},column_p^{set0}}, U_{,1}))$
  - **end**
  - $C^0 \leftarrow \{column_p^{set0} | rank_p^{score}\ is\ the\ largest\ rank^{score}\}$
  - $append(C^{set}, C^0)$
  - $C^C \leftarrow C^C \setminus C^0$
- **end**
- **return** C^{set}

if *Input is column-side* **then**
- $R^{set} \leftarrow NULL$
- $R^C \leftarrow \{1, ..., M\}$
- **while** $R^C \neq NULL$ **do**
  - $X \leftarrow X_{R^C,.}$
  - $rank^{score} \leftarrow NULL$
  - **for** *each* $column_p^{set}$ *in* $column^{set}$ **do**
    - $\mathrm{U}\Sigma V^T \leftarrow$ **SVD**$(\mathrm{X}_{.,column_p^{set}})$
    - $row_p^{set0} \leftarrow \{j | X_{j,column_p^{set}}\ is\ of\ the\ top\ P\ largest$ **cor**$(X_{j,column_p^{set}}, V_{,1})\}$
    - $rank_p^{score} \leftarrow$ **mean**(**cor**$(X_{row_p^{set},column_p^{set0}}, V_{,1}))$
  - **end**
  - $R^0 \leftarrow \{row_p^{set0} | rank_p^{score}\ is\ the\ largest\ rank^{score}\}$
  - $append(R^{set}, R^0)$
  - $R^C \leftarrow R^C \setminus R^0$
- **end**
- **return** R^{set}

Algorithm 3 Determine_Slicing_oneside computes the index of the first slice of an input matrix with fixed slice indices of rows or columns. The input of **Determine_Slicing_oneside** is a real matrix $X^{M\times N}$, slicing size P and fixed slice indices or rows or columns. The algorithm iteratively identifies the indices of the unfixed side of rows or columns that maximize the averaged correlation between its first column or row base and the data, with respect to each fixed slice, i.e. the indices of the unfixed side that maximize the rank-1 property over one slice on the fixed side.

1.2 SAMPLING SMALL MATRICES FOR THE FILTER BASED PREDICTION LOCAL LOW RANK PATTERNS

The computational assumption of FLLRM is that the small $Q \times Q$ matrices sampled from a $P \times P$ matrix slice that is enriched by a true local low rank matrix are with a higher probability to be with a local low rank pattern. The algorithm **Random_Sampling** randomly samples small $Q \times Q$ matrices from each $P \times P$ matrix slice.

Algorithm 4: Random_Sampling

Inputs: A slice X_{E_i,F_j}, sampling size Q, sampling times r.

Outputs: r $Q \times Q$ sub-matrices $\mathbf{Y}^{E_i,F_j} = \{Y^{E_i,F_j}_{1,...,r}\}$ randomly sampled from X_{E_i,F_j}.

$\mathbf{Y}^{E_i,F_j} \leftarrow NULL$

for *k = 1 to r* **do**

  $row_k \leftarrow$ Q indices randomly sampled from E_i

  $col_k \leftarrow$ Q indices randomly sampled from F_j

  $append(\mathbf{Y}^{E_i,F_j}, \{X_{row_k,col_k}, row_k, col_k\})$

end

1.3 PREDICT THE HITTING OF LOCAL LOW RANK PATTERN

If a local low rank pattern is enriched to a $P \times P$ matrix slice, the $Q \times Q$ matrix sampled from the $P \times P$ matrix is with a higher probability to have certain rows and columns hit the local low rank pattern, i.e. resulting in consistent high row-wise inner product with the filters that are close to its true linear row bases (see details in Mathematical Considerations). To ensure a fair and comprehensive screening, the filters generated by the full permutation of pre-given filter patterns will be utilized. Our mathematical consideration and experiments suggested that $P = 50, Q = 7$ and the full permutation of f_1=$\{-1, -1, 0, 0, 0, 1, 1\}$, f_2=$\{-1, -1, -1, 0, 1, 1, 1\}$, f_3=$\{-2, -1, 0, 0, 0, 1, 2\}$, and f_4=$\{-2, -1, -1, 0, 1, 1, 2\}$ form an optimized filter set. The algorithm **Low_Rank_Bases_Filtering_NN** compute the inner product between each row of a $Q \times Q$ matrix and each filter, and further conducted convolution and max-pooling operation to amplify the signal of local rank hitting and output a scoring matrix O^Y. More larger values in O^Y suggest the $Q \times Q$ is more likely to hit a local low rank pattern. The algorithm **Local_Low_Rank_Prediction** utilizes a variational auto-encoder based unsupervised approach to predict the $Q \times Q$ matrices that hit a local low rank pattern.

Algorithm 5 Low_Rank_Bases_Filtering_NN generate a scoring matrix to characterize the strength of local low rank hitting of each randomly sampled $Q \times Q$ matrix, by computing the inner product between each row of the matrix and a pre-defined filter set and downstream convolution and max-pooling operations. With the inner products computed, it first conducts the convolution of the top $k = Q, Q-1, ..., 1$ largest inner product with respect to each filter. The rational here is that if the $Q \times Q$ matrix hits a local low rank pattern with k columns ($k \leqslant Q$), the inner product between the k columns and the filters that are closest to the top row bases of the local low rank pattern are consistently large, hence their convolution. **Low_Rank_Bases_Filtering_NN** further conducts a max-pooling with respect to groups of the filters. Here the filter groups were determined by a hierarchical clustering by using their cosine similarity. The rational here is that if the $Q \times Q$ matrix hits a local low rank pattern, the convolution of the inner products between the row bases and all the filters that are close to its top row bases should be large, hence large inner products could be seen on multiple filters of a large cosine similarity. The max-pooling can effectively denoise such redundant information.

Algorithm 6 Local_Low_Rank_Prediction utilizes an unsupervised classifier **LR_classifier** to classify the $Q \times Q$ matrices and predict if the ones that are with a significant hit to a local low rank pattern by using the scoring matrix. Here **LR_classifier** is a classifier trained by a VAE model. The detailed structure of the network is an encoder with seven layers and 49-128-64-32-16-8-4-(2,2) neurons in each layer and a decoder with seven layers and 2-4-8-16-32-64-128-49 neurons in each layer.

An unsupervised predictor were trained from a VAE model. We generated a large collection of $P \times P, P = 50$ matrices containing different level of local low rank matrix, with size uniformly

Algorithm 5: Low_Rank_Bases_Filtering_NN

Inputs: A $Q \times Q$ matrix Y, a set of low rank filters F, pre-defined low rank filter groups F^G, $F^G_1 \cup ... \cup F^G_{|F^G|} = F$, and size of the output scoring matrix $O_r \times O_c$.
Outputs: Convoluted local low rank scoring matrix O^Y.
$F^S, O^S \leftarrow Q \times |F|\ matrix$
$F^S_{i,j} \leftarrow Y_{i,.} \cdot F_j$
for $k = 1\ to\ |F|$ **do**
  $F^0 \leftarrow sort(F^S_{.,k})$ #by decreasing order
  for *q = 1 to Q* **do**
    $O^S_{q,k} \leftarrow mean(F^0_{1,...,q})$ #convolution the signal of row-wise local low rank property
  end
end
for $k = 1\ to\ |F^G|$ **do**
  for *q = 1 to Q* **do**
    $O_{q,k} \leftarrow max(O^S_{q,F^G_k})$ #max pooling with respect to pre-defined low rank filter groups
  end
end
Reorder the column of O by the decreasing order of the mean of each column
$O^Y \leftarrow O_{\{1,...,O_r\}\times\{1,...,O_c\}}$ #truncation of less sensitive signals

Algorithm 6: Local_Low_Rank_Prediction

Inputs: The local low rank scoring matrices **O** of a randomly sampled $Q \times Q$ matrices.
Outputs: Predicted low rank hitting level of the $Q \times Q$ matrix.
L $\leftarrow$ **LR_classifier**(O)

ranged from 10×10 to 40×40, and randomly draw $Q \times Q, Q = 7$ matrices from them with known labels of local-low-rank-hitting, to train an unsupervised classifier of the $Q \times Q$ matrices. The low rank sub-matrix (pattern) was simulated by the inter product of two vectors randomly generated from $U(0, 1)$. The simulated data fall into four major categories: (1) local low rank matrix without error, (2) local low rank matrix with low error ranges from $(0 - 0.5)\times$ **sd** of the pattern, (3) local low rank matrix with high error $(0.5 - 1)\times$ **sd** of the pattern, and (4) 50×50 matrix with only back ground noise. The background noise of the 50×50 matrix is a normal distribution with mean equals 0 and σ^2 equals **sd** of the pattern. We processed these simulated 50×50 matrices by our **Random_Sampling**, **Low_Rank_Bases_Filtering_NN**, max-Pooling and denosing steps to compute the $T \times T$ matrices, which serves as the inputs of the VAE model to train the predictor. When applying the predictor in the FLLRM framework, all the $Q \times Q$ matrices are predicted label "1" or label "0". The detailed structure of the network is an encoder with seven layers and 49-128-64-32-16-8-4-(2,2) neurons in each layer and a decoder with seven layers and 2-4-8-16-32-64-128-49 neurons in each layer.

1.4 Reconstruction of Local Low Rank Matrices

FLLRM further utilizes to **Algorithm 7 Random_Walk_Clustering** reconstruct the local low rank matrices. It first counts the times of any **row-row, column-column, row-column** pair that appear in one $Q \times Q$ matrix of a significant hitting to a local low rank pattern predicted by **Algorithm 5-6**. Noted, the indices of a local low rank matrix should have a high counts, which could be identified by a Bi-Clustering approach. Here we utilized the **Spectral_BiClustering** method developed by kluger et al [1] and the python package provided by scikit-learn [2]. With the indices of each possible local low rank matrix identified, the local patterns were ranked by the level of their top singular values normalized by the sum of all singular values. Here the local patterns of the top K significant low rank property or with the top singular values large than a certain threshold form the final output of FLLRM.

Algorithm 7: Random_Walk_Clustering

Inputs: A real matrix $\mathbf{X}^{M\times N}$, the predicted label $\mathbf{L}$ and the row and column indices of $Q \times Q$ sampled matrices $\mathbf{Y}$ (from all $P \times P$ matrix slices), local low rank z.
Outputs: The set of Local Low Rank sub-matrices **LLRM**.
$\mathbf{LLRM} \leftarrow NULL$
$\mathbf{LLR}^{set} \leftarrow NULL$
for *each class* L_p *in* $\boldsymbol{L}$ of *a local low rank hit* **do**
  $X^L \leftarrow \boldsymbol{0}^{M\times N}$
  for *each* $Y_{p,k}$ *in* $\boldsymbol{Y}_p = \{Q \times Q \; matrices \; whose \; predicted \; label \; is \; L_p\}$ **do**
    $X^L_{row_{index}(Y_{p,k}),column_{index}(Y_{p,k})} + +$
  end
  $append(\boldsymbol{LLR}^{set}, \boldsymbol{Spectral\_BiClustering}(X^L))$
end
$\boldsymbol{Singular}_{normalized} \leftarrow NULL$
for *each* LLR^{set}_p *in* $\boldsymbol{LLR}^{set}$ **do**
  $U\Sigma V^T \leftarrow \boldsymbol{SVD}(\boldsymbol{X}_{row_{index}(LLR^{set}_p),column_{index}(LLR^{set}_p)})$
  $append(\boldsymbol{Singular}_{normalized}, \frac{\Sigma_{1,1}+...+\Sigma_{z,z}}{trace(\Sigma)})$
end
Return *the local low rank patterns of top* $\boldsymbol{Singular}_{normalized}$ *or with* $\boldsymbol{Singular}_{normalized}$ *larger than a certain threshold.*

2 Mathematical Derivations And Considerations

2.1 Comparison of filters in CNN and FLLRM

Note that in CNN, image patterns are identified by training a set of filters. However, despite that the low rank bases and indices of a local low rank matrix are unknown, its low rank property always suggests higher inner products between each of its row or column and the vectors that are close to its linear row or column bases. This indicates that instead of training filters from scratch, we could pre-define a set of filters that could maximally capture the possible patterns of the linear bases. Here we implement a two-step matrix sampling approach to first slice a large matrix into medium matrix slices and randomly sample small matrices from each slice, which is equivalent to random projection with highly sparse projection matrix. The mathematical consideration of FLLRM is that a small matrix randomly sampled from a region enriched by a local low rank pattern is more likely to inherit the low rank property, which could be feasibly identified by a set of pre-defined filters.

2.2 Lemma Proof

Lemma 1. For a real matrix $X^{M\times N}$, if there is at least one LLR-1 sub-matrix in X, denote the size of the largest LLR-1 sub-matrix as $M_0 \times N_0$, if $r_{MS,1} > 1/max\{\frac{M_0}{M}, \frac{N_0}{N}\}^{s_{MS,1}}$ and the other $r_{MS,i}$ are large enough, by expectation at least one matrix slice determined by the thresholding SVD approach described in **Matrix_Slicing** will enrich to the sub-matrix.

Proof. We conduct the following probabilistic derivation: Given a matrix X of size $M \times N$, assume there is a low rank submatrix X_0 of size $M_0 \times N_0$, the number of hits to the row or columns in X_0 in a random sampling of $s_{MS,1}$ rows or columns follows binomial distribution $Bin(s_{MS,1}, a)$ or $Bin(s_{MS,1}, b)$. It then follows that the probability that a random sampling of $s_{MS,1}$ rows or columns could hit more than K rows or columns in X_0 is $1 - \sum_{l=1}^{K} Bin(l; s_{MS,1}, max\{a, b\})$. Here $Bin(n, p)$ denotes the binomial density function with n trials and success probability of p, and $Bin(n_0; n, p)$ denotes the probability of observing n_0 success under n and p. This enables us to derive the lower bound of $s_{MS,1}$ and $r_{MS,1}$ to achieve a significant hit to the true pattern. □

For example, we can derive if $max\{a, b\} > 0.15$, using $s_{MS,1} = 5$, the probability of having $K \geqslant 3$ hits in row or column of the true pattern is 0.0022; and if we set $r_{MS,1} = 1000$, we could expect 2

samples with more than 3 row or column hits to the true pattern. We have empirically justified a 3/5 hits to the true pattern suffices to guarantee that one slice enriches to the true pattern in the threshold SVD computation.

Lemma 3. For a given matrix X, if a method can detect $I_k \times J_k$, s.t. $tSVD^*(X_{I_k \times J_k}) = U_k \Sigma_k^{(1)} V_k^T$, under the constraint of **mean**($X_{I_k \times J_k}$)=**mean**(X), it can solve both LLR-1C and LLR-1 problem.

Proof. When **mean**($X_{I_k \times J_k}$)=**mean**(X), **mean**($X_{I_k \times J_k}$)=**mean**($X \setminus X_{I_k \times J_k}$). Denote Y=X-**mean**(X), **mean**($Y_{I_k \times J_k}$)=0, i.e., the method can identify the local low rank pattern if its mean is 0. On the other side, by Lemma 1, for any LLR-1 problem:$tSVD^*(X_{I_k \times J_k}) = U_k \Sigma_k^{(1)} V_k^T$, by **Lemma 1**, at least one matrix slice will enrich $X_{I_k \times J_k}$, and the mean of the matrix slice forms a approximation of the mean of $X_{I_k \times J_k}$. Denote Y^{ij}=X-**mean**(X_{E_i,F_j}), in Y^{ij} the pattern $Y^{ij}_{I_k \times J_k}$ mean will be 0 and applying the method on all Y^{ij} will ensure the identification of the $I_k \times J_k$. Similarly, for any LLR-1C problem: $E(X_{ij}) = u_k, \forall (i,j) \in I_k \times J_k \; and \; E(X_{ij}) = u_0, \forall (i,j) \notin I_k \times J_k$, the rows in $X_{i,j}, i \notin I_k, j \in J_k$ or the columns in $X_{i,j}, i \in I_k, j \notin J_k$ with smallest variance will form a LLR-1 sub-matrix with $X_{I_k \times J_k}$, whose first row or column base is $(u_k, ..., u_k, u_0, ..., u_0)$, hence can be identified by the method as derived above. □

LLR-1C problem is a special case of LLR-1 problem. We also conducted an empirical analysis by testing the largest R-square value between our selected filter set and a randomly generated linear base of length Q. Our analysis suggested that on average, our selected filter set achieve a 0.91 and 0.84 R-square value to the low rank base generated from $U(0,1)$ and $N(0,1)$, respectively, suggesting the low rank filters can effective capture possible local low rank structure. Detailed evaluation of the low rank filters is given in 4.2.

2.3 Optimization of the filter set and Fairness

As the **Lemma 2 and 3** largely reduce the number of filters need to be considered to the filters in dimension $\mathbf{R}^Q$ and with 0 mean. Due to the presence of a local low rank matrix is lack of fixed structure in the whole matrix, for any filter, its full permutation is needed for a fair computation. Considering the detection power and computational feasibility, we select $Q = 7 or 9$. For $Q = 7$, it can be proved that the full permutation of f_1=$\{-1,-1,0,0,0,1,1\}$ and f_2=$\{-1,-1,-1,0,1,1,1\}$ can have at least 1/sqrt(2) cosine similarity to any vector in $\mathbf{R}^7$ with mean equals 0. Similarly, the filter set of $Q = 7$ can be further expanded to include f_3=$\{-2,-1,0,0,0,1,2\}$ and f_4=$\{-2,-1,-1,0,1,1,2\}$, and further f_5=$\{-5,-1,0,0,0,1,5\}$ and f_6=$\{-5,-2,-1,0,1,2,5\}$. The vector with smallest cosine similarity to the whole filter set can be identified. Similarly, for $Q = 9$, we consider a possible growth of the optimal filter set as f_1=$\{-1,-1,0,0,0,0,0,1,1\}$ and f_2=$\{-1,-1,-1,0,0,0,1,1,1\}$, f_3=$\{-1,-1,-1,-1,0,1,1,1,1\}$ and f_4=$\{-2,-1,0,0,0,0,0,1,2\}$, f_5=$\{-2,-1,-1,0,0,0,1,1,2\}$, f_6=$\{-2,-2,-1,-1,0,1,1,2,2\}$.

With the top informative filter set and its growth can be derived, we further confirmed the optimal filter by synthetic data based experiments. Specifically, we generated an extensive set of synthetic data (as described in the main text), we tested the two length settings, i.e. $Q = 7, 9$ and test the filter set combinations {f1,f2}, {f1,f2,f3,f4}, and {f1,f2,f3,f4,f5,f6} for each length setting. Our experiments suggested $Q = 7$ and the {f1,f2,f3,f4} as the optimal filter setting.

It is noteworthy different filter sets have different modes. Intuitively, the largest inner product between a random vector in $\mathbf{R}^Q$ and the full permutation of $\{-1,-1,-1,0,1,1,1\}$ is larger than the full permutation of $\{-1,-1,0,0,0,1,1\}$, hence a certain weight are given to ensure the inner products of different filter sets are comparable. See details of the derivation of the weight in 3.7.

2.4 Scalability Analysis

The **Matrix_Slicing** targeted the low rank pattern by filtered with a P by P patch. The time complexity of **Determine_Slicing** is $O(\frac{M}{P} \times \frac{N}{P})$ and the complexity of **Determine_Slicing_onside** is $O(\frac{M}{P})$ or $O(\frac{N}{P})$. Since the **Matrix_Slicing** will be conducted multiple times, the complexity of **Matrix_Slicing** is $O((\frac{max\{M,N\}}{P})^3)$=$O(max\{M,N\}^3)$. The **Random_Sampling** has a

complexity of $O(r+1)$, where r is the total sampling times on the P by P patch. The **Low_Rank_Bases_Filtering_NN** and **Max_pooling** step calculated the inner product of the identified bases, where the complexity is $O(F)$ and F is total number of filters. Thus the last three sub-algorithms are efficient enough whose computational complexity can be ignored compared with **Matrix_Slicing**.

We conducted all experiments on the the high performance cluster (HPC) of our institute. The HPC has eight large-memory compute nodes, each with 512 GB of RAM. In support of deep learning applications and research, Carbonate also features 12 GPU-accelerated Lenovo ThinkSystem SD530 deep learning (DL) nodes, each equipped with two Intel Xeon Gold 6126 12-core CPUs, two NVIDIA GPU accelerators (eight with Tesla P100s; four with Tesla V100s), four 1.92 TB solid-state drives, and 192 GB of RAM. FLLRM has a good time and memory performance comparing to SOTA methods. We fix the matrix size as 1000x1000 and 4000x2000, for which FLLRM can generate the final results within 2 minutes.

3 Experiment Details

Detailed experimental parameters, data and analysis are provided below. The original codes of FLLRM and experiments are available at **https://github.com/ICLRanonLab**.

3.1 General Experimental Setting of Synthetic Data Based Tests

We first simulated data with a local low rank sub-matrix by the general form: $X = X_{pattern} + X_{noise}$. Here X is fixed as a 1000×1000 matrix, $X^{n\times n}_{pattern} = U \times V^T$, where $U, V \in \mathbb{R}^{n\times k}$. The pattern is referred as $\sum_{l=1}^{k} X_{I_l J_l}$, where I_l and J_l are the index set of non-zero element in l_{th} column of U, V, respectively. The non-zero index set of each column of U, V is with a fixed size P and each non-zero element follows a uniform distribution on $(0, 1)$, i.e., $U_{il}, V_{jl} | i \in I_l, j \in J_l \sim U(0, 1)$. The input data (simulated data and real data) was further normalized by forcing its mean to 0 by our normalization function:

$$X^{M\times N} = \frac{X^{M\times N} - Mean_{row}(X^{M\times N})}{sd_{row}(X^{M\times N})} \tag{3.1}$$

3.2 Experiment Setting of Baseline Methods

FLLRM is implemented by python 3.6.3 version and the python libraries numpy(1.19.0), pandas(1.4.1), pytorch(1.6.0) etc. For trainning the classifier, 60% of the input data as training data, 20% of the input data as validation data and 20% of the input data as testing data. We trained the neuron network 2048 epochs with 512 batch size and 0.001 learning rate. The Adam optimizer and MSE loss function are used in the training process.

The baseline methods includes SSVD [3], SPCA [4], Plaid [5], CC [6], LLORMA [7]. The first four methods were performed in R enviroment. For SSVD, we used R package ssvd (version 1.0) and default parameters. For SPCA, we used R package sparsepca (version 0.1.2) and default parameters. For two bicluster methods Plaid and CC, we used R package biclust (version 2.0.1). Specifically for Plaid, we set 'background' parameter as True, 'fit.model' parameter as $y \sim m + a + b$ as tutorial. And for CC method, the parameter δ and α was set as 1.0 and 1.5 as tutorial.

For LLORMA, we used the Global LLORMA from author's GitHub[1] with library Tensorflow-GPU 1.4.0, For synthetic data experiments, all experiments were tested in PRE_RANK=1 along with other default parameters, except for time testing for different size of 10000×10000 and 5000×5000 matrices. We applied lower PRE_LARNING_RATE = 2e-5 for lower RMSE score to a more reasonable range during training. In real-world data experiments, the setting used PRE_RANK = 10 along with other parameters in default.

[1] https://github.com/JoonyoungYi/LLORMA-tensorflow

3.3 EVALUATION METRIC

For the simulated data. The Accuracy of detecting the true pattern and the back ground noise is used in our experiments to evaluate the performance of FLLRM and the benchmarks. For one simulated input matrix $X^{M\times N}$, we labeled the true pattern as "1" and the back ground noise as "0". We keep tracking the index of the true pattern, thus we could generate a binary matrix as the ground-truth $GT^{M\times N}$. If we define the output from the methods above to be $X^{M\times N}_{output}$ and change the non-zero elements in $X^{M\times N}_{output}$ to "1". We could get the output binary matrix $GT^{M\times N}_{output}$. Then we can get the criterion of True Positive(TP), True Negative(TN), False Positive(FP), False Negative(FP) and the Accuracy by the following function:

$$TP, FP, FN, TN \leftarrow Confusion\_Matrix(GT^{M\times N}_{output}, GT^{M\times N}) \tag{3.2}$$

$$Accuracy \leftarrow \frac{TP + TN}{TP + TN + FP + FN} \tag{3.3}$$

3.4 SYNTHETIC DATA BASED EXPERIMENT

We tested the following simulation setting, by (1) changing the mean of the pattern from 0-2.8 times of **sd** under the fixed pattern error of 0 and 0.1 **sd** and size $P = 200, 500$, (2) adding different level of errors from 0-2.8 times of **sd** to the pattern under a fixed mean of 0 and 0.1 **sd** and size $P = 200, 500$, and (3) perturbing the pattern size from 100-500 with 50 step size under a fixed mean of 0 and 0.1 **sd** and error of 0.1 and 0.5 **sd**. In total, we achieved 284 different simulation scenarios. In theses tests, the method SPCA only return NA value, so we gave its Accuracy of 0. All "0" Accuracy in the figures are because the methods could only return NA value, they couldn't give an output. The method CC could only output the binary matrix $GT^{M\times N}_{output}$ of all "1", it means that CC could not identify the true pattern at all, but under our Accuracy function TP=1,TN=0,FP=0,FN=1, we gave the Accuracy of 0.5.

As shown in (Fig 3 a-c), FLLRM achieved consistent high Accuracy score with respect to different mean, error and size changes, while SV and plaid only works when the pattern has a distinct mean comparing to the background noise, SPCA and LLORMA did detected any pattern while CC predict all entries as a pattern. Our data clearly suggested FLLRM outperforms other methods on the LLR-1 problem. We further evaluate if the FLLRM can detect multiple patterns through its iterations. We simulated data with two and four patterns, and run FLLRM two and four times on each data set with 100 replicates. On average, the patterns detected by FLLRM hits 95% of the top pattern, 50% of the second pattern and 10% of the third and fourth pattern (Fig 3 d1,d2). We also tested FLLRM on LLR-k problem by setting the local rank-1 pattern, the local rank-2 pattern, the local rank-3 pattern and the local rank-4 pattern (Fig 3 d4). FLLRM successfully identified the local rank-k pattern with the error range from 0-3 times of **sd** and fixed mean 0 and size $P = 500$. We tested that the running time of FLLRM and benchmarks on the different input matrix size($100 \times 100, 1000 \times 1000, 10000 \times 10000$) (Fig 3 d3).

3.5 REAL-WORLD DATA BASED EXPERIMENT

The real application performed on two biomedical datasets, which are melanoma and head and neck cancer scRNA-seq data. We collected these two datasets from Gene Expression Omnibus (GEO) database, with accession ID GSE72056 and GSE103322. The cell type label and sample information provided in the original work were directly utilized. The GSE72056 data is collected on human melanoma tissues. The original paper provided cell classification and annotations including B cells, cancer-associated fibroblast (CAF) cells, endothelial cells, macrophage cells, malignant cells, NK cells, T cells, and unknown cells. The GSE103322 data is collected on head and neck cancer tissues. The original paper provided cell classification and annotations including B cells, dendritic cells, endothelial cells, fibroblast cells, macrophage cells, malignant cells, mast cells, myocyte cells, and T cells. Notably, as indicated by the original work, malignant cells have high intertumoral heterogeneity. Theses two datasets provide us great opportunity to analysis the biological mechanism by identifying the local low rank pattern within data. And the total citation of two work is above 2000.

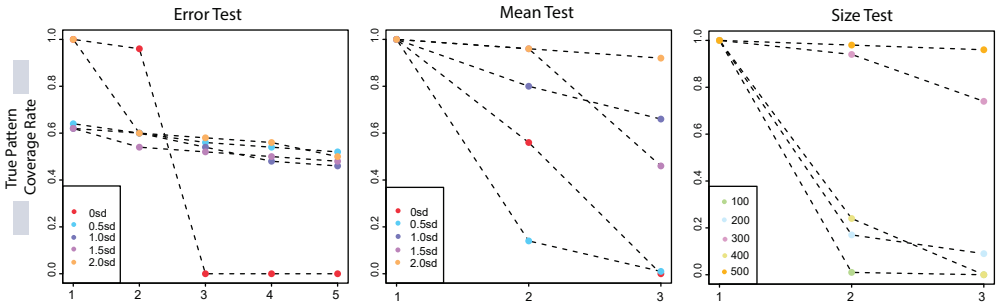

Fig S1. Matrix_Slicing Step Experiment.

Both data sets have been utilized in more than 100 studies, in which GSE72056 contains 23684 genes and 4486 cell, and GSE103322 contains 22494 genes and 5902 cell. We utilize the standardized TPM measure of gene expression level as the input. Firstly, we first conducted a standardized normalization of the data by taking log+1: $GSE \leftarrow log_2(GSE + 1)$. We further selected the 4000 genes (rows) of top averaged expression level and the 2000 cells (columns) of the top total expression level to build our input testing data $GSE^{4000\times2000}$. The Low Rankness, Coverage Size(Size), Coverage Rate, Running Time were used as metrics in our experiments to evaluate the performance of FLLRM and benchmark with other methods. For each sub-matrices calculated by the methods above, we compute its COR rate, Size(how many elements in it) and the coverage of rate of the sub-matrix to the input real data.

The rows represent the genes and the columns represent the cell. Each element in the matrix means the gene expression. The scRNA-seq data is sparse, the zero value in the matrix means the gene is not expressed in the cell. Thus, we just select 4000 rows and 2000 columns by their top mean value in our experiments. Specifically, the scRNA-seq data is the unstructured data, it isn't like the image data, the order of row or column doesn't have special mean. The experiments on scRNA-seq data, our goal is to get the local low rank sub-matrix from these data. The results(Fig 4) show that FLLRM performances great to get the sub-matrices of **LLR-1** structure. The figures of **COR Rate** and **Size** show that FLLRM can get the most obvious **LLR-1** structure sub-matrix and coverage enough sub-matrix size(product of number of row and column). The **Running Time** shows that FLLRM has good time performance. FLLRM consistently identified the local rank-1 pattern under this experimental setting.

3.6 Experimental Setting of the Test of Matrix_Slicing Algorithm

We tested the sub-algorithm **Matrix_Slicing** on an extensive synthetic data set. We tested the following simulation setting, by (1)adding different level of errors from 0-2 times of **sd** to the pattern under a fixed mean of 0 and size $P = 500$, (2) changing the mean of the pattern from 0-2 times of **sd** under the fixed pattern error of 0 and size $P = 500$, and (3) perturbing the pattern size from 100-500 with 100 step size under a fixed mean of 0 and error of 0. In total, we achieved 47 different simulation scenarios and 100 replicates. As shown in (Fig S1), different colored lines represent different level of error in **Error Test**, different mean of the pattern in **Mean Test** and different true pattern size in **Size Test**. The y-axis represents the true pattern coverage rate and the x-axis represents the the order id of the top k coverage rate, ie the first largest rate, the second largest rate etc. On average, some slices generated from the sub-algorithm **Matrix_Slicing** of FLLRM can hit 100% true pattern with respect to different error, mean and size changes.

3.7 Experimental Setting of the Test of Different Filter Sets

We tested that how the size of filter affects the performance of FLLRM. For the filter size of 7 and 9, we tested that the 2 filters, 4 filters and 6 filters are used in the sub-algorithm **Low_Rank_Bases_Filtering_NN** respectively. The details of the setting as bellow:
Filter Size of 7:

We firstly calculate the weights for each filter to balance them by order statistics: $W_7 = Mean_{col}(Sort_{row}(N(0,1) \in R^{1000000*7})), then, W_7 \in R^{1*7}$, $W_7 = [-1.35, -0.76, -0.35, 0.00, 0.35, 0.76, 1.35]$

2 filters:

f_1=$\{-1,-1,0,0,0,1,1\}$, f_2=$\{-1,-1,-1,0,1,1,1\}$, $w_1 = sum(f_1 * W_7)$, $w_2 = sum(f_2 * W_7)$, then $f_1 = f_1$ and $f_2 = f_2 * \frac{w_1}{w_2}$. The number of the full permutation of f_1 and f_2 is 350, thus, if we use 2 filters we can get the filter set $F^{350\times 7}$.

4 filters:

f_3=$\{-2,-1,0,0,0,1,2\}$, f_4=$\{-2,-1,-1,0,1,1,2\}$, $w_3 = sum(f_3 * W_7)$, $w_4 = sum(f_4 * W_7)$, then $f_3 = f_3 * \frac{w_1}{w_3}$ and $f_4 = f_4 * \frac{w_1}{w_4}$. The number of the full permutation of f_1, f_2, f_3 and f_4 is 2450, thus, if we use 4 filters we can get the filter set $F^{2450\times 7}$.

6 filters:

f_5=$\{-5,-1,0,0,0,1,5\}$, f_6=$\{-5,-2,-1,0,1,2,5\}$, $w_5 = sum(f_5 * W_7)$, $w_6 = sum(f_6 * W_7)$, then $f_5 = f_5 * \frac{w_1}{w_5}$ and $f_6 = f_6 * \frac{w_1}{w_6}$. The number of the full permutation of f_1, f_2, f_3, f_4,f_5 and f_6 is 8330, thus, if we use 6 filters we can get the filter set $F^{8330\times 7}$.

Filter Fize of 9:

Similarly, we firstly calculate the weights for each filter to balance them by order statistics: $W_9 = Mean_{col}(Sort_{row}(N(0,1) \in R^{1000000*9})), then, W_9 \in R^{1*9}$, $W_9 = [-1.48, -0.93, -0.57, -0.27, 0.00, 0.27, 0.57, 0.93, 1.48]$

2 filters:

f_1=$\{-1,-1,0,0,0,0,0,1,1\}$, f_2=$\{-1,-1,-1,0,0,0,1,1,1\}$, $w_1 = sum(f_1 * W_9)$, $w_2 = sum(f_2 * W_9)$, then $f_1 = f_1$, $f_2 = f_2 * \frac{w_1}{w_2}$. The number of the full permutation of f_1 and f_2 is 2436, thus, if we use 2 filters we can get the filter set $F^{2436\times 9}$.

4 filters:

f_3=$\{-1,-1,-1,-1,0,1,1,1,1\}$, f_4=$\{-2,-1,0,0,0,0,0,1,2\}$, $w_3 = sum(f_3 * W_9)$ and $w_4 = sum(f_4 * W_9)$, then $f_3 = f_3 * \frac{w_1}{w_3}$ and $f_4 = f_4 * \frac{w_1}{w_4}$. The number of the full permutation of f_1, f_2, f_3 and f_4 is 6090, thus, if we use 4 filters we can get the filter set $F^{6090\times 9}$.

6 filters:

f_5=$\{-2,-1,-1,0,0,0,1,1,2\}$, f_6=$\{-2,-2,-1,-1,0,1,1,2,2\}$, $w_5 = sum(f_5 * W_9)$ and $w_6 = sum(f_6 * W_9)$, then $f_5 = f_5 * \frac{w_1}{w_5}$ and $f_6 = f_6 * \frac{w_1}{w_6}$.The number of the full permutation of f_1, f_2, f_3, f_4, f_5 and f_6 is 43890, thus, if we use 6 filters we can get the filter set $F^{43890\times 9}$.

When Accuracy of detecting the true pattern and back ground noise was used to evaluate its performance. The figures (Fig S2) show that when we use the filter size of 7 and 4 filters in the sub-algorithm **Low_Rank_Bases_Filtering_NN**, the Accuracy can reach over 95%. FLLRM has the best performance.

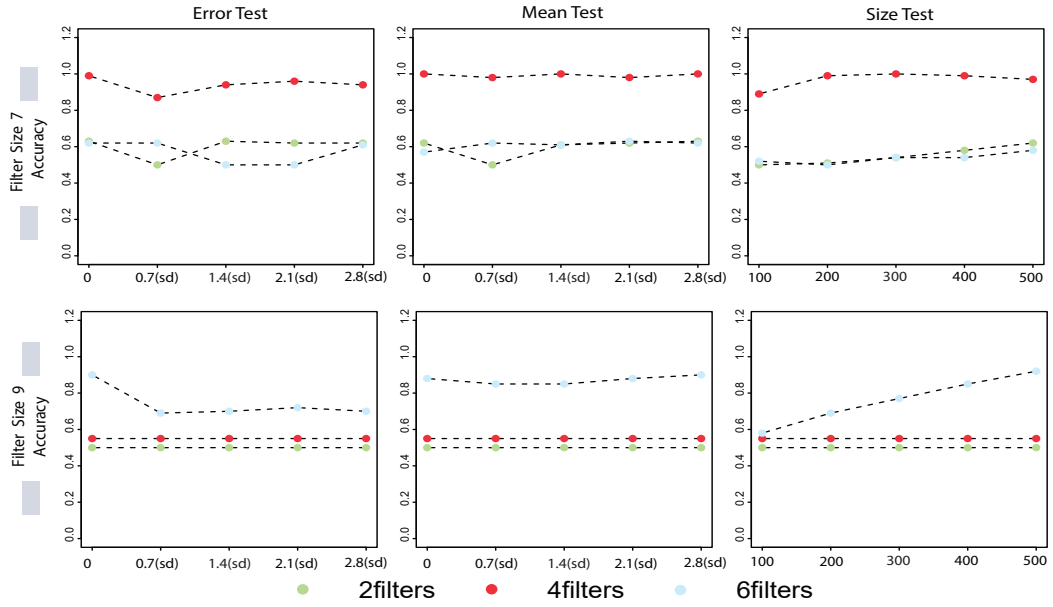

Fig S2. Filter Parameters Experiment.

REFERENCES

[1] Yuval Kluger, Ronen Basri, Joseph T Chang, and Mark Gerstein. Spectral biclustering of microarray data: coclustering genes and conditions. *Genome research*, 13(4):703–716, 2003.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[3] Dan Yang, Zongming Ma, and Andreas Buja. A sparse singular value decomposition method for high-dimensional data. *Journal of Computational and Graphical Statistics*, 23(4):923–942, 2014.

[4] N Benjamin Erichson, Peng Zheng, Krithika Manohar, Steven L Brunton, J Nathan Kutz, and Aleksandr Y Aravkin. Sparse principal component analysis via variable projection. *SIAM Journal on Applied Mathematics*, 80(2):977–1002, 2020.

[5] Laura Lazzeroni and Art Owen. Plaid models for gene expression data. *Statistica sinica*, pages 61–86, 2002.

[6] Arindam Banerjee, Inderjit Dhillon, Joydeep Ghosh, Srujana Merugu, and Dharmendra S Modha. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. *Journal of Machine Learning Research*, 8(Aug):1919–1986, 2007.

[7] Joonseok Lee, Seungyeon Kim, Guy Lebanon, Yoram Singer, and Samy Bengio. Llorma: Local low-rank matrix approximation. *The Journal of Machine Learning Research*, 17(1):442–465, 2016.